Zur Konstruktion algorithmischer Entwurfsräume

Generative Methoden zur gestalterischen Interpretation
von Freiformflächen

Markus Schein

Als Dissertation vorgelegt
dem Fachbereich
Architektur, Design und Kunst der
Bergischen Universität Wuppertal

Mai 2007

Betreuer

Prof. Dr. Dr. h.c. Siegfried Maser
Bergische Universität Wuppertal
Fachbereich Architektur, Design und Kunst

Prof. Hans Dehlinger (PhD)
Kunsthochschule der Universität Kassel
Studiengang Produkt-Design

Herzlichen Dank ...

... an meine beiden Betreuer Siegfried Maser und Hans Dehlinger für ihre Beratung und die Geduld, die sie eine unerwartet lange Zeit für mich aufgebracht haben.

Die vielen Gespräche und Projekte mit meinen Kollegen aus der Architektur, Gregor Zimmermann, Oliver Tessmann und Daan Willems, waren in vielerlei Hinsicht eine wertvolle Hilfe und sind in Form einiger Gedanken und Skripte in diese Arbeit mit eingeflossen.

Danke auch an Ben Kossmann für die Unterstützung bei der Programmierung zweier parametrischer Programmmodule, an Breido Botkus für die Lösung eines anspruchsvollen geometrischen Problems, an Thomas Wortmann für die Anbindung von *VisualBasic*® an *RSTAB*®, die ich für meine Zwecke nutzen konnte und an alle Studierenden des Design und der Architektur, die ‚verrückt' und neugierig genug waren, mit mir gemeinsam in Lehr- und Forschungsprojekten zum generativen und algorithmischen Design zu studieren und dadurch viel zu meinem Verständnis für dieses Entwurfsgebiets beigetragen haben.

Nicht zuletzt einen lieben Dank an Christine Krüger und meine Familie, die mich auf sehr vielfältige Weise unterstützt haben und an all die Übrigen, die meine leichte Neigung zur inneren Emigration in den letzten beiden Jahren verständnisvoll ertragen haben.

Prolog

Mr. Knipe, eine literarische Figur *Roald Dahls*, hält sich als Schriftsteller für ein verkanntes Genie, ist zu seinem Glück aber auch ein begnadeter Ingenieur und vor Allem – mehr als 50 Jahre vor unserer Zeit – der Prototyp eines algorithmischen Entwerfers. Getrieben von tiefer Enttäuschung, die aus der permanenten Ablehnung seines schriftstellerischen Werkes durch unzählige Verleger erwachsen ist, macht er sich an die schwierige, aber letztlich erfolgreiche Entwicklung des großen, automatischen Grammatisators, einer Maschine, die selbsttätig Literatur erzeugt. Seinem Vorgesetzten, *Mr. Bohlen*, gebührt die Ehre, dieses Wunderwerk einzuweihen:

„ ‚Ich habe die Absicht, einen bedeutenden Roman zu schreiben, Knipe.'

‚Ich bin sicher, dass es ihnen gelingt, Sir. Ganz sicher.'

Langsam und bedächtig drückte Mr. Bohlen auf die Vorwahlknöpfe:

Hauptknopf – satirisch,

Thema –Rassenproblem,

Stil – klassisch,

Personen – sechs Männer, vier Frauen, ein Kind,

Umfang – fünfzehn Kapitel.

Zugleich richtete er sein Augenmerk auf die drei Orgelregister Kraft, Geheimnis und Tiefe.

‚Sind Sie bereit, Sir?'

‚Ja, ja, ich bin bereit.'

Knipe betätigte den Schalter. Die große Maschine summte. Fünfzigtausend gut geölte Zahnräder, Stangen und Hebel brachten ein dumpfes, schwirrendes Geräusch hervor; dann begann die schnelle elektrische Schreibmaschine mit einem fast unerträglich lauten Geklapper zu arbeiten. Die vollgeschriebenen Seiten flogen in den Korb – alle zwei Sekunden eine.

‚Ich darf Sie zu Ihrem ersten Roman beglückwünschen', sagte Knipe und nahm das dicke Bündel beschriebener Blätter aus dem Korb. […]

Binnen einer Woche hatte ein begeisterter Verleger das Manuskript gelesen und angenommen. Knipe zog mit einem Roman nach, für den er selbst als Verfasser zeichnete; dann fabrizierte er – da er schon einmal dabei war – ein Dutzend weitere. Bald war Adolph Knipes literarische Agentur berühmt für ihren Stall vielversprechender junger Romanciers. Und wieder begann das Geld hereinzuströmen." (Dahl, 1953) [01]

Oder aber:

‚Ich habe die Absicht, einen bedeutenden Entwurf hervorzubringen, *Knipe*.'

‚Ich bin sicher, dass es ihnen gelingt, *Sir*. Ganz sicher.'

Langsam und bedächtig führte *Mr. Bohlen* den *Cursor* auf die Vorwahlmenüs:

Hauptmenü – Möbel und Inneneinrichtungen,

Thema – Sitzen, Liegen, Entspannen,

Stil – modernistisch, mit einem ornamentalen *Touch*,

Materialien – Edelstahl, Leder, Polyamid, Birnbaum,

Produktionskosten – 450 Euro.

Zugleich richtete er sein Augenmerk auf die Register Zeitlosigkeit, Klarheit, Eleganz und leise Verspieltheit.

‚Sind Sie bereit, *Sir*?'

‚Ja, ja, ich bin bereit.'

Knipe betätigte den Schalter. Ein kurzes Knistern der Prozessoren, das typische Schreibgeräusch der Festplatten, ein kaum merkliches Aufstöhnen der Grafikkarte und Sekunden später waren fünf der wunderbarsten Designklassiker auf den Bildschirmen erschienen. Während *Mr. Bohlen* noch andächtig auf die erzeugten Werke blickte, setzte sich rund um den Globus bereits die digitale Produktionsmaschinerie in Gange, um die virtuellen Schönheiten in die materielle Wirklichkeit zu überführen. Gleichzeitig schossen schon Bilder dieser Maschinenschöpfungen, die realer erschienen als jede physische Erfahrung, mit Lichtgeschwindigkeit um die Welt und füllten all die *blogs, newsgroups,* Datenbanken und sonstigen Mulitplikatorensysteme der digitalen

Gesellschaft. Noch bevor *Knipe* seinen Glückwunsch für die gelungene Arbeit aussprechen konnte und sich der warme Geruch lasergeschnittenen Holzes in den Werkhallen verloren hatte, nannten einige Kunden die ersten Produkte bereits ihr Eigen ...

Man könnte fast sagen, dass *Roald Dahl* ein Art Prophet des algorithmischen Designs war, ohne dass es diesen Begriff zu der Zeit, in der er den großen automatischen Grammatisator geschrieben hat, überhaupt gegeben hätte – ganz zu Schweigen davon, dass damals irgend ein Gerät existierte, welches wir aus heutiger Sicht als *Computer* bezeichnen würden, ohne dabei abschätzig die Nase zu rümpfen. Trotzdem enthält diese Geschichte viele Elemente, die sich auch in der aktuellen Diskussion wiederfinden.

Dahl beschreibt weiter, wie *Knipe* von seinem Chef, *Mr. Bohlen*, zunächst für einen penetranten Spinner gehalten wird, als er bei diesem mit der Idee einer automatischen, interaktiven Bücherschreibmaschine ankommt. Sehr schnell wirft *Mr. Bohlen* seine Bedenken gegenüber den Schwierigkeiten einer solchen Entwicklung über Bord, als er sich der ökonomischen Möglichkeiten bewusst wird, die sich hier eröffnen könnten. *Knipe* wiederum, als Literat ein Versager, tobt seine unbändige Kreativität erfolgreich am Entwurf und der Umsetzung dieser Maschine aus. Er kann sich dabei auch weit genug von seiner Schöpfung distanzieren, um zu erkennen, dass maschinengeschriebene Bücher möglicherweise Unbehagen bei potentiellen Kunden auslösen könnten und dieser Aspekt nicht gerade verkaufsfördernd wäre. Deshalb versucht er, bereits anerkannte Autoren dafür zu gewinnen, ihre Namen als Pseudonyme für die Maschinenbücher zur Verfügung zu stellen. Dafür erntet er zunächst Schimpf und Prügel. Als er aber die erste Autorin überzeugen kann, schmilzt der Widerstand der übrigen Gilde wie Wachs in der Sonne: Die Empörung über die Zumutung, ja die vermeintlich schiere Unmöglichkeit, dass Maschinen maßgeblich an der Erzeugung hoher Literatur beteiligt sein könnten, löst sich auf in der Vorstellung eines sorgenfreien, durch beständigen Tantiemenfluss gesicherten Lebens. Nachdem *Knipe* diese Hürde gemeistert hat, beginnt er, ermutigt vom überwältigenden Verkaufserfolg der Maschinenbücher, mehr und mehr unter seinem eigenen Namen zu veröffentlichen, wird nach und nach immer mutiger und führt auch virtuelle Autoren erfolgreich ein. Zum Schluß bleiben nur ein paar hoffnungslos nostalgische, rückständige Autoren übrig, die sich der Zukunft verschließen. *Dahl* läßt einen dieser Fortschrittsverweigerer das letzte Wort in seiner Kurzgeschichte sprechen: *„Herr, gib mir die Kraft, meine Kinder verhungern zu lassen."*

Heute, mehr als ein halbes Jahrhundert nach dem ersten Erscheinen dieser Geschichte, gibt es weder eine Software, die Literatur hervorbringen könnte, die diese Bezeichnung auch verdient, noch ein Programm, das irgend etwas anderes entwerfen könnte. In dieser Hinsicht greift sowohl *Dahls* Text als auch die oben vorgenommene Abwandlung etwas zu weit in eine ungewisse Zukunft. Gleichzeitig enthalten beide aber auch einige Elemente, die nicht nur schon Realität, sondern darüber hinaus bereits zur Selbstverständlichkeit geworden sind.

Man kann noch nicht so genau abschätzen, wo uns die digitalen Entwurfswelten hinführen werden. Es ist auf jeden Fall wohltuend zu sehen, wie eine ehemals oft dogmatisch geführte Diskussion um das Für oder Wider von Rechnern im Entwurf einer weitaus differenzierteren Betrachtung gewichen ist. Welche praktischen und methodischen, aber auch erkenntnistheoretischen Perspektiven sich aus dem allmählichen Übergang des Computers von der digitalen Zeichenmaschine zum Entwurfsmedium letztlich herausbilden werden, zeichnet sich erst in groben Zügen ab. Trotz eines mittlerweile beachtlichen Spektrums an digital inspirierten Entwurfsarbeiten sind hier die Finger für die ersten Kapitel gerade einmal angefeuchtet.

Es wäre spannend in 50 Jahren zu sehen, was sich ein heutiger *Dahl* wohl ausgedacht hätte …

Abstract – deutsch

Die Geschichte der computerbasierten, generativen Designmethoden ist in etwa so alt, wie die der *CAD*-Systeme selbst. Während Letztere eine nicht mehr weg zu denkende Selbstverständlichkeit im Entwerfen sind, haben Erstere bis vor rund 10 Jahren kaum Spuren in der alltäglichen Entwurfspraxis hinterlassen. Dies stellt sich in letzter Zeit deutlich verändert dar, eingeleitet vor allem durch einige wenige Architekten, die sich konsequent dem generativen Entwerfen verschrieben haben und befördert durch die rasante Entwicklung und Verbreitung computergesteuerter Fertigungsverfahren. Obgleich immer noch in einer eher kleinen Nische beheimatet, hat sich die Frage nach dem Wert und der Relevanz generativer Entwurfsmethoden bereits positiv beantwortet.

Im Spannungsfeld aus etablierten Designansätzen und generativen Methoden eröffnen sich aktuell eine ganze Reihe weiterer Themen, von denen sich Folgende im theoretischen Teil dieser Arbeit wiederfinden: Als Ausgangspunkt wird betrachtet, welche Unterscheidungsmerkmale sich zwischen generativem Entwerfen und traditionell geprägtem Designverständnis finden lassen, wie sich Ersteres in Letzteres einfügen und es erweitern kann und welche methodischen Konsequenzen dies nach sich ziehen wird. Ein wesentlicher Stichpunkt hierzu ist die Unterscheidung zwischen algorithmischem und generativem Entwerfen, also der Entwicklung generischer Werkzeuge, die allgemeines Designwissen repräsentieren und der Anwendung und Erweiterung solchen Wissens im projektspezifischen Kontext.

Weiterhin wird untersucht, wo die Besonderheiten generativer Entwurfsstrategien zu verorten sind, wo ihr praktischer Nutzen liegt und warum sie überwiegend im Zusammenhang mit freien, naturhaft anmutenden Formen Verwendung finden. Es wird diskutiert, wie sich der Umgang mit dem Medium *Computer* wandeln sollte, damit mit Hilfe dieser neuen, noch ungewohnten Methoden auch neue Ergebnisse hervorgebracht werden können. Dies zieht die allgemeine Frage mit sich, wie sich Designdenken im Zusammenspiel mit generativen Entwurfsmethoden ändern könnte – und nicht zuletzt: Welche besonderen gestalterischen Qualitäten mit generativen Werkzeugen und Methoden zu erreichen sind und wie realistisch das häufig geäußerte Ziel ist, über sie neue Entwurfsphilosophien zu entwickeln, die über die Traditionen von euklidisch geprägter Formensprache, additiver Entwurfslogik und einer gedankliche Orientierung an diskreten Einheiten hinausführt.

Die Methodik der Arbeit läßt sich am Treffendsten als ein reflektiertes, ergebnisoffen angelegtes, entwerferisches Selbstexperiment beschreiben: Die im praktischen Teil der Arbeit gewonnen Erfahrungen dienten zur Entwicklung und Überprüfung der theoretischen Erkenntnisse – und umgekehrt. In den anwendungsbezogenen Entwurfsexperimenten wurden aus evolutionären Algorithmen, parametrischen Designansätzen, Methoden der Topologioptimierung sowie der strukturellen Analyse von Stabtragwerken verschiedene generative Entwurfsansätze entwickelt und an einigen Beispielen aus praxisnahen Entwurfskontexten erprobt. Spielwiese hierfür waren Freiformflächen und *Structural Shapes,* also Bauformen, bei der aus der geometrischen Logik von Flächen integrierte, physisch umsetzbare Tragstruktur abgeleitet werden.

Abstract - english

The history of computer aided generative design methods is as far reaching as the history of computer aided drafting and design systems itself. Whereas the latter have gained a major importance in nearly every design process, the former have left only little traces in daily design practice for a considerable amount of time. This started to change about ten years ago when a few architects began to devote their work to generative design methods, accompanied by the rapid development and distribution of computer numeric controlled manufacturing processes. The question, whether generative design methods will play an important role in future design practice is not longer relevant. Nowadays the concern is more about understanding, what and how this role might be.

Currently a couple of new questions are rising in the intermediary field of established design approaches and generative design methods. The following ones amongst them are examined in this thesis: First it tries to describe the relationship between traditional and generative views on design processes and gives some arguments about some of the potentially upcoming methodical consequences. A crucial point here is the distinction between algorithmic and generative design methods which is on one hand the development of generic tools, representing common design knowledge and on the other hand the application and transformation of this knowledge to and by a specific problem in hand.

Further on, it elaborates some of the typical properties of generative design methods, tries to give an idea of potential fields of application and of their practical relevance and explains, why so far the use of generative design methods is mostly related to free, nature like forms. Another issue is how the common habits of dealing with computers should change in favour to bring these new digital design methods to their full performance. This inhibits the question, how design should be thought in order to apply generative methods and vice versa, how the use of these methods influences design thinking. And last but not least, remains the attempt to give some answer to the question, about what kinds of new qualities might be reached throughout generative design practice – generative tools and methods as potential drivers for design philosophies which go beyond the tradition of formal expression based on euclidean geometry, a design logic based on addition of elements and mental models, which tend to understand design problems in terms of discrete elements.

The method of this thesis is best described as a sort of reflected designerly experiment on oneself: The experiences and insights gained from the practical part of this work have served the purpose to develop and to evaluate the theoretical insights – and the other way round: Evolutionary algorithms, parametric design approaches and different structural analysis and optimization techniques have been combined and implemented into the development and programming of new generative design tools and methods. These generative design environments have been tested and evaluated in fields close to realistic design tasks. Free form surfaces and structural shapes have been chosen as playground for experimentation.

Inhaltsverzeichnis

1. Einführung

1.1 Allgemeine Einführung

Der hier verfolgte Ansatz sieht sich im Kontext einer Reihe von jüngeren Forschungsarbeiten und Entwurfsprojekten, in denen versucht wird, generative Designtechniken und -methoden mit digital gesteuerten Fertigungstechnologien zu kombinieren. Solche Entwurfsprozesse, die stark auf das Medium *Computer* setzen, bestechen oft durch eine komplexe und neuartige Methodik, häufig auch durch eine außergewöhnliche Formensprache. So ist es nicht verwunderlich, dass sie gerne als Beleg für den Paradigmenwechsel aufgeführt werden, den der Einzug der Rechner in die Entwurfsarbeit eingeleitet hat. Zweifellos sind die Auswirkungen der digitalen Informationsverarbeitung zahlreich und drastisch, und sie haben dabei auch eindeutig zu einer Erweiterung der gestalterischen Ausdrucksmöglichkeiten geführt – Entwürfe wie das *Kunsthaus Graz* [Abb. 01] oder das *Phaeno* in Wolfsburg [Abb. 02] wären ohne intensiven Computereinsatz in Entwurf und Umsetzung kaum realisierbar.

Abb. [01 / 02]) Ikonen digital inspirierter Formensprache: Das Kunsthaus in Graz (oben) (Cook und Fournier, 2003) sowie das Phaeno in Wolfsburg (Hadid, 2005).

Gleichzeitig wird beim Entwerfen aber auch mehr und mehr gedankliche Leistung daran gebunden, sich im ständig undurchschaubarer werdenden Dschungel aus Programmen, unterschiedlichen Funktionen, Netzwerkstrukturen, Dateiformaten etc. zu orientieren und zu einer souveränen Beherrschung computerbasierter Werkzeuge und Methoden zu finden. Wer hier Pferd und Reiter ist, das ‚Werkzeug' oder sein ‚Bediener', ist nicht mehr in jedem Fall eindeutig zu unterscheiden. Auch die Weiterentwicklung generativer Entwurfstechniken ist unter diesem Blickwinkel durchaus ambivalent zu betrachten. Trotz des Enthusiasmus, der diese neuen Designformen gelegentlich kommentiert, muss die Eigendynamik, die im Wettrüsten aus neuer Möglichkeit und deren Beherrschbarkeit steckt, sorgfältig abgewogen werden. Es ist nicht gänzlich von der Hand zu weisen, dass sich hier eine Entwurfshaltung herausbilden könnte, bei der das *Label* ‚generativ' über eine Bereicherung existierender Designphilosophien hinaus geht und zu einem Selbstzweck wird, dem etablierte Werte selbstverständlich untergeordnet werden.

Die Suche nach neuen, digital inspirierten gestalterischen Potentialen ist eingebettet in einen Prozess der kulturellen Orientierung, der allgemeinen Suche nach einem Umgang mit dem Medium *Computer*: *„Wie behandelt man eine Maschine, wenn sie Rollen übernimmt, die früher Menschen vorbehalten waren, Maschinen deren praktische Unterschiede zu Menschen bei der Ausführung dieser Rollen immer schwerer zu erkennen sind?" (Turkle, 1998)* [01] Dass es im Entwerfen, das eine stark persönlichkeitsbezogene Tätigkeit ist, besonders schwer fällt, sich dieser Frage offensiv zu stellen, ist nachvollziehbar. Hier hat ein ähnlich umwälzender Prozess eingesetzt, wie ihn vor rund 250 Jahren die beginnende Industrialisierung in Gang gebracht und – durch immer stärker werdende Spezialisierung – den Grundstein für die modernen Entwerferberufe gelegt hat. Dass der Übergang zu einer digitalen Entwurfskultur gerade erst begonnen hat, drückt sich etwa dadurch aus, dass digitale Techniken in der Designpraxis bisher überwiegend konservativ, zur Darstellung von Entwurfsideen genutzt werden. Die Anwendungen von Rechnern bewegen sich entlang gewohnter Metaphern

wie Schreibtisch, Zeichenmaschine oder Werkstatt. Sie dienen als Ersatz, zur Ergänzung oder Beschleunigung von Arbeiten, die ehemals nur von Hand ausgeführt wurden. Doch allmählich beginnt man sich im Design daran zu gewöhnen, in Computern mehr als nur Werkzeuge zu sehen. Der Umgang mit den Möglichkeiten dieser Maschine erfährt seit den letzten Jahren eine Veränderung, das Automatisierungspotential der Rechner wird zunehmend in Designprozessen ausgenutzt, es findet ein Blick über die Grenzen hin zur Informatik und den Ingenieursbereichen statt. Generative Entwurfsprozesse sind einer der maßgeblichen Motoren dieses Entwicklung: Hier werden Formen nicht mehr nur im klassischen Sinne entworfen und (über *CAD*-Systeme) gezeichnet, sondern, in Erweiterung dieser über lange Zeit etablierten Tradition, nun auch mit Hilfe generativer, computerbasierter Designprozesse hervorgebracht, untersucht und verändert.

Abb. [03]) Rapid-Prototyping-Techniken, eine der Triebfedern generativer Entwurfsstrategien, ermöglichen individualisierte Designlösungen in einer Formensprache, die ohne solche Herstellungsmöglichkeiten nicht realisierbar wären. Bunnylamp (Vink, 2006) (links) und Hidden (Yeffet, 2006) aus der Materialise.MGX - Kollektion (2006).

Zwar sind solche Entwurfsansätze im Grunde nichts völlig Neues, sie werden schon seit Jahrzehnten erforscht. Aber erst in jüngerer Zeit erlangen sie auch mehr an praktischer Relevanz. Es ist mittlerweile ein Grad an Selbstverständlichkeit im Umgang mit digitalen Entwurfstechniken erreicht, der es erlaubt, sie nicht nur als ausführende Mittel zu begreifen, sondern als eine Umgebung, die selbst zum Zwecke des Entwerfens gestaltet werden kann. Entwicklungen bei den computergesteuerten Fertigungsverfahren lassen ehemals als digitale, unbaubare Spielereien gesehene Entwürfe realisierbar werden. [Abb. 03] Konzepte wie die *Mass Customization*, die anstatt ein Produkt für viele, den individuell angepassten Entwurf propagieren, bieten möglicherweise einen sinnvollen Rahmen für die Leichtigkeit, mit der über generative Ansätze Variationsreichtum zu schaffen ist. Die Pionierarbeit einiger Architekten, die sich dem digitalen Entwerfen verschrieben haben, hat ein breiteres Interesse an der besonderen Anmutung, aber auch an der Methodik geweckt, die mit computerbasierten Formfindungsprozessen einher gehen.

Generative Verfahren, die im Designprozess eine Rolle spielen können, führen eine Vielzahl verschiedener Titel: Kinematische Strukturen, *Diagramming*, *Morphing* über *Keyframe*-Animation in *Maya*®, eine *Software*, die ursprünglich dem Trick- und Animationsfilm vorbehalten war, genetische Algorithmen, digitale Morphogenese, parametrisches Design, *Rewriting Rule Systems*, zellulare Automaten, *Shape Grammars*, usw. Was alle diese unterschiedlichen Ansätze vereint ist, dass die Entwurfsarbeit nicht mehr nur direkt auf das im *CAD*-System sichtbare Entwurfsmodell fokussiert, sondern auf die mittelbare, abstrakte Ebene von Programmiersprachen ausgedehnt ist. Es wird nicht mehr nur eine Form modelliert, sondern das Design von Regelsystemen, von einer generativen Logik, von Computeralgorithmen wandert in den Vordergrund und wird zum Mittel des Design: Entwerfen wird zum algorithmischen Entwerfen.

1.2 Gegenstand, Zielsetzungen, Arbeitshypothesen

Im Kern dieser Arbeit stehen Experimente mit verschiedenen generativen Verfahren, die entwickelt wurden, um den Entwurf und die Umsetzung konstruktiver Systeme aus virtuell modellierten Flächen zu unterstützen. Die entstandenen digitalen Prozessketten, die von der Ausformung

einer Fläche bis hin zur Ausgabe von Daten für Fertigung und Montage spannen, fußen auf parametrischen und evolutionären Algorithmen, die mit Methoden der Topologieoptimierung sowie der strukturellen Analyse von Stabtragwerken kombiniert wurden.

Neben der Anwendung von Algorithmen als Entwurfsmedium zeichnen sich die Experimente durch drei weitere Besonderheiten aus, die hier entwickelt und untersucht wurden: Erstens sind virtuelle Flächen nicht nur als Begrenzung für die konkrete Form eines Designs zu verstehen, sondern es werden aus ihren topologischen Eigenschaften heraus Entwürfe abgeleitet. Dafür ist die genaue formale Ausprägung einer Fläche zunächst von untergeordneter Bedeutung. Sie übernimmt vielmehr die abstrakte Funktion eines Designträger, eines ‚Transportmediums' einer Designlösung, die in verschiedenen Entwurfskontexten zur Anwendung kommen kann. Zum Zweiten werden Analyse- und Optimierungsverfahren direkt in formerzeugende Prozesse eingebunden und als gestaltende Elemente genutzt: Üblicherweise stellen sich Fragen der Optimierung eher gegen Ende eines Designprozesses, wenn es um die Umsetzung und abschließende Verfeinerung einer bereits weitgehend ausdifferenzierten Lösung geht. Hier wird dieser Aspekt dadurch erweitert, dass Fragen der Optimierung an den Anfang gestellt und daraus gestalterische Interpretationen abgeleitet werden. Die dritte Besonderheit schließlich besteht in der Rolle, die die *Constraints*, die Randbedingungen einer Entwurfsaufgabe spielen *(vgl. Kilian, 2006)* [02]. Sie treten normalerweise eher unliebsam in Erscheinung, als einschränkende und behindernde Elemente bei der Suche nach Designlösungen. Beim algorithmischen Entwerfen übernehmen sie die Funktion von *Design Drivern*, also von Bedingungen, deren Änderungen weitreichende Auswirkungen auf den Entwurf zur Folge haben. Sie werden strategisch genutzt, um Lösungsräume zu beschreiben und einzugrenzen und dienen so als Ausgangspunkte für die Entwicklung generativer Algorithmen.

Soweit ist die Arbeit dem Bereich *„research through art and design" (Frayling, 1993)* [03] zuzuordnen, bei dem Themenstellungen definiert, projektbezogen entwickelt, untersucht und schließlich diskutiert werden. Aus entwurfspraktischer Sicht war das Ziel zudem, die hier entworfenen generativen Werkzeuge und Methoden soweit zu bringen, dass sie auch tatsächlich in einem realen Entwurfskontext Anwendung finden könnten – diese Zielsetzung ist abgeleitet aus der Beobachtung, dass nicht wenige der in experimentellen Kontexten entwickelten generativen Designansätze im virtuellen Möglichkeitsraum verharren und den Nachweis ihrer praktischen Tauglichkeit schuldig bleiben. Insofern fällt die Arbeit zum Teil auch in die Kategorie *„research for art and design" (Frayling, 1993)* [04], bei der die Ergebnisse die Relevanz und den Wert des Forschungsansatzes belegen sollen. Die Entwicklung generativer Werkzeuge und Methoden sowie der Umgang damit, macht recht schnell erfahrbar, dass hier ganz andere Erwartungshaltungen, Entwurfsstrategien und Arten, Design zu denken angebracht sind, als aus traditionellen Zusammenhängen gewohnt. In Folge dessen hat sich der praktische Anspruch im Fortlauf der Arbeit mehr und mehr zu der allgemeineren Fragestellung verdichtet, ob und inwieweit man bei der Anwendung von Computeralgorithmen überhaupt noch von Entwurfsmethoden sprechen kann. Oder ob sie auf Grund ihrer deterministischen Natur nicht vielmehr ‚nur' zur Lösung kniffeliger geometrischer Detailprobleme, zur Automation von Routineprozessen oder als relativ beliebiger Motor zum Variieren eines im Grunde schon abgeschlossenen Entwurfs geeignet sind.

Einige der hier gefundenen Antworten sind ausgedrückt im Titel der Arbeit: *„Konstruktion algorithmischer Entwurfsräume"*, der im wesentlichen auf drei Aspekte hinweist, die hier als Arbeitshypothesen formuliert werden sollen und die gleichsam als Leitfaden für diese Arbeit gedient haben: Algorithmisches Design ist, allgemein gesprochen, eine Entwurfshaltung, bei der generative Verfahren entworfen und angewendet werden, um Designprobleme zu entwickeln und zu lösen. Wenn dabei generative Prozeduren auf der Ebene von Werkzeugen betrachtet werden, gilt für sie, was für alle anderen Entwurfswerkzeuge auch gilt: Sie stehen in ständiger Wechselwirkung mit dem methodischen Kontext, in dem sie angewendet werden, darüber mit dem Denken

und Handeln eines Entwerfenden und bedingen somit auch die Lösung einer Designaufgabe. Sie können dabei Mittel der Erkenntnisgewinnung und praktischen Umsetzung gleichermaßen sein. Wenn aber generative Verfahren entworfen werden, werden dadurch die Mittel selbst zum Gegenstand des Entwerfens und es wird damit eine zweite Arbeitsebene eingeführt – die des Algorithmendesigns. Die Entwicklung von Algorithmen ist aber schon selbst ein aufwändiger, kreativer Prozess, der entsprechende Ressourcen und intellektuelle Aufmerksamkeit bindet, bevor sie überhaupt im eigentlichen Designkontext Anwendung finden können. Dies führt dazu, dass die Entwurfsaufgabe – ob bewußt oder unbewußt – gedanklich so konstruiert wird, dass sie der algorithmischen Bearbeitung zugänglich wird. Werkzeuge und Methoden nehmen also ganz deutlich eine das Entwurfsdenken und -handeln dominierende und damit das Ergebnis konstituierende Rolle ein – hier ist vor allem nach der Art der Interaktion zwischen Designdenken und generativem Entwerfen sowie den Bedingungen, die Letzteres dafür setzt, zu fragen.

Der zweite Aspekt ergibt sich daraus, dass die Begriffe *‚Algorithmus'* und *‚Entwerfen'* auf den ersten Augenschein unterschiedlichen Welten entlehnt zu sein scheinen: Die Entwicklung von Algorithmen hat sich aus praktischer Sicht an Determinismus und Finitheit zu orientieren, während Designprozesse grundsätzlich nicht vorhersehbar und ihre Lösungsräume unendlich sind. Wenn Algorithmen zur Erzeugung von Formen genutzt werden sollen – und nur dieser Bereich wird hier betrachtet – ist es zuerst nötig, dafür ganz klare Randbedingungen und Zielkriterien festzulegen. Damit werden deutliche Weichen bezüglich des Lösungsweges gestellt und gleichzeitig das Spektrum möglich Lösungen drastisch reduziert. Der Entwurf und die Umsetzung einer generativen Logik konstruiert also immer auch einen klar abgrenzten Entwurfsraum. Solche Entwurfsräume sind zwar auf Grund der parametrischen Natur von Algorithmen variabel und können eine Vielzahl unterschiedlicher Lösungen erzeugen, aber nur innerhalb einer sehr schmalen Bandbreite – Variation anstatt Varietät. Entsprechend sind Designmodelle, die auf diesem Weg erzeugt werden, in aller Regel klar als Kinder ein und des selben generativen Prinzips erkennbar [Abb. 04]. Es ist ein Gebot der Ökonomie, dass innerhalb eines Projekts nicht beliebig viele solcher Entwurfräume entwickelt werden können. In diesem Zusammenhang sind die methodischen Konsequenzen, aber auch die entwerferischen Qualitäten und die Relevanz generativer Entwürfe zu betrachten.

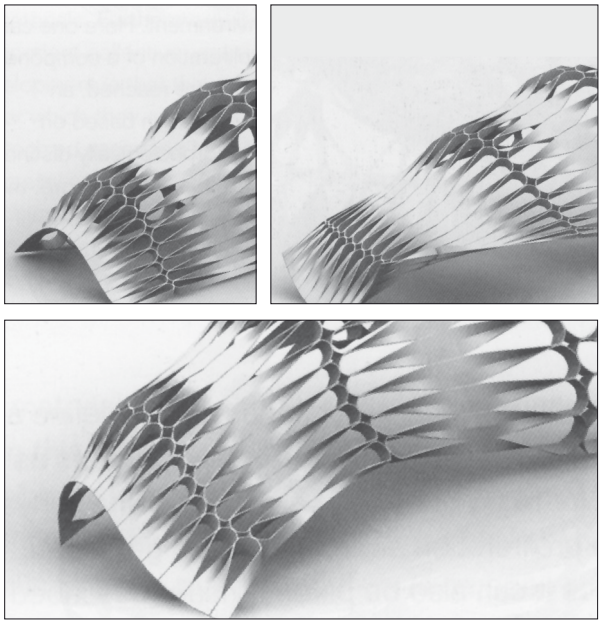


Abb. [04]) Algorithmischer Entwurfsraum – trotz einer fast unendlichen Anzahl an Variationsmöglichkeiten ist jedes einzelne erzeugte Ergebnis eindeutig einer formerzeugenden Logik zuordenbar. („Digitale Morphogenese – paper stripes". Hensel; Menges, 2005)

In dem Maße, wie die Verfügbarkeit von Algorithmen zur Formerzeugung und -transformation wächst, erweitert sich auch das Spektrum an Aufgabenstellungen, die mit generativen Methoden bearbeitet werden können. Verschiedene Programmbausteine eines generativen Prinzips können verändert, erweitert und mit Bausteinen anderer Ansätze kombiniert und vernetzt werden und sind nicht für jedes Projekt von Neuem zu entwickeln. Auf individueller Ebene bedeutet dies zunächst eine Spezialisierung, die Ausbildung eines neuen Entwurfsfelds, das Algorithmen speziell für Designzwecke entwickelt – was bereits Realität ist. Wenn das hier erarbeitete, wertvolle Verfahrenswissen öffentlich gemacht und in Datenbanken zur Verfügung gestellt würde,

könnte der Umgang mit generativen Algorithmen spielerischer werden. Aus der Konstruktion einzelner, individueller algorithmischer Entwurfsräume könnte auf kollektiver Ebene eine neue Entwurfskultur erwachsen: So wie heute Formen am *CAD*-System über die Ausführung einer langen Kette einzelner Befehle erzeugt werden, die Logik der Formerzeugung aber in den Köpfen der Entwerfer verbleibt, würden in Netzwerken verfügbare, generative Algorithmen dieses Wissen nicht nur beinhalten, sondern innerhalb kürzester Zeit auch in jedem beliebigen Entwurfskontext verfügbar und direkt anwendbar machen. Dies ist der dritte Aspekt, der mit *„Konstruktion algorithmischer Entwurfsräume"* gemeint ist, er zielt auf die mittelfristige Entwicklungsperspektive solcher Ansätze und auf die Entwurfsstrategien, die sich darüber entwickeln können.

Die Herausbildung einer neuen, algorithmisch geprägten Entwurfskultur ist etwas, was auf Grundlage einer jahrhundertealten Entwurfstradition stattfindet und diese dabei radikal verändern könnte – aber sie betritt gleichzeitig auch in vielerlei Hinsicht Neuland und ist in den Köpfen noch nicht auf breiter Ebene etabliert. Sie erscheint aber in Anbetracht der jüngeren Entwicklungen bei den digitalen Designtechniken als realistisches und folgerichtiges Szenario, das eine Erweiterung des Design- und Methodenverständnisses, aber auch eine Veränderung des intellektuellen Umgangs mit dem ‚Werkzeug' *Computern* brauchen und bedingen wird.

1.3 Einführung in die Kapitel

Die *Kapitel 2, 3, 4* und *5* dieser Arbeit besprechen einige der allgemeinen theoretischen Grundlagen des Themenkomplexes *‚Algorithmische Entwurfsräume und generative Methoden',* wobei sich *Kapitel 5* auf die Methoden beschränkt, die im praxisbezogenen Teil der Arbeit Verwendung finden.

Kapitel 2 („Design / Methoden / Computer und Algorithmen") versucht anhand verschiedener Aspekte beispielhaft einige Brücken zu bauen zwischen einem allgemeinen Designverständnis und dem, was sich als algorithmisches Entwurfsverständnis heraus zu bilden beginnt.

In *Kapitel 2.1* wird ein grundlegender Widerspruch bei der Anwendung von Algorithmen in Designprozessen thematisiert, der darin besteht, dass das ständige Angleichen von Problemstellung, Lösungsvorschlägen und Zielen ein wesentlicher Bestandteil entwerferischer Arbeit ist, Algorithmen hingegen wohldefinierte Problemstellungen und klare, detaillierte Zielvorgaben brauchen, damit sie überhaupt sinnvoll zur Anwendung kommen können – per Definition wäre damit ein Entwurfsproblem aber auch schon wesentlich gelöst. Hierzu wird vorgeschlagen, dass Algorithmen selbst als zu gestaltende Mittel begriffen werden, als gedankliche Anlässe und praktische Instrumente, mit deren Hilfe sich Entwurfsprobleme entwickeln und lösen lassen.

In *Kapitel 2.2* wird, ausgehend von der Designerpersönlichkeit, ein entwerferisches Bild von Design gezeichnet, das einige seiner Alleinstellungsmerkmale wie schlechte Strukturierbarkeit, Unvorhersehbarkeit oder hohe Komplexität gegenüber anderen Problemtypen in den Vordergrund stellt. Daran anknüpfend wird versucht zu zeigen, wie die algorithmischen Prozeduren genutzt werden könnten, um Entwurfsprozesse zu strukturieren, zu informieren sowie ihre Komplexität zu reduzieren. Danach wird ein weiterer Anknüpfungspunkt zwischen gängiger und algorithmisch inspirierter Designpraxis diskutiert: Strukturen, Muster und Regeln, nach denen Formen erzeugt werden und die sich bei geeigneter Problemstellung auch in Form von Algorithmen darstellen lassen – implizites Wissen über Formerzeugung wird explizit gemacht. Zuletzt wird argumentiert, dass diese Art des Entwerfens einen neuen Designertypus hervor bringen wird, der sich in der abstrakten Welt der Programmierung ebenso wohlfühlen muss, wie in deren Anwendung auf spezifische Aufgabenstellungen.

Der Entwurf und die Anwendung generativer Verfahren hat einige methodische Konsequenzen, die in *Kapitel 2.3* besprochen werden: Hier wird zunächst gezeigt, wie sich

die Idee eines offenen, persönlichkeitsbezogenen, in ständiger Erweiterung und Veränderung begriffenen Clusters an Designmethoden erweitert: Der Entwurf von Designalgorithmen bringt Methodenwissen nach außen, es wird wertvolles Verfahrenswissen in Form einer Sammlung algorithmischer Prozeduren erzeugt und gespeichert – es entsteht eine neue Organisationsform von Designwissen, die auch Einfluss auf den bisherigen, oft künstlerisch geprägten Habitus des Entwerfers nimmt. Dann werden einige Hürden besprochen, die zu nehmen sind, wenn sich solche neuen Entwurfsansätze auf breiterer Basis etablieren sollen: Im Wesentlichen betrifft dies den gedanklichen Umgang mit Computern sowie den Randbedingungen in Designprozessen: Hier wird diskutiert, wie sich die Wahrnehmung von *Computern* von Hilfs- zu Denkmitteln verschieben sollte, wie sie nicht nur auf der konvergenten, sondern auch auf der divergenten Seite des Designdenkens verankert werden sollten und wie die Randbedingungen eines Designproblems dabei eine Rolle als *Design Driver*, als wesentliche, das Entwurfsgeschehen modellierende Elemente einnehmen.

In *Kapitel 2.4* werden einige Aspekte der Computeranwendung besprochen, die im Zusammenhang mit generativen Verfahren eine besondere Rolle spielen: Das Medium *Computer* ist in seiner Allgegenwart ein unverzichtbarer und völlig selbstverständlicher Teil des Entwerfens geworden – es transportiert aber nicht nur Ideen in die reale Welt, es transformiert auch das Designdenken, was im Alltagsgeschehen kaum mehr wahrgenommen wird. Programmierung, als grundlegende Fähigkeit und Fertigkeit des generativen Entwerfens, bietet die Chance, sich den evozierenden Charakters der digitalen Informationsverarbeitung deutlich zu machen und sich so ein Stück weit davon zu emanzipieren. Ein Emanzipationsbestreben ganz anderer Art liegt in der häufigen Verknüpfung generativer Entwurfsansätze mit freien, naturhaft anmutenden Formen. Hier wird zunächst besprochen, wie diese Verbindung zum Symbol einer neuen Entwurfskultur wird, die über ein traditionelles, an der Formensprache der euklidischen Geometrie orientiertes Entwurfsverständnis hinausgehen will. Gleichzeitig werden einige grundlegende praktische Probleme dieser Allianz angesprochen. Das Kapitel schließt mit einer kurzen Stellungnahme zum Thema ‚Rechnerintelligenz', deren unterschiedliche Bewertung eine häufige Quelle der Mißinterpretation generativer Entwurfsansätze ist.

Unter der Überschrift *„Algorithmisches Design" (3. Kapitel)* werden einige grundlegende Bedingungen und Einschränkungen diskutiert, die dem algorithmischen Entwerfen inne wohnen.

In *Kapitel 3.1* wird zunächst die Möglichkeit eines umfassenden Designanspruchs bewertet, der sich daraus ableitet, dass sich aus dem Zusammenspiel von menschlicher Kreativität und Rechnerleistung komplexere Designmodelle entwickeln lassen, als dies bisher möglich war. Darauf folgend werden die Begriffe *‚Generatives Design'* und *‚Algorithmisches Design'* beschrieben, voneinander abgegrenzt und definiert.

Unter dem Titel *„Constraints und Algorithmen" (Kapitel 3.2)* wird zunächst die Struktur im Entwurf generativer Algorithmen betrachtet und eine Unterscheidung zu konventionellen Entwurfsprozessen getroffen, um nochmals die Rolle der *Constraints* als modellierende Elemente eines algorithmischen Entwurfsraums zu betonen. Daran anknüpfend werden verschiedene Typen von *Constraints* diskutiert und beschrieben, welche Arten von *Constraints* überhaupt für eine Bearbeitung in generativen Prozessen in Frage kommen und welche Funktion sie dabei übernehmen. Weiterhin werden die miteinander verknüpften Aspekte der Ökonomie von Programmierung und der Komplexität von Algorithmen besprochen, um einen Eindruck über mögliche Anwendungsfelder zu vermitteln und um darüber die drei Thesen zur Konstruktion algorithmischer Entwurfsräume zu überprüfen. Anschließend werden Ausführbarkeit, Terminierung, Determiniertheit und Determinismus als die grundlegenden Eigenschaften von Algorithmen im Hinblick darauf diskutiert, welche Bedingungen sie für den Entwurf generativer Programme vorgeben. Zum Abschluss diese Kapitels werden die Kommunikationsschwierigkeiten besprochen, die sich ergeben, wenn Entwurfsideen mit den Mitteln for-

maler Sprachen (Programmiersprachen) ausgedrückt werden sollen.

Während im vorigen Kapitel die grundlegenden Bedingungen besprochen wurden, die das algorithmische Design setzt, werden in *Kapitel 4 „Algorithmische Entwurfsräume"* einige der Möglichkeiten skizziert, die aus diesen Bedingungen für das Entwerfen erwachsen.

Kapitel 4.1 beschreibt die besondere Komplexität algorithmischer Entwurfsräume anhand der verschiedenen Modellierebenen, die hier verwendet werden. Daran anknüpfend wird eine Entwurfsstrategie besprochen, die sich aus dem experimentellen Teil der Arbeit entwickelt hat und die geeignet ist, diese Komplexität handhabbarer zu machen: Eine Unterteilung eines Entwurfsprozesses in die Einheiten *‚Programm'* und *‚Stil'*. *‚Programm'* bedeutet in diesem Zusammenhang, generative Verfahren zu nutzen um räumliche Bezüge, funktionelle und andere Zusammenhänge zu organisieren, auf einer Abstraktionsebene, die unterhalb dem Informationsniveau liegt, das für eine bauliche Umsetzung notwendig wäre. Mit *‚Stil'* ist die konkrete, formale Interpretation eines solchen Programms gemeint, das sämtliche Bauteile soweit detailliert erzeugt, dass sich daraus die jeweiligen Fertigungsdaten ableiten lassen. So entsteht eine gedankliche und in den Programmen tatsächlich manifestierte Schnittstelle, die es ermöglicht, einzelne Programmbausteine einfacher in andere Entwurfskontexte und damit Bedeutungszusammenhänge zu überführen. Weiterhin wird die Voraussetzung für eine solche Strategie diskutiert: Es ist eine Art, Design zu denken, die sich aus einer konkreten Entwurfsaufgabe emanzipiert, die Ergänzung des lösungsorientierten durch problemorientiertes Denken.

Der Frage, welches Designdenken im Kontext des algorithmischen Entwerfens angemessen ist bzw. sich daraus entwickeln wird, widmet sich *Kapitel 4.2*. Es werden dafür beispielhaft das *Population Thinking* (Denken in Lösungsräumen), das *Intensive Thinking* (Denken in graduellen Veränderungen) sowie das *Topological Thinking* (Denken in gleichbleibenden Eigenschaften) diskutiert. Es geht hier nicht um Normen des Denkens, sondern um gedankliche Möglichkeitsräume, die es erlauben, gewohntes Entwurfsdenken in Frage zu stellen und etablierte disziplinäre Barrieren aufzuweichen. Dadurch werden Wege eröffnet, algorithmische Entwurfsräume zu mehr als einer Extrapolation bestehender Entwurfsphilosophien zu machen.

Kapitel 4.3 befasst sich unter dem Stichwort *‚Versioning'* etwas genauer mit einer besonders ungewohnten Eigenschaft algorithmischer Entwurfsräume: Dem Erzeugen von Lösungsräumen anstatt von Einzellösungen. Die Idee des *Versioning* kann auf unterschiedlichen Ebenen zum Tragen kommen: Anpassung eines Entwurfs an sich verändernde Kontextbedingungen, Designproduktion im Sinne der *Mass Customization*, die Übertragung und Variation formerzeugender Programme zwischen verschiedenen Entwurfsaufgaben bis hin zu einer Entwurfshaltung, die Design aus der Perspektive eines informationsverarbeitenden Prozesses sieht. Ausgehend von einer Betrachtung dieser Ebenen werden kurz entsprechend unterschiedliche Erwartungshaltungen von Entwerfern bezüglich der gestalterischen Qualität besprochen, die durch die Variationsfähigkeit generativer Programme erreicht werden kann. Eine der besonders erstrebenswerten Qualitäten, die in diesem Zusammenhang häufiger genannt wird, ist die eines ökologischen Designverständnisses, das durch ein Zusammenwirken menschlicher Kreativität und Computerleistung möglich werden soll. Das Kapitel schließt mit der These, dass sich dieser Aspekt vor allem dann herausbilden und im Entwurf manifestieren kann, wenn der entwerferische Blick weg von diskreten Unterteilung hin zu kontinuierlichen Übergängen geht und sich das Augenmerk nicht nur auf Varietät (Differenz), sondern vor allem auf Variation (Selbstähnlichkeit) richtet.

Das *5. Kapitel* der Arbeit befasst sich eingehender mit den drei Themenbereichen, die bei den Entwurfsexperimente der Arbeit im Vordergrund stehen: *„Evolution, Parametrik, Optimierung".* Hier werden für alle drei Bereiche einige allgemeine theoretische Grundlagen skizziert und Beispiele dafür gegeben, wie sich aus ihnen algorithmische

Entwurfsräume entwickeln lassen. Sie werden dabei sowohl als generische, als auch als allgemein anwendbare Werkzeuge und Methoden besprochen und daran anknüpfend versucht, ihre Eignung als generative Entwurfsmedien zu bewerten.

In *Kapitel 5.1 („Evolution als Algorithmus")* werden zunächst verschiedene Ebenen der Verknüpfung des Evolutionsbegriffs mit Design eingeführt und dargestellt, dass Vorstellungen von Vielfalt, wie sie aus biologischen oder kulturellen Evolutionsprozessen entstehen, nicht auf evolutionäre Algorithmen übertragbar sind. Daran anschließend wird die allgemeine Architektur solcher Algorithmen beschrieben und anhand einiger Beispiele mögliche Anwendungsfelder skizziert. Danach werden evolutionäre Algorithmen, die über eine Interaktion mit einem Anwender ästhetische Urteile als Auswahlkriterien möglich machen, besprochen und schließlich die Anwendung evolutionärer Algorithmen als generative Entwurfswerkzeuge bewertet.

„Parametrisches Design" ist die Überschrift von *Kapitel 5.2.* Hier wird zunächst genauer beschrieben, was unter parametrischen Designansätzen verstanden wird und wie sie sich von den üblichen Arten, virtuelle Modelle über *CAD*-Systeme zu erzeugen, unterschieden. Dann werden die zwei Kategorien parametrischen Designs, die in den Entwurfsexperimenten verwendet werden, an einigen Beispiele beschrieben: Die Entwicklung von Designmodellen nur über geometrische Parameter sowie die Verknüpfung geometrischer Parameter mit der Simulation physikalischer Größen. Abschließend werden parametrische Designansätze kurz auf ihrer Eignung als generative Entwurfswerkzeuge diskutiert.

Das Thema von *Kapitel 5.3 („Design aus Optimierung")* ist die spezielle Art, wie Optimierungs- und Analysemethoden aus dem Ingenieurbereich in dieser Arbeit verwendet werden: Als Mittel, um Designprozesse zu informieren und Anlässe für gestalterische Interpretationen zu gewinnen. Zunächst wird allgemein dargestellt, wie sich der Themenkomplex *‚Design und Optimierung'* aus den Sichtweisen von Entwerfern und Ingenieuren unterscheiden kann. Daran anknüpfend wird argumentiert, dass aus Entwerferperspektive Optimierungsverfahren, die in der Regel deduktiven Lösungsstrategien entsprechen, auch in einen induktiven Zusammenhang gestellt werden können: Nicht Design optimieren, sondern Design aus Optimierung ableiten. Danach werden kurz die beiden Optimierungs- bzw. Analyseverfahren beschrieben, die in den Entwurfsexperimenten angewendet werden: Topologieoptimierung und strukturelle Analyse von Stabtragwerken, Letzteres eingebunden als Fitnesstest in einen genetischen Algorithmus. Dann wird noch kurz der wesentlich Aspekte bei der Verwendung dieser Optimierungsmethoden als generative Designwerkzeuge besprochen: Man kann von den Ergebnissen zu Recht behaupten, dass eine Logik von Optimierung in den formgebenden Prozess mit eingeflossen ist, man kann die Ergebnisse aber nicht als statisch optimierte Strukturen ansehen.

Die *Kapitel 6* bis *9* beschreiben den entwurfspraktischen, experimentell angelegten Teil der Arbeit.

Kapitel 6 führt zunächst anhand des Gegensatzpaares *‚Konstruieren von Form'* und *‚Modellieren von Form'* in das Themenfeld ein. Daraus abgeleitet wird die konkrete Problemstellung der Experimente: Die Überführung einer virtuell modellierten Fläche in eine physische, baulich umsetzbare Struktur. Daran anknüpfend werden zwei grundsätzliche Möglichkeiten diskutiert, wie dieser Prozess mit generativen Verfahren durchgeführt werden kann: Die Entwicklung einer Form aus strukturellen Bedingungen oder die nachträgliche Rationalisierung einer Form, um darüber eine detaillierte Struktur abzuleiten. Ausgehend davon wird das Konzept der *Structural Shapes* vorgestellt, anhand einer Reihe von Beispielen illustriert und schließlich eine allgemeine generative Logik von *Structural Shapes* entwickelt. Dann wird die Methodik der Entwurfsexperimente dargestellt: Zuerst wird die Idee, Flächen als Träger von Designlösungen zu verwenden, erläutert, dann die gemeinsame Architektur der algorithmischen Entwurfsräume und der entwerferische Umgang damit beschrieben. Das Kapitel endet mit einem Blick auf die verwendeten Entwicklungsumgebungen und

einer Diskussion der Zielsetzungen, die den Entwurfsexperimenten unterlegt waren.

In *Kapitel 7* werden die Entwurfsräume vorgestellt, die aus der Kombination von Topologieoptimierung und parametrischem Design entwickelt wurden. Zunächst werden die *Constraints* des Experiments, die entworfene Prozesskette und die Methode der Flächenelementierung sowie der Topologieoptimierung erläutert. Dann werden die beiden hier entwickelten Konstruktionsmodule beschrieben und abschließend einige Beispiele aus diesem Entwurfsraum vorgestellt.

Die Entwurfsräume, die aus evolutionärer Flächenstrukturoptimierung und parametrischem Design entwickelt wurden, stellt *Kapitel 8* vor. Auch hier werden zunächst die definierten *Constraints* und die entworfene Prozesskette erläutert. Dann wird das Verfahren der Flächenelementierung, der genetische Algorithmus zur Optimierung von Flächenstrukturen sowie die beiden parametrischen Programmodule zur konstruktiven Interpretation beschrieben. Abschließend werden die Optimierungsergebnisse, die durch den genetischen Algorithmus erzielt wurden, bewertet und einige Beispiele für die Anwendung der Konstruktionsmodule gegeben.

Kapitel 9 schließt den angewandten Teil der Arbeit ab. Hier wird aus praktischer Sicht ein knappes Resümee des Konzepts der algorithmischen Entwurfsräume gezogen und an Hand einiger weiterer Beispiel kurz skizziert, wie sich die in *Kapitel 7* und *8* beschriebenen Entwurfsräume kombinieren und erweitern lassen.

Kapitel 10 schließt die Arbeit mit einer kurzen persönlichen Bewertung und Nachschau.

2. Design, Methoden, Computer und Algorithmen

2.1 Design und Algorithmen

2.1.1 Widersprüche und ein methodisches Dilemma

‚Algorithmus' und ‚Design' sind zwei Begriffe, die man noch nicht ohne weiteres in Einklang bringt, die sogar sehr schön in Widerspruch zueinander zu stellen sind: Algorithmus klingt nach strikter Regelhaftigkeit, im Design hält man es gerne mit der entwerferischen Freiheit. Design ist unvorhersehbar, Algorithmen sind – zumindest im Allgemeinen – deterministisch. Entwerfen zielt auf die Lösung des Einzelfalls, ein Algorithmus ist besser zur Lösung wiederkehrender Probleme geeignet. Das Vorgehen beim Entwerfen ist systemisch, die Arbeitsweise der meisten Algorithmen systematisch. Programme speichern Wissen, viel Designwissen wird immer wieder *ad hoc* am konkreten Projekt erzeugt. Entwerfen ist der sinnlichen Wahrnehmung verpflichtet, Algorithmen werden mit Abstraktion und Rationalismus in Verbindung gebracht. Design entwickelt Ziele, algorithmische Prozeduren brauchen diese für ihre Anwendung. Algorithmen bringen als Konsequenz Automatisierung mit sich, der Entwerfer fürchtet, entbehrlich zu werden.

Solche Widersprüche sind teilweise faktisch begründbar, zum Teil eine Frage der Anschauung. Darüber hinaus stößt man jedoch auf ein grundlegendes methodisches Dilemma, wenn Algorithmen in einem umfassenden Sinn zur Lösung von Designproblemen genutzt werden sollen: Ein wesentlicher Bestandteil der entwerferischen Arbeit besteht darin, dass Problemdefinitionen, Lösungsvorschläge und Ziele ständig aneinander angeglichen und verändert werden: *„Changing the problem in order to find a solution is the most challenging and difficult part of designing." (Jones, 1970)* [01] Dagegen brauchen Algorithmen wohldefinierte Problemstellungen und klare Zielvorgaben, damit sie überhaupt erst entwickelt werden und zur Anwendung kommen können. Wenn aber eine Problemstellung umfassend untersucht und verstanden ist und dabei die zu erreichenden Ziele genau definiert wurden, ist ein Entwurfsproblem auch schon wesentlich gelöst *(vgl. Rittel, 1972)* [02] – was wiederum generative Methoden zu puren Mitteln der Umsetzung, nicht aber der Lösung von Entwurfsaufgaben degradieren würde. Dieses Problem scheint die Einsatzgebiete von Algorithmen in der Designpraxis stark einzuschränken. Entsprechend wichtig sind dort auch die Aspekte der Automation von Routineabläufen, der Bewertung von Designmodellen und der Variation eines bereits weitgehend ausformulierten Designs. Einen Beitrag zur Erzeugung von Entwurfslösungen können Algorithmen auf den ersten Augenschein nur dort liefern, wo genau beschriebene und abgegrenzte Problemstellungen vorliegen.

2.1.2 Algorithmen als Gegenstand des Entwerfens

Eine naheliegende Ebene, sich aus der Außensicht generative Programme zu erschließen, ist sie mit anderen Programmen (etwa *CAD*-Anwendungen) zu vergleichen. Das Arbeiten mit *CAD*-Systemen ist ein *Hands-On*-Prozess, bei dem jeder einzelne Schritt vom Entwerfer entschieden werden muss. Dagegen bringen generative Algorithmen virtuelle Designmodelle, scheinbar wie von selbst, mit einer sehr großen Geschwindigkeit auf den Computerbildschirm. Dies kann zu der irrigen Vorstellung verleiten, dass es das Programm wäre, das entwirft: Auf der Ebene ihrer Ausführung betrachtet sind generative Programme jedoch tatsächlich kaum mehr als Helfer zur Lösung kniffeliger Geometrieprobleme und zur Automation sich wiederholender Arbeitsabläufe. Generative Algorithmen können nur dann eine kreativere Rolle im Designprozess übernehmen, wenn sie selbst als Gegenstand des Entwerfens behandelt werden. Sie sind gedankliche Anlässe und praktische Mittel gleichermaßen, über die ein Entwurfsproblem konstruiert und verstanden wird, mit deren Hilfe Lösungsräume definiert und erforscht, sowie Designmodelle erzeugt und verändert werden können. Wenn der Entwurf von Algorithmen in die Lösung einer Designaufgabe einbezogen wird, sind sie damit ein Mittel, Problemstellung, Lösungen und Ziele einander

anzugleichen und werden in diesem Prozess kontinuierlich selbst mit verändert. Dies ist genauso ein *Hands-On*-Prozess, nur bedient sich der Entwerfer einer weiteren Modellierungsebene: Der Metaebene der Programmiersprachen. Auf dieser Ebene kummulieren Entwurfsentscheidungen, die erst dann als ikonisches Designmodell dargestellt werden können, wenn sie in einen streng logischen Zusammenhang gebracht sind.

Damit ist zwar das beschriebene Dilemma aufgelöst, gleichzeitig verändern sich dadurch natürlich nicht die Eigenschaften von Algorithmen *(vgl. Kap. 3.2.5)*. Ihre Entwicklung hat sich aus praktischer Sicht an Determinismus und Rationalität zu orientieren, man kann in der Sprache von Algorithmen nur das ausdrücken, was sich über mathematische Funktionen und logische Operatoren eindeutig beschreiben läßt. Algorithmisches Entwerfen verlangt (zumindest in Teilen) eine lineare Betrachtung einer Designaufgabe, die klare Definition von Randbedingungen, Zielen und Schnittstellen. Es ist daher nicht unbedingt geeignet für einen chaotischen, wenig organisierten Umgang mit Designproblemen. Auch wer sich dieser Art des Entwerfens mit einer Vorstellung von Varietät nähert, wie sie etwa mit einem Zeichenstift zu erzeugen ist, wird enttäuscht werden. Die Domäne der generativen Algorithmen ist eher die Variation, so zum Beispiel bei der Entwicklung und Abbildung selbstähnlicher Strukturen.

Weiterhin ist zu sehen, dass sich in den komplexen Netzwerken, die Entwurfsprobleme sind, eine Vielzahl verschiedener Parametern finden, die auf vielfältige Weise miteinander verwoben sind. Algorithmisch gestützte Erzeugung von Form arbeitet mit einem eher kleinen Ausschnitt aus diesem Parameterraum. Dies zeigt schon, dass sich generative Entwurfsansätze weder für alle Erwartungshaltungen, Entwurfsgegenstände noch Entwerfertypen gleichermaßen eignen – wie dies auch für alle anderen Designwerkzeuge und -methoden gilt. Das Entwerfen mit Hilfe von Algorithmen ist noch nicht selbstverständlich, es muss sich mit seinen Vor- und Nachteilen erst in ein allgemeines Design- und Methodenverständnis einfügen und im praktischen Kontext genauer untersucht werden. Was *Evans (1995)* [03] für 3D-Darstellungen in der Architektur allgemein beschrieben hat, gilt auch für generative Designprozesse: Erst wenn die Logik der Instrumente und Methoden, die eingesetzt werden, verstanden ist, ihre Limitierungen vollständig begriffen sind und adäquat damit umgegangen wird, können sie zu ihrem vollen Ausdruck geführt werden. Dies ist ein Prozess, der beim einzelnen Entwerfer beginnt.

2.2 Design: Verzwickt, unvorhersehbar, schlecht strukturierbar?

2.2.1 Entwerfer, Entwurfsprozesse und -probleme

Entwerfen ist eine anspruchsvolle Tätigkeit, was sich schon an den umfangreichen Anforderungen zeigt, die an professionelle Gestalter gestellt werden. Produktdesigner oder Architekten etwa müssen über ein wahres Sammelsurium an Werkzeugen, Methoden, Fertigkeiten und Fähigkeiten verfügen, um auf den *brief* mit einer adäquaten gestalterische Umsetzung antworten zu können: Im Zentrum ihrer Arbeit stehen materielle Artefakte, deren formalästhetische, symbolische und sonstigen produktsprachlichen Funktionen zu entwerfen sind. Ein Architekt hat die programmatischen, funktionalen, strukturellen und kontextbezogenen Anforderungen an ein Gebäude zu überlegen, genauso wie man von einem Produktdesigner erwartet, dass er einen adäquaten Umgang mit Materialien pflegt, ergonomisch durchdachte Lösungen liefert und im besten Fall innovative Produkte für eine bestimmte Zielgruppe hervorbringt. Beide sollten sie in der Lage sein, ihre Wertvorstellungen mit denen ihrer Auftraggeber und denen der künftigen Nutzer ihrer Produkte in Einklang zu bringen. Sie müssen die ökonomischen und ökologische Konsequenzen ihrer Arbeit im Blick halten und etwaige soziale oder psychologische Implikationen beachten. Sie sollten die Folgen ihres Tuns abschätzen können und müssen in der Lage sein, Designprozesse zu strukturieren, Abläufe zu organisieren, ihre Entwürfe zu kommunizieren und zu argumentieren. Es ist sicher sehr von Vorteil, wenn sie gute Konstrukteure, Kaufleute, Moderatoren und letztlich auch fähig sind, ihre

Tätigkeit in einen größeren Zusammenhang zu stellen und philosophisch zu hinterfragen. Entwerfer müssen *„Könner"* in ihrem Fach und *„Kenner"* vieler verschiedener Disziplinen sein *(vgl. Maser, 1972)* [04].

Die Komplexität von Design entsteht vor Allem dadurch, dass das Verhandeln und Zusammenbringen vielfältiger Kriterien in einer künftigen Lösung nicht in einer klar strukturierten Weise geschehen kann: Die eben genannten Anforderungen sind weder vollständig, noch haben sie alle objektive Gültigkeit oder sind in jedem Fall gleich gewichtet. Sie kommen situativ und kontextabhängig zum Tragen, oft genug widersprechen sie einander und werden erst im Laufe eines Entwurfsprozesses als relevant erkannt. Die Entwicklung einer Form als Antwort auf eine Aufgabenstellung ist keine Tätigkeit, die in diskreten Schritten abläuft, sondern geschieht in gleichzeitigen, miteinander verflochtenen Abläufen. Design beginnt irgendwo, mit wenig Wissen darüber was möglich ist und was getan werden sollte. Im Entwurfsprozess wird ständig Wissen erzeugt, das vernetzt und verwebt und am Entwurfsmodell zusammengeführt wird. Es entsteht ein unordentlicher, zuweilen chaotisch anmutender Prozess, der gekennzeichnet ist von vielfältigen Wechselwirkungen zwischen den einzelnen Entwurfsentscheidungen und bei dem Problembeschreibung, Zielvorstellungen und Lösungen ständig detailliert, aneinander angeglichen oder neu formuliert werden müssen. Designprozesse sind komplex, dynamisch und *„ill-structured" (Simon, 1973)* [05], d.h. sich einer klaren Strukturierung entziehend. Die Qualität ihrer Ergebnisse ist von einer Vielzahl disparater Einflüsse abhängig, die zu Beginn der Entwurfsarbeit oft nicht alle be- oder erkannt sind oder sein können.

Rittel und Webber (1972) [06] verwenden zur Charakterisierung von Entwurfsproblemen den treffenden Begriff ***‚wicked'***, *„bösartig in der Bedeutung, die den Begriffen ‚boshaft' […], vertrackt' […], mutwillig' […] oder ‚aggressiv' […], entspricht"* und arbeiten einige Eigenschaften heraus, um solche bösartigen Problemstellungen von zahmen Problemen abzugrenzen. Eine dieser Eigenschaften bezieht sich beispielsweise auf die wesentliche Einzigartigkeit von Entwurfsproblemen. Damit ist gemeint, *„dass trotz langer Listen von Ähnlichkeiten zwischen einem aktuellen und einem vorangegangen Problem immer eine zusätzliche unterschiedliche Eigenschaft von überragender Wichtigkeit existieren kann." (Rittel et al, 1972)* [07] Diese Eigenart macht Designprozesse unvorhersehbar und dem entsprechend sind auch Patentrezepte, die angeben, wie ein Entwurf sicher zu einem guten Ergebnis geführt werden kann, mit Vorsicht zu genießen: Eine bestimmte Problemlösungsstrategie bedingt auch einen bestimmten Blick auf die Aufgabenstellung und wer sich zu früh auf eine bestimmte Lösungsart festlegt ist in Gefahr, die Besonderheiten des Entwurfsproblems nicht zu erkennen. Deshalb gehört zur Kunst des Entwerfens auch, *„nicht zu früh zu wissen, welcher Art Lösungstyp anzuwenden ist" (Rittel et al, 1970)* [08].

Eine weitere Eigenschaften von Entwurfsproblemen besteht darin, dass sie kein logisches Ende haben *(vgl. Rittel et al, 1972)* [09]. Es bleibt immer etwas, was noch zu prüfen, zu hinterfragen und zu verbessern wäre: Der Lösungsraum von Designproblemen ist unendlich. Der entwerferische Umgang damit besteht hauptsächlich darin, Lösungsalternativen zu erzeugen, diese in kritischem Diskurs mit sich selbst oder mit anderen zu untersuchen und zu bewerten, unerwünschte Lösungen auszuschließen und sich so allmählich einem möglichst guten Ergebnis anzunähern *(vgl. Mitchell, 1990)* [10].

Ärgerlicherweise ist dabei nicht immer klar, warum eine Lösung einer anderen vorzuziehen ist. Eine zunächst als gut befundene Alternative kann sich später doch als schlecht oder nicht umsetzbar erweisen und was man anfänglich verworfen hat, mag später, in einem veränderten Kontext unentdeckte Qualitäten zeigen. Wo es keine klaren und verläßlichen Bewertungskriterien gibt oder nicht gewußt wird, wie sie zu messen wären, bleibt nur schätzen, raten, intuitiv entschieden. Aber selbst da, wo es klar strukturierte und eindeutig messbare Kriterien gibt und es naheliegend scheint, diese anzuwenden, bleibt zu entscheiden, ob sie in einem gegebenen Fall wirklich sinnvoll sind und wie mit dem Ergebnis der Bewertung umgegangen werden soll. Der

Weg, der letztlich zu einer Entwurfslösung führt, ist nicht über eine dem Problem inherente Logik vorgezeichnet, sondern geht nur über die entscheidende Stellungnahme des Entwerfers.

2.2.2 Algorithmen als strukturierende Elemente im Design

Bis hierher ist Design dargestellt als ein komplexer, sehr vielschichtiger, stark durch Persönlichkeiten geprägter, verzwickter und unwägbarer Prozess. Nun ist es sicherlich von unbestreitbarem Charme, sich mit solchen, durch hohen Schwierigkeitsgrad geadelten Problemen auseinanderzusetzen. Dennoch sind die genannten Punkte Alleinstellungsmerkmale und nicht die alleinigen Eigenschaften von Entwurfsprozessen und -problemen. Sie fallen auch nicht bei jeder Designaufgabe gleichermaßen ins Gewicht oder werden von jedem Designer in gleicher Weise wahrgenommen und bewertet. Zudem kann die ganze Unsicherheit und Zufälligkeit, die mit dem Entwerfen verbunden ist, das Gefühl, dass sich ein Ergebnis eher ergibt als dass es planvoll entwickelt worden wäre, das ständige Damoklesschwert des Scheiterns, überaus frustrierend sein. Deshalb ist auch das Bestreben, Komplexität zu reduzieren, klare Ebenen des Entwerfens herauszuarbeiten, intersubjektive Kriterien zur Bewertung von Ergebnissen zu finden, Problemstellungen zu vereinfachen und das Entwurfsgeschehen in abschätzbare Bahnen zu lenken, notwendiger Bestandteil jeder Entwurfsarbeit.

Aus dieser Perspektive können algorithmische Designstrategien in einem anderen Licht erscheinen. Ihre augenfälligen Gegensätze zu einigen Eigenschaften des Designs müssen nicht zwangsläufig eine Antithese bilden, sondern könnten sich gerade als ein geeignetes Mittel erweisen, um Entwurfsprobleme zu strukturieren und ihnen darüber etwas von ihrer ‚Bösartigkeit' zu nehmen. Dies soll hier beispielhaft an einigen Punkten, die ihm vorigen Kapitel angesprochen wurden, illustriert werden.

Zwar geht der Weg zu einer Entwurfslösung nur über die Stellungnahme des Entwerfers, dabei ist der gesamte Prozess der Alternativenbildung und Bewertung aber auch von einer gewissen Beliebigkeit und Zufälligkeit durchzogen. Manchmal trifft der Entwerfer bei seinen Entscheidungen mit traumwandlerisch anmutender Sicherheit ins Schwarze, oft genug liegt er aber auch völlig daneben oder findet nach so und so vielen Durchläufen, dass die erste Idee dann doch die Beste war. Um Sicherheit in der Entscheidungsfindung zu bekommen, benötigt der Entwerfer einen ihm und anderen verständlichen Rahmen, innerhalb welchem Entscheidungen nachvollziehbar getroffen und argumentiert werden können. Die Anwendung generativer Methoden braucht klare Randbedingungen und Zielvorgaben und kann so – bei geeigneten Problemstellungen – einen Beitrag zur Herausbildung einer Entscheidungsgrundlage bilden, als roter Faden dienen, an dem sich die Designarbeit ausrichtet. Dies steht nicht im Widerspruch zum Gefühl eines Entwerfers für ein gutes Design, sondern führt lediglich klar strukturierte Ebenen der Entscheidungsfindung ein, die jederzeit wieder verlassen werden können.

Bezüglich der Einzigartigkeit von Entwurfsproblemen ist zu fragen, ob man sich nur auf die Besonderheiten einer Problemstellung kaprizieren sollte, wenn es auch eine Reihe von Gemeinsamkeiten gibt, auf die man von Projekt zu Projekt mit den selben oder zumindest ähnlichen Problemlösungsstrategien antworten kann – was in der Entwurfspraxis schon allein aus ökonomischen Erwägungen heraus geschieht und dann als die Herausbildung eines typischen Entwurfsstils wahrgenommen wird. Gleichzeitig ist dies auch ein idealer Fall für die Anwendung algorithmischer Entwurfsstrategien, da hier ja einmal erarbeitetes Wissen gespeichert ist und jederzeit – lapidar und vereinfacht gesagt auf Knopfdruck – wieder verwertet werden kann. Zwar birgt das wiederholte Abrufen bewährter Lösungsmethoden das Risiko, einen verkürzten Blick auf das Entwurfsproblem zu haben und deswegen seine Besonderheiten zu übersehen, weil es eben entsprechend den Anforderungen einer Methode ‚passend' gemacht wird. Gleichzeitig entsteht aber auch eine klare Entwurfshaltung, die Raum schaffen kann, um sich den Besonderheiten einer Themenstellung zu widmen.

Was die Komplexität sowie die schlechte Strukturierbarkeit anbelangt, finden sich zwei Anknüpfungspunkte. Um generative Verfahren zu implementieren, muss eine Problemstellung durch möglichst wenige, klar definierte Parameter beschrieben werden, was zunächst ein reduktionistisches Vorgehen beinhaltet, also die Verminderung der Komplexität eines Entwurfsproblems durch Rückführung auf einige wenige Eigenschaften. Von der Seite einer Designaufgabe aus betrachtet finden sich in Teilbereichen auch immer einfachere und klar definierbare Strukturen. Algorithmisches Entwerfen bedeutet in diesem Kontext also, sowohl zu versuchen, eine Problemstellung zu vereinfachen und der algorithmischen Bearbeitung zugänglich zu machen und gleichzeitig solche Ansatzpunkte zu identifizieren, die sich von vorneherein für einen generativen Entwurfsansatz eignen. Man sollte hier allerdings nicht den Fehler begehen, algorithmisches Entwerfen grundsätzlich mit reduktionistischen Lösungsansätzen oder *Straight-Forward*-Methoden gleichzusetzen. Es ist vielmehr eine Strategie, die sich zunächst auf einige Eigenschaften eines Entwurfsthemas konzentriert, um darüber den weiteren Entwurfsprozess zu informieren, also Erkenntnisse für das weitere Vorgehen abzuleiten. Design beginnt irgendwo, dies kann die Ausarbeitung eines konstruktiven Details, eines Raumplans aber eben auch die Entwicklung eines formerzeugenden Regelsystems sein.

2.2.3 Algorithmische Strukturen im Design

Neben solchen Möglichkeiten, die Implementierung generativer Designstrategien als ein entwurfsstrukturierendes, komplexitätsreduzierendes und den Designprozess informierendes Mittel zu sehen, ist das Finden und Erfinden algorithmischer Strukturen natürlich ohnehin in sehr vielen Entwurfsprozessen enthalten – zumindest implizit. Die Suche nach Regeln, Schemata, Verhältnissen und Organisationsprinzipien, nach denen sich Formen entwickeln und zueinander in Beziehung setzen, ist seit jeher ein ganz wesentlicher Bestandteil der entwerferischen Arbeit. Beispiele dafür ist etwa der strikte Proportionskanon, nach dem die Tempelbauten der klassischen griechischen Antike entworfen sind, *Marcus Vitruvius Pollio's (~30 v.Chr)* [11] *„De Architecture libri decem"*, das eine ganze Sammlung von ‚Rezepten' zu architektonischen und stadtplanerischen Fragestellungen enthält wie etwa die optimale Gestaltung einer Befestigungsmauer oder die Herleitung der drei Säulenordnung, *daVincis* Proportionsschema des menschlichen Körpers oder *LeCorbusiers* Modulor, die ihre moderne Entsprechung in der Anthropometrie finden, *Fullers* geodätische Kuppeln, Seifenblasen, die sich mathematisch als Minimalflächen beschreiben lassen und aus denen etwa Gebäudeformen, Tragwerke oder Oberflächenstrukturen abgeleitet werden können, Analogien zu den formerzeugenden Prozessen der Natur und viel dergleichen mehr. Hier bietet sich eine immens reiche Spielwiese von gedanklichen Ansatzpunkten, aus denen sich generative Prozeduren entwickeln lassen.

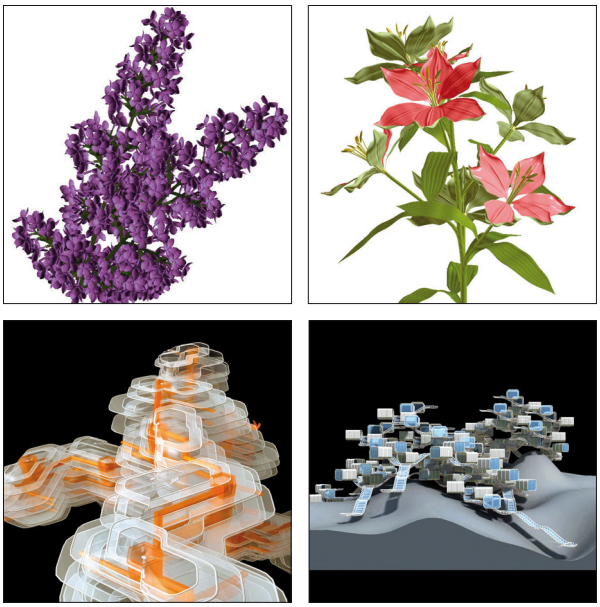


Abb. [01]) Mit L-Studio® (Prusinkiewicz, 1999-2004) erzeugte Pflanzenmodelle und das selbe Erzeugungsprinzip, experimentiert im architektonischen Kontext (Hansmayer, 2007).

Dabei ist es durchaus üblich, dass sich die entwerferischen von anderen Disziplinen inspirieren lassen – bekanntes Beispiel hierfür sind die Lindenmayer-Systeme, eine vom theoretischen Biologen *Aristid Lindenmayer 1968* [12] vorgestellte formale Grammatik, über die sich das Wachstum

von Pflanzen beschreiben läßt und die im architektonischen Kontext etwa zur Erzeugung von Raumstrukturen experimentiert wird [Abb. 01]. Die Übertragung eines generativen Prinzips aus der Natur in den Bereich des Entwerfens bedeutet nun nicht, dass sich das Design grundsätzlich an natürlichen Vorbildern orientieren sollte – obwohl es hier einen überbordenden Reichtum an Formenvielfalt, intelligenten strukturellen Lösungen, komplexen Raumorganisationen etc. zu begreifen und entdecken gibt. Genauso, wie die L-Systeme aus der Naturbeobachtung hervorgegangen sind, aber ein eigenständiges gedankliches System ausbilden, entstehen aus der Übertragung solcher Algorithmen in den Entwurfsbereich wieder neue Gedankengebäude. Wohl ist es wünschenswert, dass sich diese neuen Vorstellungen auch in Entwürfen manifestieren, aber dies ist nur ein Aspekt bei der Entwicklung algorithmischer Entwurfsmethoden. Es geht darüber hinaus auch um eine erkenntnistheoretische Perspektive, die andere Sichtweisen auf den Prozess der Formerzeugung entwickelt, um darüber – das ist immer die Hoffnung – zu besseren Designlösungen zu kommen.

Es ist wünschenswert, dass die entwerferischen Disziplinen hier in weitaus stärkerem Maße als bisher den Mut finden, über das lange etablierte Prinzip der intuitiven Formfindung und der persönlichen Entwurfsstrategie hinauszugehen, um ihre eigenen algorithmischen Beschreibungen zu entwerfen – schließlich gelten sie seit Langem als Spezialisten für Formfragen schlechthin. In gewisser Weise ist es eine immense Verschwendung, dass der kulturelle Reichtum, der hier erzeugt wird, in den Köpfen verborgen bleibt und oft nur mittelbar, am Ergebnis eines Entwurfs, nachvollzogen werden kann. Würde das so konservierte Wissen in Form algorithmischer Beschreibung oder gar direkt weiter verwendbarem Programmcode explizit gemacht, würde der Entwerfer seinen Entwurfstil, seine entwerferische Haltung öffentlich machen. Dies wäre dann gleichbedeutend mit einer Entwurfsphilosophie, bei der die Urheberschaft, die entwerferische Leistung des Einzelnen nicht mehr ausschließlich am konkreten, gegenständlichen Objekt festgemacht würde, sondern an dem Entwurf generativer Algorithmen.

2.2.4 Algorithmische Entwerfer

Dies führt wieder zurück zur Entwerferpersönlichkeit. Es sind zwar einige Ansatzpunkte skizziert worden, wie sich algorithmisches Design in ein allgemeines Designverständnis einfügen könnte. Gleichzeitig soll aber nochmals betont werden, dass sich solche speziellen Designansätze nicht für alle Typen von Designaufgaben oder Designer eignen – die Abstraktheit der Modellierungsebene generativer Algorithmen läßt so manchen Entwerfer schon bei einfachen Beispielen wie der Erzeugungsregel für die *Schneeflockenkurve* [Abb. 02] zurückschrecken – was auch nicht weiter verwunderlich ist: Einschlägige universitäre Ausbildungsmöglichkeiten in diesem Bereich sind bislang rar gesät, entsprechend gering ist die Sensibilität für die Besonderheiten und Chancen des algorithmischen Designs.

Abb. [02]) Erzeugungsregel für die Koch'sche Schneeflockenkurve
über ein L-System:

Startwort: F + + F + + F
Ersetzungsregel: P = { (F -> F + F - - F + F)}

Interpretationsregel:
F = Bewegung nach vorne um eine definierte Länge
+ = Drehung nach links, gegen Uhrzeigersinn, um einen definierten Winkel
- = Drehung nach rechts, im Uhrzeigersinn, um einen definierten Winkel

So muss sich ein Entwerfer, der sich in diesem Gebiet betätigt, nicht nur darin wohlfühlen, seine Kreativität auf der gegenständlichen Seite des Designs, sondern darüber hinaus auch auf der abstrakten Ebene des Algorithmendesigns auszuleben. Wenn *Watanabe (2002)* [13] formuliert, dass algorithmisches Entwerfen ein Prozess ist, der vom Entwerfer verlangt, implizite Ideen von Formerzeugung explizit zu machen und in logische Zusammenhänge zu überführen, betont er damit den wesentlichen Aspekt algorithmischen Designs: *„The task is to „bring into the open" the (subconscious) thought processes at work in the brain, and make them available to (verifiable by) anyone."* Wenn er dem aber vorausschickt: *„[…] we must learn to design with our hands tied. To design*

without the hand, using logic alone.", geht er ein Stück zu weit: Dies wird sicher nicht jeder Entwerfer können oder wollen und es wäre ein großer Rückschritt zu versuchen, Design in einen rationalen, deterministischen Rahmen zu pressen, indem Humor und Ironie, Metapher und Analogie, Symbolik und Gefühl kaum mehr Ausdruck fänden. Es geht vielmehr um eine Synthese der bewährten Fähigkeiten und Fertigkeiten von Entwerfern und den neuen Möglichkeiten, die das algorithmische Entwerfen dazu gibt: *„For the first time perhaps, form might be aligned with neither arbitrary creativity nor computational determinism, but with creative computation or computational creativity." (Terzidas, 2003)* [14]

2.3 Methodendenken, Constraints und algorithmische Lösungsräume

2.3.1 Methodenkompendium

Man kann im Design weniger von einem fest umrissenen Verfahrenswissen sprechen, sondern mehr von einem offenen, breit angelegten Methodenkompendium. Entsprechend wird viel Methodenwissen eher in osmotischer denn klar strukturierter Form erworben und weitergegeben. Im Zentrum designspezifischer Methoden stehen Verfahren zur Formerzeugung und -transformation, zusätzlich werden häufig Anleihen bei anderen Disziplinen wie beispielsweise den Ingenieurwissenschaften, der Psychologie oder den Wirtschaftswissenschaften gemacht und deren Verfahren für das Design adaptiert. Man kann Entwerfern – vielleicht zu Recht – eine gewisse Abneigung gegen methodisches Arbeiten oder gar eine *„Theorienphobie" (Jonas, 1994)* [15] nachsagen, die sich ausdrückt durch einer gewissen Laxheit in der Handhabung von Methoden und einem freien Denken, das Regeln und Rezepten ablehnend gegenübersteht *(vgl. Chouchoulas, 2003)* [16], wobei dies sicher nicht auf alle Entwerfer und Designfelder gleichermaßen zutrifft.

Ein etwas beliebig erscheinende Umgang mit Methoden erklärt sich zum Teil darüber, dass Entwerfen weniger eine problemorientierte als eine lösungsorientierte Tätigkeit ist *(vgl. Lawson, 2006)* [17], die den Einzelfall fokussiert und bei der sich eine klare Logik in der Art und Weise der Lösungsfindung häufig erst retrospektiv einstellt bzw. darstellen lässt. Zudem ist viel Wissen im Design nicht kumulativ angelegt: Die Breite der Anwendungsfelder bedingt, dass Methodenwissen immer wieder kontextabhängig und situativ am Projekt entwickelt werden muss, zum Beispiel wenn sich aus einem Problem heraus eine bestimmte Vorgehensweise anbietet oder erforderlich erscheint, für die es im ‚Werkzeugkasten' des Designers noch keine Entsprechung gibt – das Entwerfen von Methoden ist zu einem gewissen Grad immer auch Bestandteil eines Entwurfsprozesses.

Hieraus wird ebenfalls ersichtlich, warum viel Verfahrenswissen im Design eher implizit als explizit vorhanden ist: Wo jede Aufgabenstellung eine neue Vorgehensweise fordern kann und die Lösung maßgeschneidert wird, ist nicht sicher, ob es den Aufwand lohnt, eine eigens entwickelte Methode genauer zu untersuchen und zu beschreiben. Auch eine mehr oder minder stark ausgeprägte Skepsis gegenüber genau definierten Verfahrensweisen wird verständlich, wenn man bedenkt, dass der Umgang mit dem persönlichen Ausschnitt aus dem Methodenkompendium und die Fähigkeit, neue Vorgehensweisen zu entwickeln bereits selbst wesentliche Bestandteile der Kreativität eines Entwerfers sind. Sie entscheiden maßgeblich mit darüber, welche Aufgabenfelder bearbeitet werden können und durch welche Charakteristika sich Lösungen auszeichnen – die Wahl der Werkzeuge und Methoden lenkt das Denken, definiert den Entwurfsstil und formt darüber auch die Entwerferpersönlichkeit.

2.3.2 Methodendatenbanken

Dieser für das Design (im Allgemeinen) charakteristische Umgang mit Methodenwissen wird sich im Zusammenhang mit generativen Entwurfsprozeduren verändern: Das Design von Algorithmen ist selbst eine aufwändige entwerferische Arbeit, die entsprechende Kenntnisse und Ressourcen verlangt. Auch wenn die Entwicklung generativer

Methoden projektbegleitend stattfindet, wird sich der Entwerferblick dabei schon aus ökonomischen Erwägungen heraus weg von der Einzellösung hin zu Lösungsräumen wenden, was gleichzeitig auch der parametrischen Natur von Algorithmen näher kommt. Das entwerferische Denken richtet sich nicht mehr vordringlich auf die Besonderheiten einer Aufgabenstellung, sondern es werden allgemeine Rahmenbedingungen gesucht, wie z.B. die Verwendung bestimmter Materialien, Konstruktionsprinzipien oder räumlicher Organisationsformen, aus deren Bedingungen heraus Regelsysteme zum Erzeugen und Verändern von Designmodellen abgeleitet werden. Anstatt eine externe Form zu modellieren wird eine interne, generative Logik entwickelt, mit deren Hilfe automatisch eine Spanne an Möglichkeiten erzeugt wird, aus der der Entwerfer dann eine für die weitere Bearbeitung auswählt *(vgl. Kolarevic, 2003)* [18]. Der logische weitere Schritt ist, dass solche allgemein gehaltenen, am Projekt entwickelten Lösungsstrategien, dann auf unterschiedliche Aufgabenstellungen angewendet werden.

Das so entstehende Methodenwissen manifestiert sich nicht mehr nur implizit am Gegenstand des Design, sondern es wird zusätzlich aus den Köpfen heraus, in Form von Programmcode explizit gemacht und kann jederzeit wieder abgerufen werden – nicht nur vom Entwerfer selbst. Als eine Folge wird der subjektive, künstlerische Habitus des Entwerfens an Bedeutung verlieren. Bei einem Algorithmus zur Formerzeugung geht es weit weniger darum, wer ihn entworfen hat, sondern um die zu Grunde liegende generative Logik, seine Geschwindigkeit und allgemeine Anwendbarkeit. Solcher Art nach außen gebrachtes Verfahrenswissen liegt als Objekt vor und kann in verschiedenen Kontexten von anderen Akteuren verwendet werden. Der Entwerfer, dessen Kreativität sich auf das Design von Algorithmen zur Formerzeugung anstatt direkt auf die Formerzeugung selbst konzentriert, tritt in den Hintergrund und wird anonym. Um hier möglichen Problemen mit dem Selbstverständnis von Entwerfern zu begegnen, macht *Terzidis (2003)* [19] den charmanten Vorschlag, so wie heute vom Satz des Pythagoras oder der Geometrie des Euklid gesprochen wird, künftige Designalgorithmen nach ihren Entwicklern zu bennen.

Solche Algorithmen ‚entwerfen' selbstverständlich genauso wenig wie ein Zeichenstift, wer dies glaubt verwechselt die Methode mit dem Designer. Mit ihrer Hilfe können aber beispielsweise automatisch und mit einer gewissen Entscheidungsautonomie geometrisch komplexe Strukturen aus einer Fläche abgeleitet, aus Einzelteilen eine Form komponiert oder Raumprogramme entwickelt werden. Jedes Designmodell, das so erzeugt wird, braucht genauso Interpretation wie auf üblichem Weg entstandene Modelle, diese weitere Interpretation kann dann wieder über eine algorithmisch gestützte Methode erfolgen. So entsteht allmählich, wie beispielsweise von *biothing* mit ihrer *„genware" (Andrasek et al, 2006)* [20] bereits experimentiert, eine neue Art und Organisationsform von Methodenwissen im Design, als Sammlungen algorithmischer Verfahren, die ständig gepflegt, erweitert und untereinander kompatibel gehalten werden.

2.3.3 Computer als Hilfs- und als Denkmittel

Aus methodischer Sicht liegt eine Hürde für die Akzeptanz und weitere Etablierung solcher Entwurfsansätze im aktuell üblichen Umgang mit *Computern* in der Designpraxis begründet. Er wird als ‚Hilfsmittel' behandelt, als Substitut, zur Ergänzung oder Beschleunigung vormals nur händisch ausgeführter Arbeiten. Dabei orientiert sich der Umgang mit diesem ‚Werkzeug' an tradierten Metaphern wie Schreibtisch oder Zeichenmaschine: Ein Strich beispielsweise, der vorher mit Bleistift und Lineal gezeichnet wurde, wird jetzt genauso von Hand, aber über den Umweg ‚Maus oder Tastatur' mit Hilfe des entsprechenden *CAD*-Befehls erzeugt. Diese Übertragung gewohnter Denkstrukturen auf neue Technologien hat *Negroponte (1995)* [21] anhand des Begriffspaars *„bits and atoms"* für die digitale Informationsverarbeitung allgemein veranschaulicht. Hier wird viel von der Logik, die auf einem materiellen Konzept von Information beruht, auf die Entwicklung und die Handhabung dieses neuen Mediums übertragen. *Dritsas (2002)* [22] argumentiert, dass sich der Umgang mit *Computern*, genauso wie der mit neuen Technologien insgesamt, entlang der Phasen *„accessability, complexity, integration"* bewegt. Das heißt, zunächst

muss ein Zugang zu den Möglichkeiten einer neuen Technologie erschlossen werden, dann muss die ihr eigene Komplexität verstanden und durchdrungen werden, was letztlich erst die Voraussetzung schafft, sie in gewohnte Arbeitsabläufe zu integrieren und diese dadurch allmählich zu verändern.

Insofern ist es folgerichtig, dass sich Rechnernutzung im Entwerfen zunächst ebenfalls an gewohnten Mustern orientiert und sich der Einsatz generativer Entwurfsmethoden, die ein anderes Denken der Computertechnologie verlangen, nur langsam durchsetzen kann. Der wesentliche gedankliche Schritt, der hier getan werden muss, ist den Rechner nicht mehr nur als ‚Werkzeug', als Hilfsmittel zur effektiveren Ausführung von Routinetätigkeiten zu sehen sondern ihn darüber hinaus als ‚Denkmittel' zu begreifen. Es ist zu fragen, durch welche inherenten Eigenschaften sich Computer auszeichnen und in welchen Bereichen sie dadurch dem Menschen überlegen sind – unbestritten ist dies beispielsweise ihre Fähigkeit, innerhalb sehr kurzer Zeit eine immens große Zahl an Rechenoperationen durchzuführen. Dann ist weiterhin zu sehen, in welchen speziellen Anwendungsgebieten diese Fähigkeit von Nutzen sein kann. Ein Beispiel hierfür sind polymorphe Systeme, bei denen eine Form durch die Interaktion der Eigenschaften vieler kleiner Elemente mit äußeren Einflüssen entsteht *(vgl. Menges, 2006)* [23]. Solche formerzeugenden Systeme von Hand zu modellieren, käme einer kaum zu bewältigenden Herkulesaufgabe gleich. Und schließlich ist zu versuchen, diese besonderen Qualitäten in Kombination mit der entwerferischen Kreativität von Menschen zu einer Erweiterung der gestalterischen Möglichkeiten zu führen. Diese Schritte setzen voraus zu akzeptieren, dass das Automatisierungspotential, das Rechner bieten, auch auf der kreativen Seite des Designdenkens einsetzbar sind.

2.3.4 Divergentes und konvergentes Denken

Eine ähnliche Trennlinie findet sich auch entlang zweier kognitiver Stile, die eine wichtige Rolle im Design übernehmen. Im Gegensatz zu konvergentem Denken, das in kleinen Schritten auf ein klar definiertes Ziel zuführt, bevorzugen Entwerfer divergente Denkweisen, die assoziativ geprägt sind und eher in die Breite führen *(vgl. Lawson, 2006)* [24]. Dieses ‚Kreuz- und Querdenken', auch laterales Denken *(deBono, 1971)* [25] genannt, werden viele Entwerfer selbst auch schon als hinderlich empfunden haben – alles wird mit allem verknüpft, alles kann immer irgendwie anders sein, jedes ist immer wieder von Neuem zu hinterfragen. Dennoch spielt es eine Schlüsselrolle im Entwurfsprozess: Es führt zur stochastischen Suche in den Lösungsräumen des Design, die mit systematischer Suche allein nur ungenügend erforscht werden können. Es wäre aber falsch, Entwurfsdenken mit divergentem Denken gleichzusetzen. Hier wird nach dem Wünschenswerten gesucht indem Varietät geschaffen, Möglichkeiten entwickelt und nach Werten gefragt wird. Gleichzeitig muss das Wünschenswerte aber auch auf seine Brauchbarkeit, Entwicklungsfähigkeit und Realisierbarkeit hin untersucht und überprüft werden, die Seite des konvergenten Denkens. In einer fraktalen Wechselbeziehung mit dem suchenden Denken erzeugt es zunehmende Komplexität, aber auch Struktur und Klarheit im Entwurfsprozess. Zwischen diesen beiden Polen, dem Wünschenswerten und dem Machbaren, liegt das Mögliche, das eine ständige Transformation aus beiden Richtungen erfährt *(vgl. Burgos et al, 2006)* [26].

Obwohl beide Denkweisen eng miteinander verwoben sind, werden Rechner ganz selbstverständlich da eingesetzt, wo es weniger kreativ zu geht, etwa zur Analyse eines Designmodells oder zur Visualisierung einer Idee. Auch diese Trennung ist nachvollziehbar: Wenn man von einem einfachen Zeichenstift als Werkzeug ausgeht, so kann der geübte Entwerfer über seine Skizzen eine sehr schnelle Kommunikation mit sich selbst führen, wobei er unmittelbar, mit dem gleichen Mittel, von einer zielgerichteten in eine divergente Denkweise wechseln kann. Schon beim Einsatz eines *CAD*-Systems verlangsamt sich dieses ‚Gespräch' mit sich selber, da hier zunächst eine ganze Sequenz kleinteiliger, aufeinander aufbauender Befehle ausgeführt werden muss, bevor der Entwerfer seine Idee vergegenwärtigt sieht. Beim Design mit Hilfe von Algorithmen erfährt diese

Kommunikation eine weitere zeitliche und – über die Ebene von Programmiersprachen – bildliche Abstraktion, was auch generative Entwurfsansätze zunächst der konvergenten, ziel- und ausführenden Seite des Designdenkens zuzuordnen scheint.

2.3.5 Algorithmische Lösungsräume aus Randbedingungen

An dieser Stelle bekommen die Randbedingungen eine besondere Bedeutung als *Design Driver*, wie es *Kilian (2006)* [27] in seiner Arbeit *„Design Exploration through Bidirectional Modeling of Constraints"* beschrieben hat: *Constraints* sind die Eigenschaften einer Designaufgabe, deren Änderung weitreichende Konsequenzen für eine ganze Reihe weiterer Eigenschaften eines Entwurfs nach sich ziehen. *Design Driver* sind die *Constraints*, die den größten Einfluss auf die Definition einer Problemstellung und ihre Lösung nehmen. Solche Randbedingungen werden üblicherweise als limitierende Faktoren, als unliebsame Hindernisse bei der Suche nach Lösungen empfunden. Sie schränken die entwerferische Freiheit ein (DIN-Normen, Bauvorschriften, etc.) und ihre Änderung kann sogar ein ganzes Entwurfsgebäude zum Einsturz bringen: Entwickelt ein Designer etwa einen Stuhl aus dreidimensional verleimten Schichtholz und es wird dann festgestellt, dass dies in der Herstellung zu teuer wird, ist der Stuhlentwurf damit obsolet geworden. Das rechtzeitige und richtige Identifizieren der Rahmenbedingungen ist eine wichtige Leitschnur auf dem Weg zur Realisierung eines Entwurfs.

Beim algorithmischen Entwerfen erfahren die *Constraints* eine neue Wertung, indem sie bewusst als Elemente zur Modellierung eines Entwurfsproblems genutzt werden. Das heißt, es wird bereits zu Beginn eines Designprozesses zielgerichtet gedacht, indem nach Randbedingungen gesucht wird, über die sich die unscharfen, sich bewegenden Grenzen eines Lösungsraums festzurren lassen. Damit ist eine wichtige Voraussetzung erfüllt, um eine Designaufgabe der algorithmischen Bearbeitung überhaupt zugänglich zu machen. Auf diesen Grenzen baut nun schrittweise der Entwurf generativer Verfahren auf, mit deren Hilfe sich der ‚kleinere', abgegrenzte Lösungsraum erforschen läßt. Algorithmisches Entwerfen beinhaltet also reduktionistisches, die Komplexität eines Entwurfs verringerndes Vorgehen. Es werden klare Zieldefinition vorgenommen und Festlegungen getroffen, um das Spektrum der Lösungsmöglichkeiten einzuschränken. Erst innerhalb dieses eingegrenzten Lösungsraums wird wieder versucht, in die Breite zu gehen, um über die Entwicklung eines generativen Prozesses ein Spektrum an Lösungen zu erzeugen: Es entsteht ein algorithmischer Entwurfsraum.

Ein solches Vorgehen ist etwas, um einen Allgemeinplatz zu verwenden, was eher ein Informatiker oder Ingenieur als Design bezeichen würde, weniger aber ein Architekt oder Produktdesigner. Diese Vermutung geht einher mit einer Unterscheidung die *Bartlett (1958)* getroffen hat: *„Thinking in closed systems"* ***versus*** *„adventurous thinking" (vgl. Lawson, 2006)* [28]. Dabei sagt er auch, dass klar strukturierte und begrenzte Lösungsräume nicht zwangsläufig bedeuten, dass hier weniger kreativ, weniger ‚abenteuerlich' gedacht wird. Dies erscheint nur dann so, wenn ‚Lösung' ausschließlich mit ‚Produkt' gleichgesetzt wird, wenn algorithmische Entwurfsräume mit einer quantitativen Idee von Varietät betrachtet werden.

Solche Lösungsräume beinhalten aber neben den geometrischen Modellen, die sie erzeugen, immer auch eine zweite Lösungsebene: Ein dezidiertes Regelsystem, nach denen sich die Formen dieses Produkts entwickeln. Die klare Definition eines Lösungsraums, das Festlegen von Rahmenbedingungen ist die Voraussetzung dafür, dass sich das Denken darauf fokussieren kann, solche Regelsysteme zu entwerfen, sie in algorithmische Strukturen und letztlich generative Programme zu überführen. Die Menge solcher algorithmisch gefasster Regelsysteme, die Möglichkeit, sie in neue Entwurfszusammenhänge zusammen zu fassen, sie auf Ebene der Algorithmen zu neuen formerzeugenden Regelsystemen zu überführen, ist das ‚Abenteuerliche' im algorithmischen Entwerfen und erzeugt Varietät, nicht die Lösungen, die ein einzelnes generatives Programm hervorbringt.

2.4 Aspekte der Computernutzung im Design

2.4.1 Hilfsmittel, Gegenstand und Medium des Entwerfens

Die Erfindung der *Beziér*-Kurven *(deCasteljau, 1959, Bezièr, 1962)*, die Entwicklung von *Sketchpad (Sutherland, 1964),* dem ersten *CAD*-System, sowie der ersten rechnergesteuerten Fräsmaschine *(Parson, MIT, 1952)* markieren Meilensteine im Bestreben, die Möglichkeiten der Rechnernutzung der Designpraxis dienbar zu machen. Die Idee, Rechner im Design einzusetzen wurde zunächst zwiespältig aufgenommen, so legt *Alexander 1964* [29] mit *„Notes about the Synthesis of Form"* eine Methode vor, Computer systematisch zur Formerzeugung zu nutzen *(vgl. Kaley, 2004)* [30], kam aber auch zu der Einschätzung, dass Computer nicht mehr zum Entwurfsprozess beitragen könnten als eine Armee von Büroangestellten. Zu Beginn der 80er Jahre begann dann die Ära der erschwinglichen Mikro- bzw. Personal Computer, die durch die *Graphical User Interfaces* ein freundlicheres Gesicht bekamen und dadurch intuitiver zu bedienen waren als frühere Rechner. Zur massenhaften Verbreitung von Computern haben sie ihren bekannten Beitrag geleistet, in der Entwurfsarbeit ist die vor über 40 Jahren geäußerte Vermutung Alexanders vielfach widerlegt. Heute findet sich im Bereich der 3D-Formgestaltung ein breites Spektrum unterschiedlicher *CAD*-Software, die Möglichkeiten der *NURBS*-Modellierung haben die formalen Ausdrucksmittel erweitert, physische Modelle können über 3D-Abtastverfahren digitalisiert werden, diverse *CAM*-Anwendungen erlauben es, virtuell entwickelte, auch komplex geformte Entwürfe direkt an rechnergesteuerte Fertigungsprozesse weiterzuleiten. Bei Letzteren steht wieder eine große Palette verschiedener Technologien zur Verfügung, mit denen sich vielfältige Materialien bearbeiten lassen und die einen fließenden Übergang von Modell zu Prototyp zu Produkt ermöglichen und das in einem Fertigungsspektrum, das von der seriellen Großproduktion bis hin zum Einzelstück mit beinahe handwerklichem Charakter reicht.

Obwohl vielfach als reines Hilfsmittel betrachtet, sind Rechner also längst darüber hinaus zu einem unverzichtbaren Medium des Entwerfens geworden, in mehrfacher Bedeutung: Einmal sind sie Träger und Übermittler der Entwurfsideen, die entlang digitaler Prozessketten aus der Virtualität der Computerdarstellungen ihren Weg in die physische Realität finden. Gleichzeitig formen sie auch eine Umgebung, einen technologischen Rahmen, in dem sich Entwurfsideen entwickeln und ausdrücken können und bilden dadurch wiederum Bedingungen des Denkens und Handelns. Dabei ist die Einflussnahme der Rechner auf die entwerferische Arbeit oft subtil und wird kaum mehr bemerkt – sie ist zur akzeptierten Selbstverständlichkeit geworden. So ist es zum Beispiel Usus, bei der Suche nach Materialien, Herstellern, Ausstattungsgegenständen im Netz zu recherchieren. Hierbei ist eine solche Menge an Informationen zu bewältigen, dass man oft nur mit einer *Browsing*- und *Scrolling*-Mentalität weiterkommt, bei der Texte oder Bilder nur noch kurz auf Relevanz *gescreent* werden. Es bildet sich eine oberflächliche Wahrnehmung heraus, die leicht eine Übertragung auf andere Bereiche finden kann.

Ein anderes Beispiel sind *Software*-Anwendungen, bei denen frei Haus immer auch gleich eine Entwurfsphilosophie mitgeliefert wird, eine verallgemeinerte Vorstellung dessen, wie ein Anwender denkt, was seine Bedürfnisse sind und wie er seine Arbeit strukturiert. Dies ist ausgedrückt in dem Vorhandensein bestimmter Befehle sowie ihrer Organisationsform in der Applikation. Ein Tischler z.B. pflegt in der Regel einen Umgang mit Formen, der sich an additiver und subtraktiver Logik orientiert. Damit ist er bei einer *Software* wie *SolidWorks®*, die gut mit Flächenkörpern umgehen kann, bestens aufgehoben. Er wird aber mit seiner Denkweise einige Schwierigkeiten bekommen, wenn er Programme wie *Rhino®* oder *Maya®* benutzt, deren Spezialität das Modellieren von Flächen ist, die Arbeitsweise also einer Art virtuellem Kneten entspricht. Jemand, der in dieser Hinsicht nicht vorgeprägt ist, wird sein Entwurfsdenken an den Randbedingungen orientieren, die die *CAD*-Software bietet, mit der er arbeitet. Rechner transportieren Ideen nicht nur, sie tranformieren auch das Denken, sie sind *„evozierende Maschinen" (Turkle, 1984)* [31] *(vgl. Randow, 1998)* [32].

Computer sind damit Medium der Gestaltung und ein Medium, das Entwurfsdenken gestaltet, aber auch selbst Gegenstand des Design. Ende der 80er Jahre, damals noch unter dem etwas sperrigen Namen ‚Mikroelektronik' ist bereits offensichtlich, dass sich hier ein neues Betätigungsfeld eröffnet sowie eine Reihe neuer Anforderungen an die Entwerfer gestellt werden, die sich durch die zunehmende Miniaturisierung und Immaterialisierung von Produkten und Produktfunktionen ergeben *(vgl. Bürdek, 1991)* [33]. In den Gestaltungsfokus rückten neben dem *Hardwaredesign* nun zunehmend der Entwurf von *Interfaces*, von Schnittstellen zwischen den Funktionen eines Programms und seinen Anwendern. Die Suche nach neuen Anwendungsfeldern für digitale Medien ist zu einer Frage des Service- und Dienstleistungsdesigns geworden, im Bereich der Spieleentwicklung, aber auch in anderen, experimentellen Designfeldern sind die Ergebnisse nur noch für virtuelle Umgebungen gedacht. Bei *Maeda (1999)* [34] schließt sich der Kreis zwischen gestaltendem und zu gestaltendem Medium, in dem er, jenseits etablierter Vorstellungen, als Kern der digitalen entwerferischen Tätigkeit die Programmierung und nicht die Anwendung von Programmen sieht: *„The true skill of a digital designer is the practiced art of computer programming."* Der Entwerfer nutzt das Medium *Computer* um seine eigenen Entwurfswerkzeuge und -methoden zu entwerfen.

In Anbetracht von Entwicklungen, wie sie hier beschrieben wurden, ist die Ausbildung eines digitalen Designverständnisses, das seine Werkzeuge und Methoden selbst zum Gegenstand des Entwerfens macht, nicht nur eine mögliche, sondern fast schon notwendige Konsequenz. Wenn *Maeda (1999)* [35] von den Fähigkeit und Fertigkeiten *(skills)* in der Kunst der Programmierung spricht, meint er solche, die über die pure Handhabung von Werkzeugen weit hinausgehen. Um dies zu verdeutlichen, führt er eine Geschichte von *Katsumi Asaba* an. Dieser erzählt von sich, dass er in seiner Jugend mit einem Bambusstift, den er selbst immer weiter verfeinert hat, in der Lage war, 10 parallele Linien in einem Abstand von einem Millimeter zu zeichnen, wozu natürlich auch ein über viele Jahre trainiertes, sensibles Körpergefühl benötigt wird. In Analogie zu einem solchen Grad an Auseinandersetzung kann der Weg des Entwurfs über die Programmierung von Entwurfswerkzeugen über die Zeit zu einem tieferen Verständnis der digitalen Informationsverarbeitung führen, einen Zugang öffnen zu deren Theorien, Konzepten und Eigenschaften, die ansonsten immer mehr hinter den grafischen *User-Interfaces* verschwinden. Es entsteht die Möglichkeit, sich von den mitgelieferten Zwängen kommerzieller Softwareanwendungen zu emanzipieren, den Grundlagen der digitalen Formerzeugung näherzukommen, dadurch den transformierenden Charakter des Mediums Computers besser zu durchschauen und sich so neue Handlungsoptionen zu erschließen, die jenseits der üblichen Werkzeugmetaphern liegen.

2.4.2 Blobs und freie Formen, Symbole digitaler Entwurfskultur

Die Adaption naturhafter, weich anmutender Formen spielt im Entwerfen immer wieder ein besondere Rolle: Die ausschweifende Ornamentik des Barock, die organischen Formen des Jugendstil, die ersten *iMacs*, deren transluzente Bonbonästhetik auf verschiedenste Produktbereiche übertragen wurde. Eine der neuen Optionen, die digital gestützte, generative Entwurfsmethoden bieten, liegt darin, solche Formen nicht mehr nur von außen zu begreifen und auf den Entwurf zu übertragen, sondern die zu Grunde liegenden formerzeugende Prinzipien nachzuvollziehen, ihre generative Logik zu entschlüsseln, in einem entsprechenden Algorithmus abzubilden und sie so im Entwurfskontext anwenden zu können: Der Entwurf generativer Algorithmen als Methode der Erkenntnisgewinnung und praktischen Umsetzung in einem. Insofern ist es naheliegend, dass digitale Entwurfstechniken im Allgemeinen und generative Designstrategien im Besonderen überwiegend mit Freiformflächen assoziiert werden. In diesem Kontext kommen sie auch tatsächlich sehr häufig zur Anwendung. Dabei ist es nun nicht so, dass sich eine generative Logik nur über Freiformflächen ausdrücken könnte. Anderseits wären aber viele Entwürfe, die mit freien Formen arbeiten, ohne den Einsatz von Rechnern, von automatisierten Abläufen und generativen Prozeduren, gar nicht realisierbar. Die *blobs*, etwas despektierlich

mit ‚Blasen' übersetzt, ein Begriff, dessen Übertragung auf die Architektur auf *Greg Lynn* zurückgeführt wird, stehen dabei zusammen mit nicht-euklidischen Geometrien als Sinnbild für ein neues Entwerfen. Sie sind gleichzeitig Antithese und Erweiterung einer viele Jahrhunderte alten Entwurfskultur, die ihrerseits symbolisiert wird durch die euklidische Geometrie.

Damit eine Form als ‚frei' wahrgenommen werden kann, braucht es auch Zwänge, aus denen es sich zu lösen gilt. Das Entwerfen im industriellen Kontext ist geprägt von einer Formensprache, die sich im Gleichklang mit Industrialisierung, Rationalisierung und Taylorismus entwickelt hat. Wohl zu Recht empfinden dies manche Entwerfer als Normierung, als Einschränkung ihres gestalterischen Ausdrucks und gleichzeitig als eine Mechanisierung der gestalteten Welt, in der sich Dinge in einer immensen Komplexität aneinander fügen, wobei aber ökonomische Zwänge kaum mehr die Möglichkeit lassen, zu hinterfragen, welche Bedeutung diese Art der Entwurfslogik für die Menschen hat, die in diese Prozesse eingewoben sind. Es ist ein wenig so, als ob in einem gigantischen Flussdiagramm überwiegend die Strukturelemente, kaum aber die tieferen Beziehungen dieser Elemente betrachtet werden.

Es ist von großem Vorteil, dass eine Drehbank überall auf der Welt den perfekten Kreis erzeugt, ein Sägeschnitt eine Gerade, dass eine Schraube, die in Amerika gedacht, in Asien gefertigt und in Europa eingedreht wird, dort auch passend sitzt. Und es ist natürlich ebenfalls eine große Errungenschaft, dass sich mit jedem *CAD*-System der Welt genau dieser Kreis, genau diese Gerade und die passende Schraube mühelos erzeugen lassen. Es entsteht aber auch ein selbstreferenzierendes System, das es sehr schwer macht, Entwürfe zu denken und zu realisieren, die sich nicht in diese Logik einfügen. *Moneo (2001)* [36] spricht von *„geometries lost to us because of the difficulties of their representation"*. Mit den Freiformen, die sich virtuell mühelos erzeugen und verändern lassen und den *CNC*-Fertigungsverfahren, mit denen sie sich auch physisch umsetzen lassen, bietet sich ein möglicher Weg, aus solchen Zwängen auszubrechen und es eröffnen sich darüber wieder neue Wege, Design zu denken und Werte auszudrücken – Freiformen als Symbole neuer Entwurfshaltungen. Weiche, sanft gekurvte, naturhaft anmutende Formen sind durch einen Prozess der Unterteilung und Aufgliederung in immer kleinere Einheiten verlorengegangen, an dessen Anfang eine Trennung von Entwurf und Herstellung stand und der heute in einer hochgradig spezialisierten Entwurfs- und Fertigungskultur gemündet ist. Es liegt eine gewisse Ironie darin, dass solche Formen und die damit verknüpften Ideen erst über den Umweg eines digital gestützten Designprozesses wieder möglich werden.

Unumstritten ist das Spiel mit den freien Formen sicher nicht, ihre gestalterischen Relevanz in einem zeitgenössischen Kontext wird unterschiedlich bewertet. Für die *blobs* sollte gelten, was *Cache (1995)* [37] für die Architektur der Moderne feststellt: *„[The modernists] knew that they had to avoid two opposite pitfalls: a dissolution into the indefinite and a return to the representation of natural forms."* Entsprechend ist im Umgang mit freien Formen auch eine eigene gestalterische Qualität zu suchen, die jenseits der reinen Nachahmung von Naturformen oder dem Abdriften in formale Beliebigkeit liegt. Die Verknüpfung von freien Formen und generativen Methoden allein kann noch keine Garantie für die Herausbildung einer neuen Entwurfsqualität sein: So wie sich eine Form nur auf sich selbst beziehen kann, kann auch eine Methode nur der Methode wegen angewendet werden. Generative Entwurfsprozesse sind schon allein wegen der Möglichkeit der Automatisierung bemerkenswert, wegen der ungewöhnlichen Formensprache ihrer Ergebnisse faszinierend, in ihrer Komplexität beeindruckend. Sie sind auch sehr aufwendige Instrumente des Entwerfens und binden viel intellektuelle Kapazität. All dies zusammengenommen erleichtert nicht unbedingt eine nüchterne Bewertung ihrer Ergebnisse. Was für den einen der Silberstreif einer neuen Entwurfskultur ist, die emergente Erscheinung beinhaltet, selbstorganisierende Prozesse benutzt, die die ökologische Intelligenz der Natur in einen modernen, technologischen Rahmen transportiert, neue Gestalten und Werte hervorbringt, mag für viele Andere eine Zeitgeist geprägte, technoide Spielerei von formaler Beliebigkeit sein.

2.4.3 Freie Formen und NURBS, praktische Aspekte

In fast allen *CAD*-Anwendungen werden Freiformflächen über *NURBS (Non Uniform Rational B-Splines)* dargestellt. Wesentlich für den praktischen Umgang mit solchen Flächen ist, dass *„(…) deren Aussehen nahezu beliebig geändert werden kann, indem gewisse Parameter und erzeugende geometrische Elemente variiert werden." (Glaeser, 2005)* [38] Das unterscheidet sie von den mathematischen Flächen, die zwar auch weich und gekurvt erscheinen können, bei denen aber meist nur eine oder wenige formverändernde Größen zur Verfügung stehen. Selbstverständlich basieren auch *NURBS* auf einer mathematischen Beschreibung, aber der rekursive Aufbau dieser Funktionen hat zur Folge, dass wesentlich mehr Parameter zur Formmodellierung genutzt werden können *(vgl. Glaeser, 2004)* [39]. Besonderes interessant an den *NURBS* ist, dass sie das gewohnte Formenspektrum wie Polylinie, Ellipse oder Rechteck bereits enthalten, sie stellen lediglich Sonderfälle in den Parametern der mathematischen Beschreibung dar. Mit den *NURBS* steht also nicht nur ein sehr flexibles, sondern gleichzeitig auch sehr präzises Mittel zur Verfügung, mit den sich ein äußerst weites Spektrum an Formen realisieren läßt. Sie sind sowohl für die interaktive Modellierung im *CAD*-System sehr gut geeignet als auch für eine Bearbeitung mit generativen Verfahren.

Abgesehen von der kontrovers diskutierten Frage ihrer gestalterischen Relevanz, gibt es auch einige praktische und ökonomisch begründete Schwierigkeiten im Arbeiten mit Freiformen: Im Produktdesign ist der ökonomische Rahmen für die Realisierung freier Formen durch große Stückzahlen und kleine Bauteile gegeben. In der Architektur, wo die Stückzahl häufig eins und das ‚Bauteil' auch eine komplette Gebaudehülle sein kann, sind freie Formen oft unbezahlbar. Eine mögliche Lösung dieses Problems scheint die *CNC*-Fertigung zu sein, bei der solche Formen nicht mehr in aufwändigen, händischen Abläufen hergestellt werden müssen, sondern in einem weitestgehend automatisierten Prozess gefertigt werden. Aber auch hier sind ökonomische Erwägungen bedeutend, dies betrifft nicht nur Bearbeitungszeiten, sondern ganz allgemein das Wissen um die Prozesse, die erarbeitet werden müssen, damit die Potentiale solcher Technologien in innovativen Entwürfen auch genutzt werden können. Ganz frei von den Zwängen der Rationalisierung sind also auch die freien Formen nicht, ganz im Gegenteil: Schon beim Modellieren im *CAD*-System sollte eine Logik implementiert werden, die die spätere Umsetzung vorausschauend berücksichtigt, was oft nicht geschieht und den Ingenieuren und Fertigungstechnikern viel Arbeit verschafft.

Dies führt zu einem weiteren Punkt, der sich verändernden Wahrnehmung virtueller Flächen. Alles, was an dreidimensionaler Darstellung auf dem Computerbildschirm zu sehen ist, ist aus einzelnen Flächen zusammengesetzt. In dem Maße, wie sich die Visualisierungsleistung von Computern bis hin zur Darstellung fotorealistischer Bilder und Animationen in Echtzeit gesteigert hat, haben solche Darstellungen auch mehr und mehr an ikonischem Charakter gewonnen, der eine direkte Assoziation mit körperlichen Gegenständen nahelegt. Dabei wird oft vergessen, was virtuelle Flächen tatsächlich sind: zweidimensionale Repräsentationen mathematischer Beschreibungen, die als Symbole für reale Objekte und deren Eigenschaften stehen *(vgl. Kilian, 2006)* [40]. Gleichzeitig verführt die Leichtigkeit, mit der Formen erzeugt und beliebig im virtuellen Raum positioniert werden können dazu, die Probleme der konstruktiven Realität auf später zu verschieben, Flächen erstmal irgendwie zusammen zu bringen, sie dabei aber schon als Bilder eines fertigen Entwurfs zu sehen. Die bekannten Schwierigkeiten beginnen dann, wenn versucht wird, die virtuellen Konstrukte umzusetzen: Flächen werden in die dritte Dimension gebracht, wobei räumliche Beziehungen nicht mehr stimmen, es zu Lücken, Überschneidungen bzw. Selbstüberschneidungen kommt, Toleranzen nicht bedacht und unterschiedliche Materialverhalten nicht adäquat repräsentiert werden, usw. Diese Schwierigkeiten sind nicht spezifisch für Freiformflächen, dort aber ganz besonders stark ausgeprägt. Solche Flächen können nicht nur lokal sehr unterschiedliche Krümmungsradien aufweisen, sondern auch die Krümmungsrichtung häufig wechseln, also lokal Eigenschaften aller drei Flächenklassen (elliptisch, hyperbolisch,

eben) aufweisen. Deshalb werden diese Flächen als ‚komplex' bezeichnet und entsprechend finden sich nur sehr wenige, standardisierte Verfahren (wie die Aufteilung in Dreiecksmaschen), die bei jeder *NURBS*-Fläche, unabhängig von ihrer konkreten Form, angewendet werden können.

Die Entwicklung freier Formen über algorithmische Methoden geht grundsätzlich über zwei Wege. Der eine kann als eine vorweggenommene Rationalisierung gesehen werden, bei der Bedingungen von Material, Fertigungsprozessen, Statik, usw. genutzt werden, um darüber generative Regeln zu definieren, die die Erzeugung einer freien Form steuern. Die zweite Möglichkeit ist eine nachgeschaltete Rationalisierung, bei der versucht wird, die geometrische Logik einer von Hand modellierten Flächen mit den Bedingungen der physischen Welt in Einklang zu bringen. Beide Wege erzeugen in aller Regel Bauteile, die sich ähnlich sind, bei denen sich aber trotzdem kein Bauteil wiederholt. Es ist offensichtlich, dass solche Entwurfsstrategien nur mit Hilfe der Rechenleistung von *Computern* möglich sind und dass es zudem eine digitale Prozesskette braucht, mit der die Fertigungsdaten möglichst ohne Umweg zu *CNC*-Fertigungsautomaten weitergeleitet werden können.

Auch aus diesem praktischen Kontext heraus wird deutlich, warum die Arbeit mit freien Formen stark mit dem Medium *Computer* und besonders mit algorithmischen Designansätzen verknüpft sind, dass diese Verbindung aber durchaus ihre Schwierigkeiten beinhaltet. Die Entwurf einer präzisen, generativen Logik, die sich an physischen Gegebenheiten orientiert und gleichzeitig die besonderen Eigenschaften der virtuellen Flächenrepräsentation bedenkt, kann tatsächlich neue Wege, Design zu denken und auszudrücken, eröffnen. Sie kann aber auch in einen digital inspirierten, überbordenden Methodismus münden, der sich nur selbst genügen will. Das Automatisierungspotential generativer Verfahren, die scheinbar spielerische Leichtigkeit mit der selbst komplexe Formen zu erzeugen und zu variieren sind, schafft Strukturen und Konstruktionen von großem ästhetischen Reiz und öffnet die Tür zu einer neuen Entwurfsvielfalt. Damit die virtuell erzeugten Schönheiten aber auf ihrem Weg vom Modell in die materielle Welt nichts von ihrem Reiz verlieren, ist oft großer technologischer Aufwand nötig, der erst einmal beherrscht und auch bezahlt sein will.

2.4.4 Quantitative Leistungsfähigkeit oder Rechnerintelligenz

Generative Programme lassen sich, wie oben beschrieben, zu Rationalisierungszwecken eingesetzen, wenn beispielsweise eine freie Form in kleinere, baubare Elemente zerlegt wird und dabei verschiedene Parametereinstellungen experimentiert werden. Sie kommen aber andererseits auch zum Einsatz, um etwa bei gewissen Problemen der Formerzeugung zwischen sich widersprechenden Bedinungen zu vermitteln und eigenständig Lösungen zu suchen, was dann häufig in Verbindung mit Schlagworten wie ‚Selbsorganisation', ‚Emergenz' oder ‚Intelligenz' diskutiert wird. Für diese Arbeit wird davon ausgegangen, dass die Intelligenz generativer Systeme nicht im *Computer* begründet liegt, sondern in der intelligenten Anwendung der dem Menschen überlegenen Rechenleistung dieser Maschinen. Algorithmisches Design bedeutet nicht, dass der *Computer* entwirft, sondern dass sich das Entwerfen auf die Ebene der Entwicklung von Algorithmen begibt. *Computer* selbst sind nicht kreativ, aber sie können von Menschen in neuen Zusammenhängen eingesetzt werden und so einen kreativen Beitrag zum Entwerfen leisten. Allerdings gibt es auch Bereiche im algorithmischen Design, die stark in Richtung künstlicher Intelligenzforschung gehen und in denen versucht wird, Programme zu entwickeln, die in der Lage sind menschliche Problemlösungskompetenz zumindest zu simulieren. Eine wichtige Rolle hierbei spielen die induktiven Algorithmen, wie neuronale Netzwerke, die etwa lernen, ästhetische Vorlieben von Menschen zu ‚verstehen' und dadurch in der Lage sein sollen, zielgerichtet schöne Entwürfe zu erzeugen. Ein anderes Beispiel wären genetische Algorithmen, die genutzt werden können, um bei gewissen Problemen der Formerzeugung zwischen sich widersprechenden Bedingungen zu vermitteln und eigenständig Lösungen zu suchen. Ob solcher Programme nun intelligent sind, intelli-

gent arbeiten oder nur intelligent genutzt werden, ist eine Frage, an der sich die Geister scheiden.

Wollte man einen ‚Erfinder' des Konflikts ‚künstliche Intelligenz gegen menschliche Intelligenz' benennen, wäre *Alan Turing (1950)* [41] mit seinem berühmten *Turing*-Test ein geeigneter Kandidat. Bei diesem Test kommuniziert eine Versuchsperson über Tastatureingabe mit einem Gesprächspartner, der ein Mensch oder eine Maschine sein kann. Gelingt es der Maschine öfter als dem menschlichen Gesprächspartner, die Versuchsperson zu überzeugen, sie sei ein Mensch, ist der *Turing*-Test bestanden, die Maschine kann als intelligent bezeichnet werden. Es gibt bis heute keine *Software*, die diesen Test uneingeschränkt besteht. Es findet sich aber zum Beispiel mit *Julia* eine virtuelln Netzintelligenz, die sich in den 90er Jahren in *Chat Rooms* tummelte und die zumindest für einige Zeit im Gespräch mit verschiedenen anderen, menschlichen *Chat*-Teilnehmern, für einen Menschen gehalten wurde *(vgl. Turkle, 1995)* [42]. Beim *Turing*-Test ist die Intelligenz einer Maschine darüber beschrieben, ob sie in der Lage ist, den Eindruck von Intelligenz hervorzurufen. Intelligenz ist also einerseits ein Kriterium der Leistungsfähigkeit einer Maschine und auf der anderen Seite der Interpretation dieser Leistung durch einen menschlichen Nutzer.

Searle (1982) [43] geht dieses Problem von seiner ontologischen Seite an. Er fragt nicht, was Computer zu leisten im Stande sind, sondern nach dem, was sie sind, ob sie im umgangssprachlichen Sinn des Wortes verstehen, was sie tun. Um diese Frage zu beantworten führt er ein Gedankenexperiment durch, das *„Chinese Room Experiment",* ein *Turing*-Test, der in chinesischer Sprache abgehalten wird und bei der *Searle* die Position der Maschine einnimmt. Er zeigt, dass er diesen Test bestehen kann, ohne dass er ein Wort chinesisch verstehen muss, in dem er auf einen bestimmten *Input* chinesischer Zeichen nach genau ausgeklügelten Regeln eine bestimmte Zeichenfolge zurück gibt – genauso wie dies ein Computerprogramm tun würde. Die Maschine erscheint also nur intelligent, ist es aber nicht, weil sie nichts von dem versteht, was sie tut.

Bei diesem Experiment rückt die Leistungsfähigkeit eines Rechners in den Hintergrund, Intelligenz wird zu einem Phänomen von Wahrnehmung und Interpretation des Menschen. Man könnte auch sagen: Intelligent ist das, was uns intelligent erscheint. Sehr schön illustriert ist dies beispielsweise durch die kybernetischen Vehikel *Valentin Breitenbergs (1972)* [44]: Mit Hilfe von Sensoren und Schaltungen von wachsendem Komplexitätsgrad interagieren kleine, mechanische Wesen. Ihre Bewegungen erzeugen dabei den Eindruck von vielfältigem sozialem Verhalten wie Gruppenbildung, Anbetung, Verfolgung, Ablehnung, Angriff etc., obwohl diese Wesen natürlich keine Vorstellung von sozialem Verhalten haben können. Ein populäreres Beispiel sind die seit Jahren im Schach als Duelle ‚Mensch gegen Maschine' zelebrierten Begegnungen von Großmeistern und Schach-*Computern*. Um die Jahreswende 2006 / 2007 verliert der Schachweltmeister *Wladimir Kramnik* mit 2:4 Partien gegen das Programm *Deep Fritz*. *Kramnik* ist bereits im Vorfeld dieses Duells zu dem Schluß gekommen, dass die Zeit, in der Menschen dieses Schachprogramm schlagen können, zu Ende sei *(vgl. spiegel.de, 2006)* [45]. Hier ist der *Computer* zwar dem Menschen überlegen, aber nicht durch eine besonders kreative oder intelligente Spielweise: Der Rechner gewinnt, weil er in der Lage ist, eine sehr große Anzahl möglicher Spielverläufe in sehr kurzer Zeit vorauszuberechnen und daraus die aktuell günstigste Handlungsalternative abzuleiten. Der Rechner verfügt über eine quantitative Überlegenheit, die uns als eine Qualität von Intelligenz erscheinen mag.

Als grundlegende Handlungsprämisse für das algorithmische Design soll hier bis auf Weiteres gelten, was auch *Maeda (1999)* [46] als ersten Hinweis für Programmiernovizen formuliert: *„The computer will do anything within its abilities, but it will do absolutely nothing unless commanded to do so […] the computer cannot think beyond anything you tell it."*

3. Algorithmisches Design

3.1 Metadesign, generatives und algorithmisches Design

3.1.1 Metadesign - Ebenen

„bang design", so der Titel eines Buches von *Norbert Bolz (2006)* [01], ist ein gut geeigneter Begriff, um die weitreichenden Ansprüche und Erwartungen zu veranschaulichen, die mittlerweile mit dem Entwerfen verknüpft werden. *Bang*, ein Akronym für Bits, Atome, Neuronen und Gene, kann interpretiert werden als die Verbindung von künstlicher und natürlicher Intelligenz, von Evolutionstheorie mit Informationstheorie, von virtueller mit physischer Realität, von Mikro- und Makrokosmos, von Physik und Metaphysik. Es ist ein Begriff der umfassenden Synthese verschiedenster Wissensfelder und Denkrichtungen, für Design zu nutzen und selbst wieder zu entwerfen. Letztlich kann er als ein Sinnbild verstanden werden für die rasanten Wandlung des Menschen vom Natur- zum Kulturwesen, dessen wohl wichtigstes Merkmal eben das Gestalten ist und das sich in letzter Konsequenz selbst als zu entwerfendes Wesen betrachten kann.

Wenn alles entworfen werden kann und wird, steigt zwangsläufig auch die Neugierde und Notwendigkeit, fundiertes methodisches Wissen um die Akteure und Prozesse, die das Entwerfen gestalten, zu erlangen. Die alles entscheidende Entwerferpersönlichkeit ist zwar noch sehr präsent, als Modell aber doch sehr limitiert und eher für überschaubarere Designaufgaben geeignet. Ein modernes Designverständnis sieht die Chancen interdisziplinären Handelns und orientiert sich an der Systemtheorie. Entwurfslösungen konstituieren sich aus dem komplexen, dynamischen Zusammenspiel verschiedener Systeme, wobei die Entwerfenden das System nicht als Objekt betrachten, sondern selbst Elemente sind, die das Verhalten der Systeme durch ihr Agieren beeinflussen und verstehen – im Sinne der Kybernetik 2. Ordnung *(von Förster, 2003)* [02]. Das Modellieren komplexer dynamischer Systeme ist ohne die Hilfe von *Computern* nur noch schwer vorstellbar. Man kennt beispielsweise eine Reihe von *Software*-Anwendungen aus dem Bereich der Großindustrie oder der Architektur, mit deren Hilfe Entwurfswissen verschiedenster Akteure zusammengebracht, kategorisiert, hierarchisiert und verfügbar gehalten wird. Ein solches Programm ist beispielsweise *Digital Project®*, eine Entwicklung von *Gehry Technologies* [03], die Elemente des parametrischen Design mit dem *Building Information Management* kombinieren, ein sich herausbildender Standard zur Zusammenarbeit der verschiedenen Gewerke bei architektonischen Projekten. Diese Art von *Software* hat den Anspruch, für unterschiedlichste Projekte einsetzbar zu sein. Dementsprechend repräsentiet sie eher statische Systeme, bei denen die Interaktionsmöglichkeiten weitgehend durch die Programmentwickler vorgegeben werden. Man kann hier also von einem Metadesign sprechen, das auf die Entwicklung *generischer* Werkzeuge abzielt.

Einen anderen Ansatz zeigt das *kaisersrot* - Projekt [04] eine Kooperation von Architekten, Stadt- und Landschaftsplaner sowie Designinformatikern. Hier wurde ein interaktives System programmiert, das die Komplexität einer Siedlungsplanung modelliert. Es spannt von der Parzellierung eines Geländes, die über die soziale Struktur seiner künftigen Bewohner gesteuert wird, bis hin zur detaillierten architektonischen Planung einzelner Gebäude. Es ist ein *generatives* System in dem Sinne, dass an ihm Wissen generiert und durch es Lösungen erzeugt werden. Es ist aus einer und für eine spezifische Lösung entwickelt worden und unterscheidet sich dadurch von den generischen Systemen.

Ein generatives Entwurfswerkzeug kann nun auch generische Elemente enthalten, die in anderen Projekten wieder zum Einsatz kommen könnten, genauso muss ein generisches Werkzeug Schnittstellen enthalten, die die Entwicklung projektspezifischer Inhalte erlauben. Trotzdem ist es sinnvoll, sie als zwei verschiedene Ebenen des Metadesign zu betrachten, weil zwischen ihnen die Trennlinie zwischen lösungsorientiertem und problemorientiertem Arbeiten verläuft, wobei bei letzterem die gewohnten methodischen Strategien des Designs nur noch zum Teil greifen und deshalb erweitert werden müssen.

3.1.2 Generatives Design – projektspezifisch

Das Finden, Nachahmen, Adaptieren (*Mimesis*) und das Erfinden, das schöpferische, kreative Hervorbringen (*Poiesis*) von Regeln, an denen sich das Entwerfen orientieren kann, ist seit jeher untrennbar mit jeder Designarbeit verbunden: Ob *‚der Neufert'* [05] oder die 10 Gebote, *Feng Shui* oder das deutsche Baurecht, *Vitruvs „De Architectura libri decem" (ca. 30 v.Chr.)* [06] oder *Alexanders „Pattern Language" (1977)* [07], *Lindingers (1983)* [08] Liste von Kriterien für eine gute Industrieform oder die Antworten *Burckhardts (1977)* [09] auf die Frage nach Kriterien für ein gutes Design, sie alle liefern Argumentationen, Hinweise, Vorschläge, Anweisungen oder Normen, nach denen sich Gestaltung richten kann oder soll. Je besser die Regeln wahrnehmbar und nachvollziehbar werden, an denen sich ein Entwerfer orientiert, umso mehr entsteht der Eindruck, das dort der Designarbeit ein genaues Programm zu Grunde liegt. Nach innen gerichtet zeigt es sich als eine klare Methodik, als eine Entwurfshaltung, die nach außen hin am Resultat des Entwurfs sichtbar wird, als deutlich erkennbarer formaler Stil: Wir sprechen dann von einem *Gehry, Meier, Eames* oder *Rams* – alle vier Beispiele für Gestalter, deren Arbeiten sich durch eine charakteristische Formensprache auszeichnen.

Ein Entwurfsstil und eine Entwurfshaltung werden üblicherweise nicht bewusst zum Designgegenstand gemacht, sie entwickeln sich im Laufe der Zeit, als Ausdruck der Reife und Persönlichkeit eines Entwerfers. In dieser Persönlichkeit und der jeweiligen Situation liegt auch die Wertung begründet, die Regeln im Entwerfen erfahren können: Wissen wir nicht mehr weiter, ist uns jede Regel recht, an der wir unsere Entscheidungen ausrichten können und die uns weiter voranbringt. Stoßen wir auf Regeln, die von außen vorgegeben sind und die uns den Eindruck von Einschränkung vermitteln, fühlen wir uns ausgeliefert und in unserer entwerferischen Freiheit geknechtet. Wenn wir Regeln aber selber aufstellen, dann können wir sie als Instrumente begreifen, in denen sich unsere Kreativität ausdrückt und die sie gleichzeitig in gezielte Bahnen lenken. Diese Vorstellung eines Programms, das an präzisen formalen Kriterien ausgerichtetet ist, die gleichzeitig die kreative Kraft des Entwerfers bündeln, findet sich in *Ecos (1962)* [10] Beschreibung des von ihm geprägten Begriffs der *arte programmata* wieder: Die programmierte Kunst entsteht in einem dialektischen Spiel aus Werk und Programm, aus Mathematik und Risiko, aus formal geplanten Prozessen und der Akzeptanz des Zufalls.

Diese Beschreibung der *arte programmata* ist so noch nicht hinreichend für eine Definition von generativem Design. Sie drückt zwar die grundsätzliche Idee aus, dass sich formerzeugende Prozesse als präzise Programme abbilden lassen. Gleichzeitig wäre damit aber aus heutiger Sicht auch nur ein allgemeines Bild von Design gezeichnet: Die Dialektik zwischen Entwurf und Programm, zwischen genau geplantem Ablauf und evolutionärem Prozess, zwischen präziser generativer Logik und dem zufälligen Finden oder Erspüren von Formen, findet sich ohnehin in jedem Entwurfsprozess, wobei das Pendel mal mehr zu der einen oder zu der anderen Seite ausschlägt. Zudem wird die Regelhaftigkeit eines Werkes bei *Eco* retrospektiv bestimmt, beim generativen Design beginnt der Entwurf von Regeln aber vor der Ausführung des eigentlichen Ergebnisses.

Galanters (2003) [11] Definition von generativer Kunst führt hier weiter. Er spricht von prozeduralen Erfindungen wie etwa in natürlicher Sprache formulierte Regeln, einem

---

[10]) „In Bezug auf die vorliegenden Werke kann, im Nachhinein, eine Art von Programm festgestellt werden, was die Möglichkeit beweist, mit der linearen Reinheit eines mathematischen Programmes Vorkommnisse zu programmieren, die natürliche Prozesse nachbilden. So wird ein dialektischer Zusammenhang zwischen Werk und Programm, zwischen Mathematik und Risiko, zwischen formal geplanten Prozessen und der Akzeptanz des Zufalls hergestellt, was angesichts der Tatsache, dass im Grunde alles auf präzisen formalen Kriterien beruht, die Spontaneität nicht negiert, sondern ihr Grenzen und mögliche Richtungen vorgibt. So können wir also von programmierter Kunst sprechen." (Eco, 1962)

Computerprogramm oder einer Maschine, die zur Entstehung eines Kunstwerks beiträgt oder ein solches hervor bringt. Damit macht er deutlich, dass die Verwendung generativer Prozeduren nicht nur selbstverständlich in einen Entwurfsprozess eingeflochten ist und sich quasi als Beigabe ergibt, sondern dass sie sich in einer Einheit zusammen fassen lassen, die zum charakteristischen Element eines Designprozesses wird. Diese Idee einer prozeduralen Methode als kennzeichnendes, eigenständiges Element rückt Galanter durch den Begriff der Autonomie, deren Bedeutungsspektrum von selbsttätig (automatisch) bis selbständig (intelligent) reicht, noch weiter in den Vordergrund. Damit ist die Eigenschaft bezeichnet, nach der generative Programme bis zu einem gewissen Grad ohne Zutun des Entwerfers ausgeführt werden und sie dabei auch innerhalb eines definierten Rahmens Entscheidungen treffen können. Der Aspekt der ‚prozeduralen Erfindung' aus *Galanters* Definition soll auf Computerprogramme beschränkt sein und der Entwurf einer generativen Logik noch etwas mehr in den Vordergrund rücken. Dadurch ergibt sich folgende Definition:

Generatives Design entwirft präzise Regelsysteme für einen gegebenen Designkontext, die als Computerprogramm implementiert und mit einem gewissen Grad an Autonomie ausgeführt werden, um Formen hervorzubringen oder zu transformieren.

Die persönliche Haltung eines Entwerfers findet sich bei einer solchen Definition von Design in zweifacher Weise wieder. Erstens in der prinzipiellen Entscheidung, generative Verfahren anzuwenden und Zweitens in der konkreten, projektbezogenen Umsetzung bestimmter generativer Methoden. Genauso wird ein persönlicher Entwurfsstil auf zwei Ebenen sichtbar: Auf der Ebene des Programmcodes, der selbst eine Ebene ästhetischer Betrachtung ist und der Ebene der Entwürfe, die durch diesen Code erzeugt werden.

3.1.3 Algorithmisches Design – generisch

Nach dieser Definition könnten algorithmisches Design und generatives Design als Synonyme verwendet werden, trotzdem sollen sie im Kontext dieser Arbeit einen unterschiedlichen Beiklang haben. Zunächst ist zu sehen, dass vor der Implementierung eines generativen Computerprogramms die Konzeption einer allgemeinen generativen Logik und die Entwicklung von Algorithmen steht, die diese Logik präzisieren, um sie dann in ausführbare Prozeduren umwandeln zu können. Auch sind Algorithmen im allgemeinen deduktiv, sie adressieren eher allgemeine denn spezielle Problemstellungen – einem Sortieralgorithmus ist es ‚egal', ob er Fitnesswerte einer evolutionären Optimierung, Flächengrößen oder Wegelängen als *Input* erhält. Entsprechend haben generative Techniken und Methoden in aller Regel auch dann eine gewisse Allgemeingültigkeit, wenn sie nur für eine spezielle Problemstellung konzipiert wurden. Genau dieses Herauslösen aus dem pragmatischen Bedeutungsgeflecht eines konkreten Projekts ist aber eine wesentliche Voraussetzung, um Algorithmen projektübergreifend einsetzen zu können, in gedanklicher wie praktischer Hinsicht – als Entwerfer fällt dieser Schritt aber mitunter schwer, weil er einen Übergang vom lösungsorientierten zum problemorientieren Denken verlangt. Um diesen Aspekt zu betonen, wird hier zwischen generativem und algorithmischem Design unterschieden: So wie das generative Design formerzeugende Prinzipien bündelt und in einer einzelnen entwerferischen Aufgabe zusammenfasst und anwendet, löst das algorithmische Design generative Methoden aus der Umgebung einer konkreten Designaufgabe:

---

[11]) „Generative Art refers to any art practice where the artist uses a system, such as a set of natural language rules, a computer program, a machine, or other procedural invention, which is set into motion with some degree of autonomy contributing to or resulting in a completed work of art." (Galanter, 2003)

Algorithmisches Design ist generatives Design, das sich aus dem konkreten Bedeutungszusammenhang eines speziellen Entwurfskontextes emanzipiert.

Man kann hier zusammenfassend folgende verknüpfte Ebenen im Netzwerk des computergestützten Entwerfens ausmachen. Einmal gibt es die schon traditionell zu nennende Ebene der direkten Repräsentation von Entwurfsideen im *CAD*-Programm. Eine erste Metaebene ist die des generativen Entwerfens, die ebenfalls Entwurfsideen visualisiert, gleichzeitig aber auch Werkzeug und Medium ist, diese Ideen für ein konkretes Projekt zu entwickeln und abstrakt zu codieren. Eine weitere Metaebene ist die Ebene des algorithmischen Entwerfens. Hier wird nicht mehr auf die Repräsentation von Ideen gezielt, sondern auf die Verarbeitung von Information im Sinne einer generischen Problemlösung, die unabhängig von einer konkreten Aufgabe ist. Weitere Metaebenen können sich anschließen, etwa die des Schnittstellendesigns: Welche Problemstellungen sollen problemorientiert, welche lösungsorientiert angegangen werden? Oder die Ebene des Designs einer allgemeinen Architektur eines digitalen Entwurfsmodells: Welche Systemelemente sollen abgebildet werden, durch welche Beziehungen sollen sie miteinander verknüpft werden? Das Reflektieren über diese Ebene wäre eine weitere Metaebene, usw.

Diese verschiedenen Ebenen machen zwei Dinge deutlich. Zum einen bietet sich mit Methoden des generativen und algorithmischen Design die Chance, weit umfassendere und sensitivere Designmodelle zu entwickeln, als dies bisher möglich war. Zum anderen stößt der ‚Einzelkämpfer', der disziplinär Gebundene, hier sehr schnell an seine Grenzen. Solche Systeme sind nur zu entwickeln, wenn verschiedene Disziplinen ihr Wissen einbringen und vernetzen. Weiterhin ist zu sehen, dass jedes System, über das Design erzeugt wird, nicht die Realität ist, sondern Realität simuliert, also nur ein Modell ist, dessen Qualitäten von den Entwerfern definiert werden. Gerade die Objektivität der digitalen Informationsverarbeitung mag dazu verleiten, digital berechnete Entwürfe als ‚richtig', ‚gut' oder ‚wahr' zu betrachten – was sie auch sind, wenn nur die Ebene der Transformation von Informationen betrachtet wird. Dies bedeutet aber in keiner Weise, dass sie auch zwangsläufig eine positive Wertung erfahren, wenn sie in individuelle Bedeutungszusammenhänge gestellt werden. In diesem Sinn versuchen die folgenden Kapitel, einige Besonderheiten beim Entwurf generativer Algorithmen heraus zu arbeiten.

3.2 Constraints und Algorithmen

3.2.1 Entwurf generativer Algorithmen

Die Entwicklung eines generativen Programms läßt sich an Hand der folgenden Schritte nachvollziehen:

1) Festlegen der *Constraints*,
2) Definieren der Zielkriterien, auf beidem aufbauend:
3) Entwurf oder Adaption allgemeiner generativer Prinzipien,
4) Entwurf und Entwicklung detaillierter Regelsysteme, nach denen die Prinzipien ausgeführt werden sollen,
5) Überführen der Regelsysteme in konsistente Algorithmen,
6) Überführen der Algorithmen in ein Programm,
7) Implementieren des Programms,
8) Ausführen und Testen,
9) sehr wahrscheinlich: Überarbeitung des Programms.

Der kreative, entwerferische Teil dieses Ablaufs ist die eigentliche Algorithmenentwicklung, bei der sich aus *Constraints*, Zielen und formerzeugenden Prinzipien allmählich algorithmische Strukturen herausbilden. Da das Entwerfen generativer Algorithmen ähnliche Eigenschaften aufweist, wie andere Entwurfsprozesse auch, läuft es nur dann in der beschriebenen Reihenfolge ab, wenn der Entwerfer ein Anhänger der Schule des strukturierten Programmierens ist. Davon abgesehen kann es hierbei genauso chaotisch, unsauber und unstrukturiert zu gehen, wie beim Bearbeiten

jeder anderen Themenstellung auch: Ein Detailproblem der Sortierung beispielsweise, das nicht zufriedenstellend zu lösen ist, verlangt eine Revidierung der formerzeugenden Logik, die eigentlich verwendet werden sollte. Darüber werden wieder die Zielkriterien modifiziert oder andere *Constraints* als Ausgangspunkte gewählt. Dass dies das eigentliche, darüber liegende Ziel, die Formerzeugung für einen bestimmten Designzweck, nicht unberührt lassen kann, ist offensichtlich: Natürlich gibt es, bevor die Arbeit des Programmierens beginnt, ein mehr oder weniger deutliches Bild von dem, was erreicht werden soll. Trotzdem wird letztendlich das entstehen, was der Entwerfer mit seinen Fähigkeiten in einem angemessenen Zeitrahmen realisieren kann. Zudem wird sich erst beim Entwurf eines generativen Algorithmus genauer herauskristallisieren, wie die eigentliche Entwurfslösung beschaffen sein soll.

Dieser entwerferische Relativismus ist in jeder Art von Designprozess in mehr oder minder großem Umfang enthalten. Beim nicht-algorithmisch gestützten Entwurf einer Getränkeverpackung, eines Wolkenkratzers, eines Tisches oder einer Bohrmaschine ist es aber im Grunde gleichgültig, wie die letztendlich realisierte Form des Entwurfs zustande gekommen ist. Ob sie sich tatsächlich entlang einer Logik, die retrospektiv argumentiert werden kann, entwickelt hat, oder ob sie entlang von Einflüssen entstanden ist, die vielleicht nicht bemerkt wurden oder schlichtweg nicht mehr rekonstruierbar sind, ist für das Ergebnis ohne Belang, solange es alle gestellten Anforderungen erfüllt – nicht aber für die Methode: *Van Berkel* und *Bos* beklagen, dass ein gewisser Zwang zur *„retrospective justification"* oder *„after theory"* den Blick auf das verstellt, was im Entwurfsprozess wirklich geschieht *(vgl. Schumacher, 1999)* [12]. Was die methodische Herangehensweise angeht, gilt dies auch für den Entwurf generativer Algorithmen, nicht aber für das Ergebnis: Egal, wie chaotisch der Entwurfsprozess auch von statten gehen mag, es wird sich schließlich die oben skizzierte Struktur einstellen – nicht nur als gedankliches Modell, sondern tatsächlich in Form einer deterministischen Folge aus bedingten und bedingenden Anweisungen. Dies ist ein gravierender Unterschied zu konventionellen Entwurfsprozessen. Insofern sind die *Constraints* doch das eigentliche Fundament, auf dem der Entwurf generativer Algorithmen basiert und die bereits zu Beginn entsprechend sorgfältig überlegt werden müssen.

3.2.2 Constraints

Bedingungen sind, allgemein gesprochen, *„dasjenige, wovon ein anderes (das Bedingte) abhängt, was ein Ding einen Zustand, ein Geschehen möglich macht" (Schischkoff, 1991)* [13]. Bedingungen bezeichnen also Möglichkeitsräume und keine zwangsläufigen, ursächlichen Zusammenhänge. Auch für die *constraints*, die im deutschen Sprachgebrauch als Rand- oder Rahmenbedingungen vorkommen, gilt diese Definition. Zusätzlich haben sie aber die Konnotation, dass sie entweder als gegeben hingenommen werden oder nur schwer zu ändern sind. Und wenn sie geändert werden, hat dies weitreichende Konsequenzen für den Entwurfsprozess. Deswegen schwingt in der Bedeutung des Wortes ebenfalls mit, dass *Constraints* eher Limitierungen oder unliebsame Hindernisse bei der Suche nach Entwurfslösungen sind.

Constraints existieren nicht *per se*, sondern sie werden über die Wünsche, Wertvorstellung und Normen der beteiligten Akteure in den Entwurfsprozess eingebracht, die *Lawson (2006)* [14] die *„generator of constraints"* nennt: Entwerfer, Nutzer, Auftraggeber und Gesetzgeber. Weiterhin führt *Lawson* auch eine graduelle Unterscheidung ein, die von den rigiden *Constraints* (vom Gesetzgeber kommend) bis hin zu den flexiblen reicht, das sind dann diejenigen, die vom Entwerfer frei festgelegt werden können. Quer zu dieser Unterscheidung liegen dann noch die internen sowie externen *Constraints*: Erstere sind die Bedingungen, die aus dem Entwurfsgegenstand entstehen, letztere Bedingungen aus dem Kontext. Und schließlich führt *Lawson* noch die Kategorien radikal, symbolisch, formal und praktisch ein, die die *Constraints* mit den entsprechenden Eigenschaften umfassen: die ‚Radikalen' sind die wichtigsten und zumeist auch offensichtlichsten Randbedingungen, die Kategorie ‚symbolisch' umfasst Bedingungen der symbolischen Interpretation eines Entwurfs, ‚formal' bezeichnet die Bedin-

gungen, die Einfluß auf Proportion, Farbwahl etc. nehmen und die Sparte ‚praktisch' schließlich umfasst die Bedingungen, die in Verbindung zu Materialwahl, Konstruktion, Fertigung und Ausführung stehen. Abb. 01

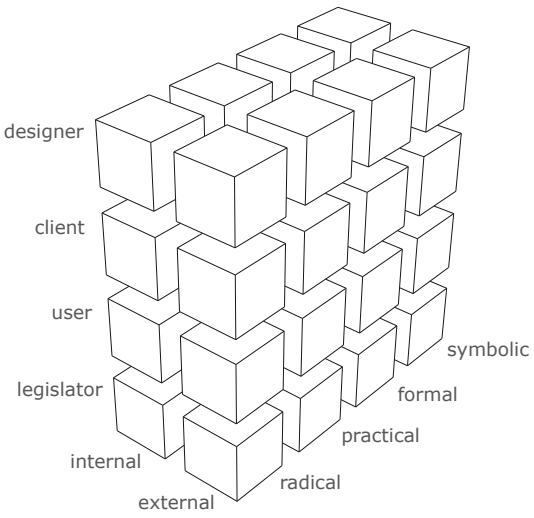


Abb. 01) Constraints als Modell von Designproblemen.
(nach: Lawson, 2006)

3.2.3 Constraints als *Design Driver*

Eine wichtige Lektion aus *Lawsons* differenzierter Betrachtung der *Constraints* besteht darin, dass sie nicht als gegeben hinzunehmen sind: Auch *Rittel (1970)* [15] weist darauf hin, dass als gesetzt angenommene Bedingungen – Sachzwänge – subjektiv konstruiert und keine objektiven Tatbestände sind. Randbedingungen sind mit mehr oder minder großem Aufwand veränderbar (rigide - flexibel) und können somit als Elemente betrachtet werden, über die sich Entwurfsprobleme modellieren und Lösungsräume eingrenzen lassen. Besonders interessant sind dabei natürlich die Bedingungen, die einen sehr weitreichenden Einfluss auf den Entwurfsprozess nehmen. *Kilian (2006)* [16] nennt diese Art von *constraints „design driver"* und betrachtet das bidirektionale Modellieren solcher Bedingungen als den Motor eines generativen Designprozesses, als Kernelemente zur Exploration von Lösungsräumen und zum Entwickeln innovativer Lösungen.

Im Kontext des algorithmischen Designs finden sich auch die internen und externen *Constraints* wieder, zum Beispiel in der Entwurfsphilosophie *Lynns (1999)* [17], bei dem Form erzeugt wird durch, *„ […] the internal combinations and relationsships of elements and their deformation within a larger contextual field"*: Aus Fragen der Konstruktion, Materialwahl oder der Raumorganisation können z.B. Bedingungen für das interne System eines Gebäudes abgeleitet werden, um daraus eine organisatorische Struktur zu entwickeln. Externe Bedingungen wie physikalische Einflüsse (Wind, Lichteinfall, Gravitation, etc.) oder auch eine bestimmte bauliche Situation, Wegeführungen und Besucherströme interagieren mit dieser Struktur und erzeugen in einem animierten Prozess sich kontinuierlich verändernde Formen.

Und schließlich interessieren das algorithmische Design vordringlich solche Bedingungen, die sich auch direkt in einer formalen Sprache (Programmiersprache) ausdrücken lassen: Die *Church-Turing*-Hypothese besagt, dass jedes Problem, das intuitiv berechenbar ist, auch von einer *Turing*-Maschine *(vgl. Turing, 1936)* [18] berechnet werden kann. Eine *Turing*-Maschine ist keine reale Maschine, sondern eine Art gedankliches Modell des mathematisch arbeitenden Menschen: Sie kann mit Hilfe eines Bandes, auf das Informationen gespeichert werden über drei einfache Operationen (Lesen, Schreiben, Bewegen) mathematische Berechnungen ausführen. Da jeder Computer eine *Turing*-Maschine simulieren kann, kann auch jeder Computer jedes intuitiv berechenbare Problem lösen. Das, was sich nicht berechnen läßt, kann folglich auch nicht durch eine Programmiersprache ausgedrückt werden. Eine Form, die durch ein Computerprogramm erzeugt wurde, kann zwar als anheimelnd, modern, schön, weich, hart, opulent, antiquiert, etc. interpretiert werden, es ist aber nicht möglich, prospektiv Regeln zu formulieren, die dann zielsicher eine bestimmte Assoziation hervorrufen: Menschliche Interpretation ist kontextsensitiv und mehrdeutig, formale Sprachen kontextunabhängig und eindeutig. Insofern kommen als Bausteine eines generativen Algorithmus vor allem folgende Typen von *Constraints* in Frage:

a) *geometrische* – Bedingungen aus und für räumliche Abmessungen und Beziehungen
b) *topologische* – Bedingungen aus und für Beziehungen zwischen Elementen, die eine topologische Einheit formen, die also unter Transformation unverändert bleiben
c) *konstruktive* – Bedingungen aus und für die Formulierung tektonischer Bezüge
d) *materialbezogene* – Bedingungen, die sich aus den Eigenschaften von Materialien ableiten lassen oder an sie gestellt werden
e) *physikalische* – Bedingungen aus physikalischen Größen wie Schwerkraft, Sonneneinstrahlung, Windlasten, Geld, etc.
f) *funktionale* – Bedingungen aus den Anforderungen, die eine Designlösung erfüllen muss, soweit sie quantitativ erfassbar sind.

Die Entwicklung generativer Algorithmen muss von eindeutig quantifizierbaren *Constraints* ausgehen. Soziale, psychologische, ästhetische etc. Bedingungen können indirekt Einfluss nehmen, über die Interpretation der generierten Ereignisse, nicht aber eindeutig im Programmcode formuliert werden. Trotzdem steht mit den genannten Kategorien eine recht umfangreiche Bandbreite an Elementen zur Verfügung, aus denen sich algorithmische Entwurfsräume bilden lassen. Es ist aber in *Kapitel 2.3* bereits gesagt worden, dass diese Bandbreite für ein konkretes Projekt nicht voll ausgeschöpft werden kann: Die Auswahl geeigneter Randbedinungen dient dazu, Lösungsräume einzuschränken und die Komplexität eines generativen Programms kontrollierbar zu halten. Dies ergibt sich sowohl aus der deterministischen Natur von Algorithmen, ist aber ebenso ein Gebot der Ökonomie von Programmierung.

3.2.4 Ökonomie und Komplexität von Algorithmen

Die Programmierung generativer Algorithmen ist ein Prozess, der umfangreiche Investitionen an Zeit, intellektueller Anstrengung und an materiellen Ressourcen verlangt *(vgl. Mitchell, 2003)* [19]. Die gestalterischen Möglichkeiten, die sich hier zumindest theoretisch ergeben könnten, werden in der Praxis gefiltert durch den ökonomischen Hintergrund eines Projekts. Sie müssen bezahlbar sein und irgend einen Vorteil gegenüber herkömmlichen Entwurfsmethoden bieten: Jede *CAD*-Software enthält vordefinierte geometrische Elemente wie Punkt, Linie, Kreisbogen oder *Spline*. Je nach Parametereingabe wird eine Instanz dieser Elemente im *CAD*-Programm visualisiert: Bei einem Punkt sind drei Parameter nötig – die Koordinaten des Startpunkts, bei einer Linie sechs – die Koordinaten von Start- und Endpunkt, bei einem Kreisbogen beispielsweise sieben – Startpunkt, Endpunkt, Radius, bei einem *Spline* eine gewisse Anzahl von Kontrollpunkten. Aus solchen Elementen werden Flächen, aus Flächen werden Solids, aus Solids Bauteile und aus Bauteilen schließlich komplette, detaillierte Designmodelle. Auf diesem Weg sind eine Reihe weiterer Instanzen geometrischer Elemente zu erzeugen, zu transformieren und räumlich zu organisieren – die entsprechenden Operationen verlangen wiederum bestimmte Parametereingaben. Aus der Perspektive der Ökonomie ist es jetzt am günstigsten, wenn mit möglichst wenigen Eingaben möglichst viele geometrische Instanzen erzeugt, transformiert und organisiert werden.

Ein Designprozess, in dem *1.)* viele verschiedene geometrische Elemente verwendet werden, die *2.)* vielen verschiedenen Transformationen unterzogen werden und bei dem *3.)* das ganze Designmodell ‚aus dem Bauch heraus',

---

[19]) „Creation of code […] is an investment of intellectual effort, time and money in the expectation of a future payoff. If you can exploit your insight into the regularities of a building type to write consice code that expandes a few parameters into a lot of explicit details, then the investment is a good one; a relatively small amount of coding time saves you a large amount of model construction time. The investment is even better if you use the code repeatedly, to generate varied instances as required in different contexts. But the investment is less attractive, if the code itself is lengthy and complicated; the coding may require more effort than the savings in model construction time." (Mitchell, 2003)

on-the-fly entwickelt wird, ohne dass sich vorab Regelmäßigkeiten definieren lassen, an denen die Entwicklung des Models angelehnt wird, ist aus Computerperspektive rein zufällig: Die Parametereingaben, die der Entwerfer bei der Konstruktion des Designmodells vorgenommen hat, entsprechen einer zufällige Zeichenkette aus Koordinatenangaben, Drehwinkeln, Rundungsradien, Skalierungsfaktoren, usw. in der sich keine geordnete Struktur erkennen läßt. In der Theorie der Zufallszahlen ist eine zufällige Zeichenkette definiert als eine Zeichenkette, deren kürzeste Beschreibung sie selbst ist. Für den angenommenen Fall bedeutet dies offensichtlich, dass auch kein Programm entwickelt werden könnte, dass irgendeinen ökonomischen Vorteil gegenüber dem händischen Arbeiten bieten würde.

Einige kurze Anmerkungen hierzu: Dass sich die Parametereingaben eines Designers als zufällige Zeichenkette darstellen lassen, heißt selbstverständlich nicht, dass diese Arbeitsweise einfach durch ein generatives Programm zu repräsentieren wäre, das zufällig ausgewählte Prozeduren mit zufälligen Werten füttert. Die Wahrscheinlichkeit, dass damit sinnvolles Design erzeugt werden könnte, ist so gering, dass sie schlichtweg unvorstellbar ist *(vgl. Dawkins, 1986)* [20]. Dass sich die Arbeitsweise eines Entwerfers als zufällige Kette von Werten repräsentiert, bedeutet auch nicht, dass der Entwerfer völlig ziellos arbeitet – er verwendet lediglich eine Logik, bei der aus Computersicht keine regelmäßigen Strukturen erkennbar werden – der Entwerfer kann dabei trotzdem das Gefühl haben, völlig stringent zu arbeiten. Und selbst wenn sich im Nachhinein aus den Parametereingaben eine geordnete Struktur mit repetitiven Elementen ableiten ließe, wäre dies für die Entwicklung eines Programms nutzlos, da der Entwurf dann ja schon existiert.

Betrachtet man andererseits einen Designprozess, bei dem *1.)* wenig verschiedene geometrische Elemente benutzt werden, die *2.)* nur wenigen unterschiedlichen Transformationen unterzogen werden und wo *3.)* vorab klare Regeln für den Aufbau des Designmodells festgelegt werden können, entsteht eine Zeichenkette aus Parametereingaben, in der sich Regelmäßigkeiten finden lassen – wie etwa beim Entwurf einer Gebäudefassade oder eines Regalsystems, die aus einem orthogonalen Raster entwickelt werden. Hier ist Programmierung aus ökonomischer Sicht dann lohnend, wenn der dadurch gesparte Konstruktionsaufwand größer ist als der Aufwand für die Programmierung. Dies ist mit großer Wahrscheinlichkeit geben, wenn der entsprechende Entwurf nicht nur einmal, sondern in verschiedenen Varianten mehrmals umgesetzt werden soll. Man kann also sagen, dass sich die Entwicklung von Skripten vor allem dann auszahlt, wenn die Beschreibung der generativen Regeln, nach denen ein Entwurf erzeugt wird, möglichst kurz ist und in kompakten Code gefasst werden kann.

Die Ökonomie der Algorithmenentwicklung ist somit direkt abhängig von der Komplexität eines Entwurfsgegenstands: Je geringer seine Komplexität ist, desto größer wird die Chance, dass sich der Aufwand der Programmierung später bezahlt macht, was *Mitchell (2003)* [21] für Entwürfe im architektonischen Kontext so formuliert: *„In summary, the complexity of a building type can (roughly speaking) be defined as the length of the shortest procedure that can expand parameter values into explicit instances. Simple types, such as standard office towers (particularly when they are to be instantiated repeatedly) repay investment in coding. Complex types (particularly when instances are not widely repeated) provide less attractive opportunities to achieve such payoffs.*" Diese Definition der

---

[20]) Dawkins (1986) hat die immens geringe Wahrscheinlichkeit, mit der rein zufällige Prozeduren etwas Sinnvolles ergeben könnten, am berühmten, unendlich lebenden Affen demonstriert, der solange auf der Schreibmaschine herumtippt, bis er zufällig „Methinks it is like a weasel" geschrieben hat, ein Zitat aus Shakespeares Hamlet: Im Vergleich zu einer kumulativen Selektionsstrategie (Evolutionsprinzip) ist eine zufällige Ein-Schritt-Selektion um einen Faktor unterlegen, den man sich schlichtweg nicht mehr vorstellen kann: Er beträgt ein sehr, sehr großes Vielfaches des Alters des Universums. Dabei ist das Hamlet-Zitat nur eine Kette mit 28 Zeichen, wobei für jede Position in der Zeichenkette nur 27 mögliche Werte in Frage kommen. Wie lange wäre wohl eine Zeichenkette die einen Stuhl modelliert und wieviele unterschiedliche Parameter könnten hier eingesetzt werden?

Komplexität eines Gebäudeentwurfs entspricht dem von *Gell-Man (1995)* [22] definierten, allgemeinen Maß für effektive Komplexität: Je länger die Beschreibung der Regeln eines Systems ist, desto komplexer ist dieses System. Demnach hätte etwas rein Zufälliges eine verschwindend geringe Komplexität. Dass in diesem Bereich generative Verfahren wenig sinnvoll sind, wurde bereits besprochen. Ebenfalls von sehr geringer Komplexität sind vollkommen geordnete Systeme, gerade hier sind generative Methoden von einem gewissen ökonomischen Reiz. Dort wo die Komplexität am höchsten ist, in einem Bereich der zwischen völliger Ordnung und völliger Unordnung liegt, sind generative Verfahren am interessantesten, leider aber auch am wenigsten rentabel.

Daraus läßt sich folgern, dass beispielsweise der Entwurf einer Bohrmaschine ein denkbar ungeeigneter Gegenstand für einen generativen Designansatz ist: Sie besteht aus sehr vielen verschiedenen geometrischen Instanzen, die daraus erzeugten Bauteile sind sehr unterschiedlich geformt, sie stehen in vielfältigen räumlichen Bezügen, die sich wiederum aus dem Zusammenspiel unerschiedlicher Anforderungen der Ergonomie, der Konstruktionsökonomie, der Statik, der funktionellen Anbindung an andere Bauteile, etc. ergeben. Kurz gesagt ist eine Bohrmaschine ein sehr komplexer Gegenstand. Dagegen ist beispielsweise das Klischee einer Plattenbausiedlung – freilich nur aus der Perspektive der ökonomischen Formerzeugung – ein minder komplexes Problem, weil sich sehr knappe erzeugende Regeln formulieren lassen: Ähnliche Baugrundstücke, orthogonale Grundraster als durchgängiges Gestaltungsprinzip, gleiche Fassadengestaltung, gleiche Wohnungsgrundrisse und Zuwegungen, und das alles in einem klar und einfach strukturierbaren formalen Zusammenhang. Glücklicherweise ist Wirtschaftlichkeit zwar ein Gebot, aber kein Diktat entwerferischer Arbeit. Sonst würden generative Algorithmen ausschließlich unter dem Aspekt der ökonomischen Rationalisierung, nicht aber als ein Medium der Gestaltung diskutiert. Ein nicht näher bestimmbarer Faktor, der ebenfalls in eine Wirtschaftlichkeitsbetrachtung der Programmierung mit einfließt, ist natürlich auch die Güte des Resultats, das man damit zu erzielen hofft. Je vielversprechender dieses Ergebnis erscheint, umso mehr wird man sich auch daran wagen, komplexeren Programmcode zu erzeugen. Trotzdem sind ökonomische Erwägungen mit ein Grund, dass Probleme mit einer geringeren effektiven Komplexität der Bearbeitung durch generative Verfahren eher zugänglich sind. Auch spielt das Automatisierungspotential, das in solchen Verfahren geborgen liegt und mit dem der Routineanteil an Designaufgaben signifikant beschleunigt werden kann, ebenfalls eine wichtige Rolle bei der Entscheidung für oder wider generative Verfahren.

Dies stützt die zweite These, die in der Einführung der Arbeit zur Konstruktion algorithmischer Entwurfsräume aufgestellt wurde, nach welcher der Entwurf eines generativen Algorithmus immer auch einen klar abgegrenzten und gut identifizierbaren Lösungsraum erzeugt. Dies ergibt sich nicht nur aus der deterministischen Natur von Algorithmen. Es ist auch nicht nur eine Frage des eingeschränkten Spektrums an Themen, die mit Computeralgorithmen bearbeitet werden können, wie im vorigen Kapitel an Beispiel der *Constraints* besprochen. Es ist ebenso eine Frage der Ökonomie des Entwerfens. Man kann dies erstaunlicherweise vor allem an Arbeiten im experimentellen und wissenschaftlichen Kontext nachvollziehen, wo die erzeugten Ergebnisse über-

---

[22]) „A measure that corresponds much better to what is usually meant by complexity in ordinary conversation, as well in scientific discourse, refers not to the length of a concise description of an entity (which is roughly what AIC is), but to the length of concise description of a set of the entity's regularities. Thus something almost random, whith practically no regularities would have effective complexity near zero. So would something completely regular (…). Effective complexity can be high only a region intermediate between total order and complete disorder." (Gell-Man, 1995)

wiegend in Reinform vorliegen und nicht durch einen übergeordneten Projektkontext verdeckt werden. Manchmal genügen nur wenige Darstellungen verschiedener Lösungsinstanzen um eine Vorstellung zu bekommen, welche Art von Algorithmen hier im Hintergrund tätig waren.

Die ökonomische Perspektive verdichtet aber auch die beiden anderen, eingangs aufgeführten Thesen zur Konstruktion algorithmischer Entwurfsräume. Soweit es den ersten Aspekt betrifft, nach dem ein Entwurfsproblem gedanklich aus der Perspektive einer generativen Methode konstruiert wird, weil sich die Denkweisen von der Ebene des Algorithmendesigns auf die Ebene des eigentlichen Entwurfsgegenstands transportieren, ist es natürlich auch von großem wirtschaftlichen Reiz einen Entwurf mit einem generativen Algorithmus zu entwickeln, der bereits fertig auf der Festplatte liegt. Wird die dadurch freiwerdende Zeit genutzt, diesen Entwurf genauer zu untersuchen und eine größere gestalterische Qualität zu realisieren, ist dies natürlich begrüßenswert, wie bereits in *Kapitel 2.2* angesprochen. Es ist aber auch naheliegend, bereits entwickelte generative Algorithmen zu nutzen, um, salopp gesagt, auf die Schnelle ein paar Regale, Inneneinrichtungen oder Bürogebäude zusammen zu klickern. Automation bedeutet Rationalisierung, Rationalisierung bringt ökonomische Vorteile. Algorithmisches Entwerfen kann also auch schlicht Dienstleistung zur reinen Designproduktion bedeuten.

Dann wurde als dritter Aspekt die These aufgestellt, dass sich das eingeschränkte Lösungsspektrum einzelner algorithmischer Entwurfsräume wieder ausweiten kann, in dem viele dieser Entwurfsräume, respektive der zugrunde liegende Programmcode, zusammengeführt wird – individuelle wie kollektive Datenbanken, die abrufbares Wissen zur Formerzeugung sammeln, zur Verfügung stellen und ausbauen *(vgl. Kapitel 2.3)*. Gerade hier ist der ökonomisch aufwendigste Teil von Programmierung und damit auch eine der großen Hürden für eine solche Perspektive: Entwicklung von austauschbaren Programmmodulen, Programmierung von Schnittstellen zur Datenein-, über- und ausgabe, Anbindung an Standard-*Software*, die Unterscheidung zwischen generischen und generativen Lösungsansätzen, etc. Hier sind verschiedene Entwicklungsszenarien denkbar, kostenpflichtige und private Netzwerke, die Entwicklung kommerzieller *Software Plug-Ins,* die bestimmte generative Anwendungen gleich mitliefert, etc. Auf jeden Fall wird es so sein, dass auf Grund des Aufwands der Spagat zwischen Algorithmenentwurf einerseits und Anwendung im konkreten Designkontext anderseits für den einzelnen Entwerfer nur sehr schwer zu leisten sein wird und mehr in interdisziplinären Teams stattfinden wird.

Der wichtigste Aspekt der ökonomischen Perspektive ist jedoch, dass in ihr der eigentliche Grund liegt, warum sich mit Hilfe generativer Verfahren überhaupt neue gestalterische Potentiale erschließen lassen: Nur weil sich über Computerprogramme Arbeitsschritte von Entwurfs- und Fertigungsprozessen beschleunigen sowie automatisieren und sich durchgängige digitale Prozessketten entlang aller Phasen eines Entwurfs etablieren lassen, sind viele der Entwürfe, die durch ihre ungewohnte Formensprache beeindrucken überhaupt möglich. Man denke dabei an die vielen architektonischen Arbeiten jüngerer Zeit, die mit Freiformen arbeiten. Die Umsetzung einer Freiform in eine Konstruktion erzeugt eine Vielzahl unterschiedlicher Bauteile, oft genug ist jedes Bauteil ein Unikat, wie beispielsweise die Scheiben des *BMW-Bubbles* von *Franken (Kap. 6, Abb. 13)* oder des Kunsthauses in Graz von *Cook* und *Fournier (Kap. 1, Abb. 01)*. Müssten bei solchen Entwürfen alle Teile einzeln von Hand konstruiert, gezeichnet und gefertigt werden, gäbe es sie mit ziemlicher Sicherheit nicht. Wo die Ökonomie der Programmierung also auf der einen Seite das Lösungsspektrum einschränkt, eröffnet sie auf der anderen Seite neue Anwendungsperspektiven.

3.2.5 Eigenschaften von Algorithmen

Bisher wurde in dieser Arbeit viel von Algorithmen gesprochen und von einigen Konsequenzen, die sich aus ihren Eigenschaften für das Entwerfen ergeben, aber noch keine genauere Beschreibung davon gegeben. Dies soll hier nachgeholt werden. Im praktischen Kontext sind allgemeine

Definitionen dieses Begriffs gebräuchlich, etwa der Art, wie sie die *Wikipedia* liefert: *„Unter einem Algorithmus versteht man allgemein eine genau definierte Handlungsvorschrift zur Lösung eines Problems oder einer bestimmten Art von Problemen." (www.wikipedia.org, 2005)* [23]. Es ist beliebt, solche Definitionen über Metaphern wie Bedienungsanleitungen oder Kochrezepte zu veranschauliche, die auf der einen Seite helfen, diesen abstrakten Begriff fassbarer zu machen, die auf der anderen Seite aber auch in die Irre führen, wenn davon ausgegangen wird, das Algorithmen als Computerprogramme implementiert werden sollen. Anweisungen wie: ‚Eier nach Belieben' oder ‚mit Salz und Pfeffer abschmecken' sind für Menschen umsetzbar, für die Ausführung in einem Computerprogramm aber viel zu ungenau beschrieben. Eine präzisere Definition ist die folgende, die zwischen Prozedur und Algorithmus unterscheidet: *„Eine Prozedur liegt vor, wenn eine Folge von durchführbaren Handlungsanweisungen so abgefaßt ist, daß bei jedem Schritt klar ist, was der nächste Schritt sein wird. Ein Algorithmus ist eine Prozedur, die nach endlichen vielen Schritten mit einem Ergebnis endet. Das Ergebnis kann auch sein, dass es kein Ergebnis gibt." (Dehlinger, 2002)* [24] Hieraus lassen sich alle wichtigen Eigenschaften von Algorithmen ableiten:

1) *Ausführbarkeit* – Jede einzelne Handlungsanweisung, die in einem Algorithmus enthalten ist, muss auch tatsächlich ausführbar sein.
2) *Terminierung* – Der Algorithmus darf nur endlich viele Schritte enthalten, damit er auch nach einer endlichen Zeit anhält.
3) *Determiniertheit* – Der Algorithmus muss bei jedem Durchlauf unter denselben Voraussetzungen *(Input)* das gleiche Ergebnis *(Output)* liefern.
4) *Determinismus* – Nach jeder ausgeführten Handlungsanweisung muss eindeutig klar sein, was der nächste Schritt sein soll.

Diese Eigenschaften haben sowohl konzeptionelle (den Entwurf eines Algorithmus betreffende) als auch praktische Implikationen – und sie sind ein Grund dafür, warum das Arbeiten mit Algorithmen aus Entwerferperspektive zunächst einschränkend und gewöhnungsbedürftig wirkt. Während es in einem Entwurfsprozess oft genügt, wenn eine Folge von Handlungen einmal zum gewünschten Ziel führt, sollten Computerprogramme natürlich in jedem Fall ausführbar sein, was kein triviales Problem ist, wie man beispielsweise an Betriebssystemen immer wieder selbst feststellen darf. Für die Entwicklung generativer Algorithmen muss also möglichst jeder Fall, der eintreten kann, auch bedacht werden: Selbst wenn nur ein Spezialfall interessiert, muss eine gewisse Allgemeingültigkeit zwangsläufig mit entwickelt werden: Lösungsraum vor Einzellösung. ‚Terminierung' ist sowohl ein handwerkliches als auch konzeptionelles Problem, das sich in den Endlosschleifen äußert: Eine Bedingung kann nicht erfüllt werden, das Programm ‚hängt'. Sie ist aber auch ein Effizienzkriterium. Aus praktischer Sicht genügt es natürlich nicht, wenn ein Programm irgendwann anhält, es sollte möglichst wenige Schritte durchführen und die möglichst schnell. Es kann also nicht nur irgendein Regelsystem zur Formerzeugung entworfen werden, sondern es sollte auch möglichst effizient ausführbar sein – dies gilt vor allem für komplexere generative Algorithmen und bindet entsprechend zusätzlich entwerferische Aufmerksamkeit.

Die beiden Eigenschaften ‚Determinismus' und ‚Determiniertheit', brauchen noch eine weitere Erklärung. Gerade bei formerzeugenden Algorithmen ist es oft gewünscht, dass sich eine Variation des Ergebnisses nicht nur über direkte Manipulation der Eingabe- und Verarbeitungsparameter erreichen läßt, sondern dass das Programm selbst in der Lage ist, gewisse Variationen vorzunehmen – das Ergebnis also nicht determiniert ist. Dies ist mit Hilfe der sogenannten nicht-deterministischen Algorithmen möglich. Solche Algorithmen arbeiten mit zufälligen Elementen, ein Beispiel sind parametrische Algorithmen, bei denen gewisse Eingabewerte zur Erzeugung geometrischer Instanzen über Zufallszahlen definiert werden und dadurch automatisch eine Bandbreite unterschiedlicher Designmodelle entsteht. Zufallszahlen können ebenfalls verwendet werden, um ein Computerprogramm innerhalb einer gewissen Wahrscheinlichkeit Entscheidungen treffen zu lassen, so dass sie selbst

ihren Programmablauf verändern können. Streng genommen sind auch solche Algorithmen deterministisch, da die durch Computer generierten Zufallszahlen durch Formeln erzeugt werden, die bei gleichem Eingabewert *(Seed)* immer die gleiche Reihe von Zufallszahlen erzeugen. Dieses Problem wird umgangen, in dem ein variabler *Seed*, etwa aus der Systemuhr eines Rechners abgeleitet, verwendet wird. Unabhängig von dieser Definitionsfrage sind solche Algorithmen zumindest in der Anschauung nicht-deterministisch, weil man mit ihrer Hilfe nicht genau vorhersehbare Ergebnisse erzeugen kann. Eine weitere Möglichkeit, ein zufälliges Element in die Ausführung eines Algorithmus zu integrieren, ist die Interaktion mit einem Anwender. Das bedeutet beispielsweise, dass ein Nutzer in ständiger Interaktion mit der Ausführung eines generativen Programms steht oder dass der Programmablauf an einer oder mehreren Stellen unterbrochen wird, an denen der Nutzer Parametereinstellungen oder andere Eingabewerte verändern kann oder ganz gezielt den Aufruf einer bestimmten Prozedur vornimmt. Ein Beispiel für einen solchen Algorithmentyp sind interaktive genetische Algorithmen, bei denen versucht wird, über ästhetische Urteile einen rechnergestützten Evolutionsprozess in eine bestimmte Richtung zu lenken.

Die Freiheitsgrade, die mit Hilfe nicht-deterministischer, generativer Algorithmen möglich sind, sind nicht zu vergleichen mit dem Gefühl von Freiheit, die ein Entwerfer im Skizzieren oder auch im Umgang mit einem *CAD*-System haben mag. Wenn eine generative Logik mit zufälligen Elementen gewürzt werden soll, ist genau und minutiös zu planen, an welchen Stellen und in welchem Ausmaß dies geschehen soll. Und natürlich sind dann alle Konsequenzen, die sich dafür für den Ablauf eines Programms ergeben, ebenso genau zu bedenken: Zufallskomponenten in generativen Algorithmen sind ein genau und eindeutig kalkuliertes Element. Wie diese Art von computergenerierten ‚Überaschungen' zu bewerten ist, kommt später unter dem Thema *Versioning (Kapitel 4.3)* nochmal etwas ausführlicher zur Sprache.

Diese Eindeutigkeit schließlich, mit der die Vorstellung einer ästhetischen Konzeption in eine präzise, ‚computertaugliche' Logik überführt werden muss, ist eine weitere Schwierigkeit beim Entwickeln generativer Algorithmen: Das Kontinuierliche (analog) muss durch das Diskrete (digital) abgebildet werden.

3.2.6 Algorithmensprache und Entwerfersprache

Diese grundsätzliche Schwierigkeit des *„Teamwork zwischen Künstler und Computer" (Nake, 1967)* [25] liegt in der Verwendung verschiedener Sprachen begründet, konkreter in dem Transformationsprozess der stattfinden muss, wenn eine umgangssprachlich (natürliche Sprache) formulierte Idee in eine Computersprache (formale Sprache) [26] umgesetzt werden muss. Aussagen in natürlichen Sprachen sind mehrdeutig und werden entsprechend dem jeweiligen Kontext decodiert. Formale Sprache sind eindeutig und darin formulierte Aussagen demnach auch kontextunabhängig [27]. Vom Entwerfer über die algorithmische Beschreibung zu dem daraus entwickelten Programm bis hin zum rechnergesteuerten Ausgabegerät werden einerseits die Anforde-

---

26) Aufbau einer formalen Sprache:

symbol: a letter, a mark or character

finite alphabete: a non empty set of symbols; $V = \{ a_1, a_2, a_3, \ldots a_n \}$

word: a sequence / string of symbols; $P = a_1\ a_2\ a_3 \ldots a_n$

language: a set of words; $L = \{ P_1, P_2, P_3, \ldots P_n \}$

rewriting rule: an ordered pair of words; (P, Q) or P -> Q

generative grammar: an ordered four-tuple

$G = (V_n, V_T, S, F)$ where:

V_N = non-terminal symbols

V_T = terminal symbols

S = starting Symbol , element from V_N

F = finite set of rules

(nach Chomsky, 1957; aus: Dritsas, 2004)

rungen an die Korrektheit der Syntax der verwendeten Sprache größer. Whärned desir Habltstaz enier narütichlen Sraphce vom Menschen (wahrscheinlich) noch gelesen werden kann, sind Programmiersprachen weitaus weniger fehlertolerant. Andererseits wird der Sprachraum enger (es kann weniger mit einer Sprache ausgedrückt werden). Wenn also beispielsweise die Idee einer ästhetischen Konzeption vorliegt, die umgangsprachlich, auch unter der zu Hilfenahme von Skizzen, problemlos zu beschreiben ist, bedeutet dies nicht selbstverständlich, dass daraus auch eine präzise algorithmische Beschreibung abgeleitet werden kann. Ist ein solche Beschreibung möglich, kann es auch sein, dass sie Aussagen enthält, die durch eine Computersprache nicht ausgedrückt werden können – Skriptsprachen erlauben beispielsweise keine gleichzeitige Einflussnahme verschiedener formerzeugender Parameter, sondern nur eine sequentielle. Und schließlich kann auch nicht jeder *Output*, der von einem Computerprogramm erzeugt wird, auch tatsächlich von einem Ausgabegeräte verarbeitet werden. Einfach gesagt muss das Konzept eines Entwurfs, dass generativ bearbeitet werden soll, so sein, dass es diesen Transformationsprozess unbeschadet durchlaufen kann – was nicht trivial ist.

Dererlei Schwierigkeiten und Sperrigkeiten, die durch Anforderungen im Kommunikationsprozess mit dem Rechner auftreten, fasst *Kalay (2004)* [28] sehr treffend zusammen: *„It is relatively easy to communicate information from computers to humans, who posses the intelligence needed to understand textual, numerical, graphical and auditory messages. But it is frustratingly difficult to communicate information from humans to computers, who lack the intelligence and the ability to interpret messages, unless they are coded in a completely unambigous manner."*

In diesem Kommunikationsprozess muss sich der Entwerfer die streng formale Logik der Programmiersprachen zu Eigen machen, umgekehrt ist dies nicht möglich: Eine Kommunikation mit Computern in natürlicher Sprache wird auf absehbare Zeit nicht praktikabel sein. Formale Sprache basiert auf einer Unterteilung von Gedanken in diskrete Einheiten und deren Verbindung durch eindeutige Beziehungen. Es ist eine streng rationale, quantitativ orientierte und statische Logik, die sich im Entwerfen dort hervorragend anwenden läßt, wo es um Beziehungen von Geometrie und Material geht. Solche Verknüpfung lassen sich in der Syntax formaler Sprache formulieren. Geht es aber um Beziehungen zwischen Menschen und Dingen oder Menschen untereinander, versagt die formale Logik: Solche Beziehungen sind kontinuierlichen Veränderungen unterworfen, die sich in wenn-dann-Beziehungen oder mathematischen Begriffen nur unzureichend formulieren lassen. Diese Trennlinie sollte ein Entwerfer im Auge behalten, wenn er mit den formalen Programmiersprachen arbeitet, um nicht eine Logik des Materials unbewußt auf eine Logik menschlicher Beziehungen zu übertragen: *„Die Grenzen meiner Sprache […] bedeuten die Grenzen meiner Welt" (Wittgenstein, 1921)* [29]. Im Falle der Programmiersprachen ist dies richtig.

Dieses Kapitel sollte zeigen, dass algorithmisches Design mit einer ganzen Reihe von Besonderheiten verbunden ist, die die Anwendungsmöglichkeiten algorithmischer Methoden limitieren – die Schnittstelle zur Synthese mensch-

---

27) Naom Chomskys (1957) Forschungsarbeit „Syntactic Structures" hatte zum Ziel, die fundamentalen Prinzipien der Sprachkonstruktion zu erklären und eine Theorie zu entwickeln, mit der Sprache abstrakt, ohne Bezug zu einer bestimmten Sprache untersucht werden kann. Das Problem bei der Ableitung einer formalen aus einer natürlichen Sprache ist, dass menschliche Kommunikation mehrdeutig ist. Ob ein ‚Haufen Mist' tatsächlich ein ‚Haufen Mist' ist, hängt vom Kontext ab. Ob der ‚Haufen Mist' überhaupt als sinnvolle Aussage innerhalb eines gewissen Kontexts decodiert werden kann, hängt vom kulturellen Hintergrund ab. Beides, Kontext und kultureller Hintergrund, kann nicht mathematisch dargestellt werden. (vgl.: http://de.wikipedia.org/wiki/Chomsky-Hierarchie; Stand: Mai 2006)

licher Kreativität und Computerintelligenz. Auf der anderen Seite sind es eben diese Besonderheiten, die neue Anwendungsperspektiven eröffnen, in praktischer wie in erkenntnistheoretischer Hinsicht: *„Was sich überhaupt sagen läßt, läßt sich klar sagen; und wovon man nicht sprechen kann, darüber muss man schweigen"* sagt *Wittgenstein (1921)* [30]. Dem kann man entgegenhalten, dass es eben genau die Aufgabe der Entwerfer ist, das, was nicht klar definiert werden kann, zu entwickeln und zu präzisieren. Wenn dabei neue Sprachen erschlossen und entwickelt werden, ist dies nur zur begrüßen. Es erweitert das Spektrum der Handlungs- und Erkenntnismöglichkeiten. Diesem Aspekt soll im folgenden Kapitel nachgegangen werden.

4. Algorithmische Entwurfsräume

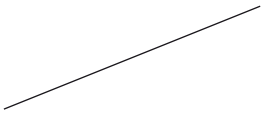
4.1 Ebenen algorithmischen Entwerfens

4.1.1 Entwerfen auf Metaebene

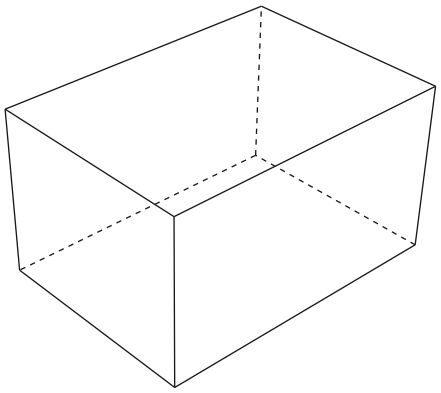
In gewohnten methodischen Kontexten, aus der Anschauung heraus, kann Entwerfen als ein Zusammenspiel von ‚sichtbaren' und ‚unsichtbaren' Modellen *(vgl. Schein, 2003)* [01] betrachtet werden. Das ‚*Unsichtbare*' steht für mentale Modelle, für Ideen, Konzepte, Wertvorstellungen usw. An den ‚sichtbaren' Modellen manifestieren sich die Vorstellungen des Entwerfers, werden physisch wahrnehmbar und überprüfbar. Dies ist kein digitaler Prozess, in dem Ideen von Umsetzungen klar getrennt wären, sondern ein Prozess des ständigen Übergangs von Idee zu Modell und zurück, ähnlich wie es *Kleist (1805)* [02] mit *„Die allmähliche Verfestigung des Gedankens beim Reden beschrieben hat."* Die Sprache des Entwerfers ist die Modellbildung, an ihr und mit ihr klärt, verfestigt und detailliert sich die entwerferische Idee.

Die Modelle des Entwerfens lassen sich in drei Hauptgruppen gliedern: Ikonische Modelle sind solche, die direkte bildhafte Bezüge ermöglichen und in ihrer äußeren Erscheinung einem tatsächlichen Produkt sehr nahe kommen. Symbolische Modelle, wie etwa eine technische Zeichnung, bei der Zeichen und Symbole zur Darstellung verwendet werden und analoge oder vielleicht besser, Analogie-Modelle, mit denen eine Entsprechung zum realen Verhalten eines Entwurfs geschaffen wird. Beispiele dafür sind Modelle zur statischen Überprüfung, Bewegungs- oder Funktionsmodelle *(vgl. Dehlinger, 1990)* [03 / Abb. 01].

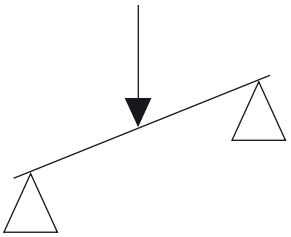
Abb. [01]) Entwurfsmodelle:



Eine einfache Linie kann ein ikonisches Modell sein, wenn sie tatsächlich als Bild für eine Linie steht.



Sie ist dann symbolisch, wenn sie beispielsweise für eine Körperkante eines Volumens oder die Verbindung zweier Elemente eines Diagramms steht.



Sie kann auch Bestandteil bei der Darstellung eines Analogie-Modells sein, wie z.B. bei diesem Balken-auf-zwei-Stützen - Modell.

```
Dim arrStart, arrEnd
arrStart = Array ( 0,0,0)
arrEnd = Array ( 100,50,0)
Rhino.AddLine arrStart, arrEnd
```

Mit den symbolischen Modellen der Programmiersprachen lassen sich weitere symbolische, aber auch ikonische und Analogie-Modelle erzeugen.

Solche Arten von Modellen werden in zweifacher Weise genutzt: Einerseits um sich beim Entwerfen die eigenen Gedanken vor Auge zu führen, zu vergegenwärtigen, zu präzisieren und zu überprüfen. Andererseits als Kommunikationsmittel, um sie anderen begrifflich und begreiflich zu machen *(vgl. Kaley, 2004)* [04]. Im Kontext des computerbasierten Entwerfens, dessen Modelle immer symbolischer Natur sind, ist es mittlerweile üblich, eine weitere Unterscheidung zu treffen: Die zwischen physischen und virtuellen Modellen. Die immer besser werdenden Visualisierungsmöglich-

keiten von *CAD*-Anwendungen, die bis zu photorealistischen Darstellungen (in annähernd Echtzeit) reichen, verführen dazu, virtuelle Modelle direkt in gedankliche Verbindung mit physischen Modellen zu bringen. Dabei sind *Computer* abstrakte Maschinen, sie verarbeiten Informationen über Funktionen, sie sind diagrammatisch, unabhängig von Substanz, Ausdruck und Inhalt *(vgl. deLanda, 2002)* [05]. Alles, was diesbezüglich in den Symbolen von Programmiersprachen ausgedrückt wird, ist menschliche Interpretation und Wahl, sowohl auf der Seite der Informationseingabe, der Art der Verarbeitung, als auch auf der Seite des Ergebnisses. Dies gilt auch, wenn über die geometrische Ebene hinaus mit der Simulation physikalischer Kräfte gearbeitet wird: Daraus läßt sich nicht schließen, dass das Verhalten eines virtuellen Modells mit dem seines physischen Pendants gleich sein muss *(vgl. Franken, 2003)* [06].

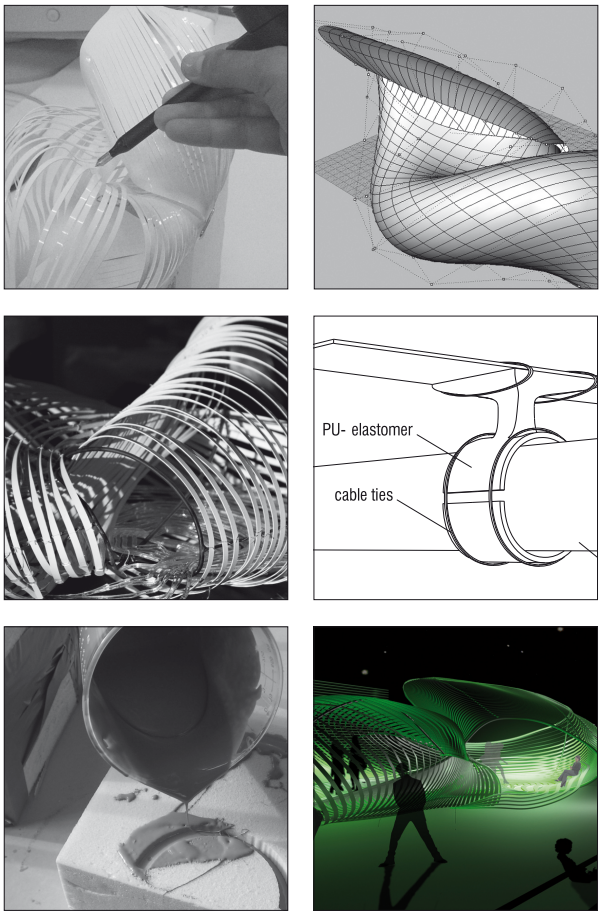


Abb. [02]) Wechselspiel zwischen virtuellen und physischen Modellen.
(Beermann et al, 2006)

Die Erkenntnis, dass sich Tektonik und Mechanik nur mit Einschränkungen durch Geometrie und physikalische Simulation darstellen lassen, insbesondere bei Entwürfen mit freien Formen, hat den ständigen Wechsel vom Virtuellen zum Physischen zu einem festen Bestandteil computergestützen Designs gemacht. Die Methodik *Gehrys* ist der Prototyp dieser Art des Arbeitens, das Bildbeispiel Abb. 02 zeigt ein Projekt von *Beermann* und *Seeland (2006)* [07]: Hier wird ein einfaches Modell, das ein physikalisches Formerzeugungsprinzip repräsentiert, digitalisiert und aus den resultierenden Punkten eine *NURBS*-Fläche dritten Grades erzeugt, deren Krümmungsverhalten eine gute Näherung an das Biegeverhalten des physischen Modells ist und die gleichzeitig die Ungenauigkeiten des kleinmaßstäblichen Ursprungsmodells ausgleicht. In einem Prozess der wechselseitigen Informationsanreicherung wird nun mehrfach zwischen virtuellen und physischen Modellen unterschiedlichen Maßstabs gesprungen. Die dadurch gewonnenen Erkenntnisse fließen als zunehmende Detaillierung sowohl in die physischen als auch die virtuellen Modelle ein.

Beim generativen Entwerfen kommt nun durchgängig eine weitere, symbolische Modellierungsebene hinzu: Die der Programmiersprachen. Der Vorteil dieser Metaebene liegt darin, dass mit ihrer Hilfe wieder die gewohnten symbolischen, ikonischen und Analogie-Modelle erzeugt werden können, sowohl auf virtueller als auch auf physischer Ebene, über die *CNC*-Verfahren. Dadurch wird der Entwurfsprozess aber auch komplexer. Beim nicht-generativen Entwerfen geht die Erzeugung, Veränderung und Bewertung (oder die Interpretation der Bewertungsergebnisse) von Modellen direkt über den Entwerfer. Beim generativen Design dagegen repräsentieren sich die Entwurfsentscheidungen auf der symbolischen Metaebene der generativen Algorithmen. Diese wiederum sind selbst automatisch in der Lage, Modelle zu erzeugen, zu bewerten und zu verändern. Die bisherigen Rückkopplungen zwischen Modellen und Entwerfenr bleiben dabei weiterhin möglich Abb. 03. Hinzu kommt, dass der Entwurf von Algorithmen selbst wieder eigene Modelle benötigt.

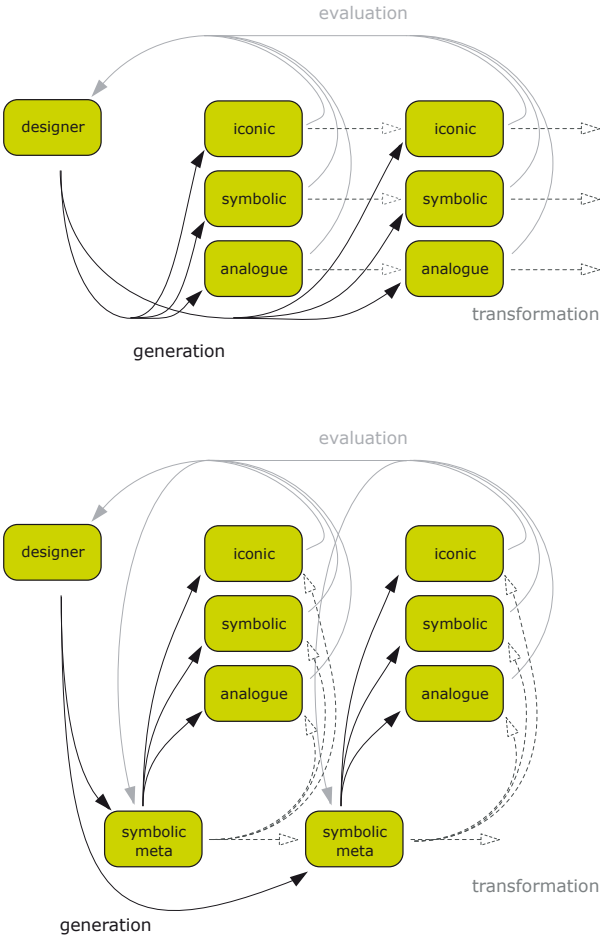


Abb. [03]) Vergleich der Modellierebenen von generativem und nicht-generativem Design.

Diese zunehmende Komplexität ist der unvermeidliche Preis einer Evolution computerbasierter Designmethoden. Zusammen mit den allgemeinen Voraussetzungen und Konsequenzen, die der Entwicklung generativer Entwurfsmethoden inhärent sind *(vgl. Kapitel 3)*, muss der Umgang damit geübt werden und verlangt die Entwicklung neuer Entwurfsstrategien und -haltungen. Viele der Publikationen über generatives Design geben einen plastischen Eindruck von dieser Komplexität, über detaillierte Schilderungen der verwendeten Werkzeuge und Methoden, aber auch den Werten, Zielen und Erwartungshaltungen, die mit den Arbeiten verbunden werden. Böswillige Zungen könnten behaupten, dass die instrumentellen und methodischen Grundlagen solcher Arbeiten in krassem Mißverhältnis zu den erzielten Ergebnissen stehen – zu Ungunsten der Letzteren. *Speaks (2002)* [08] dagegen sieht in computerbasierten, generativen Designansätzen ein Medium, in dem sich *„non-metaphysical, experimental forms of thinking as doing“* ausdrücken, eine dem experimentell-praktischen zugewandte Form des Denkens als Handeln. Entsprechend führt das angewandte generative Entwerfen und die Begegnung mit der innewohnenden Komplexität zur Reflexion und Revision entwerferischer Positionen, das Hinterfragen gewohnter Entwurfsstrategien drückt sich in der experimentellen Suche nach Anwendungsfeldern für generative Strategien aus. *Speaks (2002)* [09] sieht darin das Potential, entwerferische Tätigkeit in Bereiche zu führen, in denen pures Problemlösen (das er kennzeichnet durch das fraglose Akzeptieren vorgegebener Parameter) abgelöst wird durch ein Verständnis von Design als Gelegenheit zur Innovation. Es mag vielleicht sogar richtig sein, dass die Komplexität, die in generativen Designprozessen erzeugt wird, nicht in jedem Fall durch die Resultate gerechtfertigt wird. Aber schließlich müssen solche neuen Entwurfsformen auch erst einmal gedacht und experimentiert werden. Dazu gehört auch, Strategien zu entwickeln, wie diese Komplexität handhabbar gemacht werden kann.

4.1.2 Programm und Stil

Die Grundlage eines generativen Programms läßt sich, jenseits der Ebene seiner konkreten Anwendung, als Diagramm begreifen, das Randbedingungen definiert und Beziehungen zwischen einzelnen Elementen einer Struktur organisiert – und eine Struktur ist bereits die Abstraktion vom konkreten Einzelfall. Es sind hier drei Aspekte zu betonen:

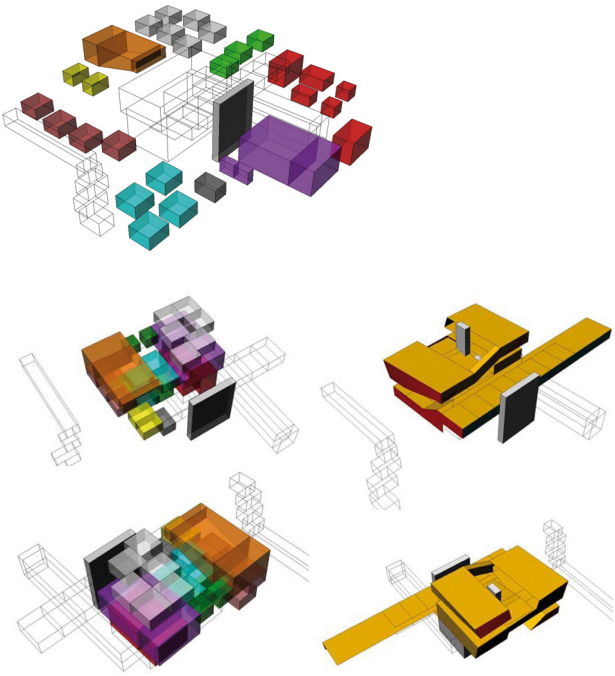
1) Auf der Ebene der Algorithmen selbst entsteht kein Design oder keine Architektur. Dies geschieht in den Köpfen der Entwerfer, während die abstrakten Diagramme in Algorithmen überführt und mit konkreten Entwurfsaufgaben in Beziehung gesetzt werden.

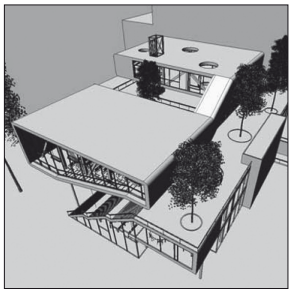
2) Was in generativen Programmen verarbeitet wird, ist nur ein Ausschnitt der vielfältigen Parameter, die bei einem Entwurfsprojekt eine Rolle spielen können.

3) Durch die gedankliche Vermischung der algorithmischen Komplexität mit der Komplexität einer speziellen Entwurfsaufgabe verwischt sich gleichzeitig der Blick auf die Übertragbarkeit der algorithmischen Lösungsstrategien auf andere Entwurfsfelder. Das heißt, die den Algorithmen innewohnende Qualität als allgemeine Problemlöser wird nicht adäquat erkannt, was die Flexibilität generativer Ansätze unnötig limitiert. Gleichzeitig werden ihre Ergebnisse schwerer nachvollziehbar, weil nicht mehr genau erkennbar ist, was nun an einer Lösung algorithmenspezifisch und was entwurfsspezifisch ist.

Auf der persönlichen Suche zu einer adäquaten generativen Entwurfsstrategie hat sich allmählich eine mögliche Methodik ergeben, die Bereiche ‚Entwurf' und ‚Algorithmenentwurf' getrennt voneinander zu entwickeln – durch eine Aufgliederung generativer Prozesses in die Einheiten ‚Programm' und ‚Stil'. Diese beiden Begriffe veranschaulichen beispielhaft die in *Kapitel 3* besprochene Aufteilung in *generativ / projektspezifisch* sowie *algorithmisch / generisch*. Mit ‚Programm' sind hier die Bestandteile gemeint, die räumliche Bezüge, funktionelle und andere Zusammenhänge organisieren, auf einer geometrischen Abstraktionsebene, die unterhalb dem Informationsniveau liegt, das für eine bauliche Umsetzung notwendig wäre. Mit ‚Stil' ist die konkrete, formale Interpretation eines solchen Programms bezeichnet, das sämtliche Bauteile soweit detailliert erzeugt, dass sich daraus die jeweiligen Fertigungsdaten ableiten lassen. Zwischen ‚Programm' und ‚Stil' entstehen gedankliche und tatsächliche Schnittstellen, die es ermöglichen, räumliche Strukturen und ihre Umsetzung entsprechend getrennt zu betrachten und zu bearbeiten. Dieses Konzept soll an zwei Beispielen erläutert werden.

Willems (2006) [10] entwickelt ein generatives Programm, das über die Verknüpfung eines genetischen Algorithmus mit einem *Path-Finding*-Algorithmus in der Lage ist, aus einer zufälligen Anfangskonstellation heraus Bezüge zwischen Räumen mit verschiedenen funktionalen Anforderungen zu entwickeln. Die Positionierung der verschiedenen Raumarten wie Hörsaal, Seminarräume, Toiletten, Zuwegungen, etc., in der schematischen Darstellung [Abb. 04 / oben] farblich unterschieden, wird durch die genetischen Operatoren verändert. Eine ganze Reihe von Fitnessfunktionen definiert und überprüft Bedingungen über Zuwegelängen, Nachbarschaftsbeziehungen zwischen verschiedenen Arten von Räumen, usw. und sorgt dafür, dass sich im Fortlauf des evolutionären Prozesses funktionierende räumliche Organisationsformen einstellen – die Abbildung [Abb. 04 / Mitte] zeigt zwei davon. Diese Ergebnisse markieren eine klare Schnittstelle im generativen Prozess, das ‚Programm' erzeugt einen eigenständigen Entwurfsraum. Die weitere Detaillierung von einem der erzeugten Raumprogramme wurde dann in klassischer Manier, von Hand am *CAD*-System durchgeführt, zunächst grob, [Abb. 04 / Mitte / rechts], dann schließlich datailliert als ein konkreter Umsetzungsvorschlag [Abb. 04 / unten]. Der Algorithmus zur Raumerzeugung und -organisation selbst ist unabhängig davon, er kann in jedem anderen Entwurfskontext Verwendung finden: Die einzelnen Raumgrößen sind definiert über ein skalierbares 3D-Raster, das für jedes beliebige, orthogonale Gebäudevolumen verwendet werden kann. Zwar beschränkt und lenkt die generierte Würfelchenstruktur die Möglichkeiten der weiteren entwerferischen Interpretation, läßt aber gleichzeitig noch einen gewissen Spielraum zum Applizieren verschiedener formaler Stile.





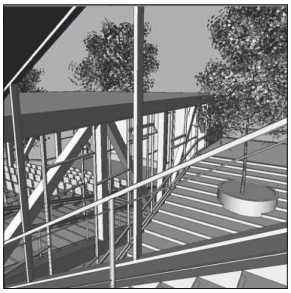
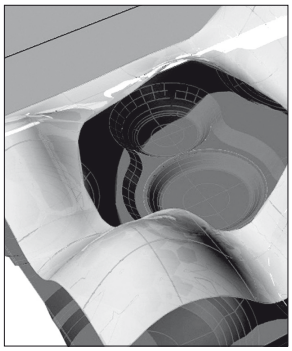
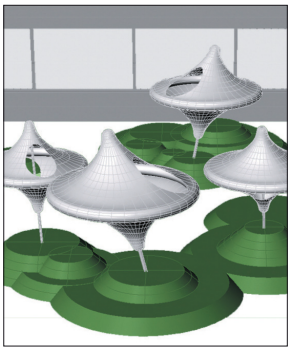
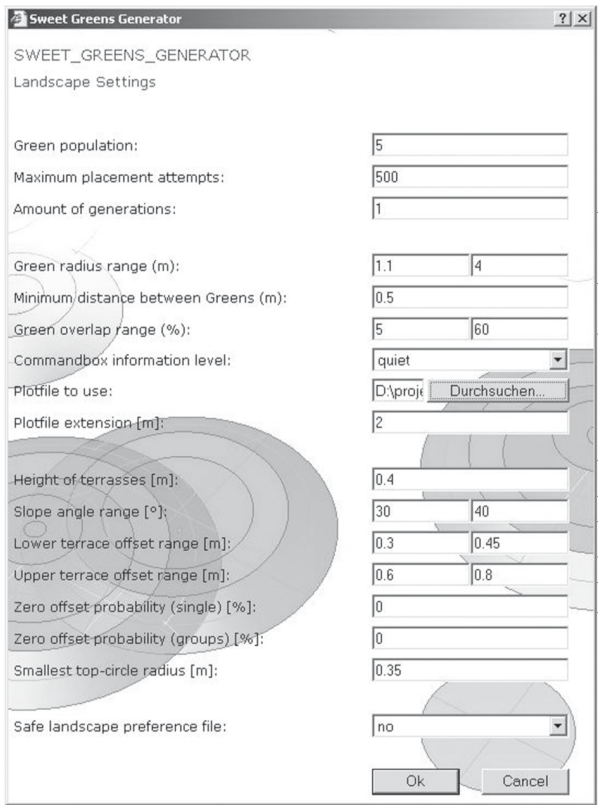


Abb. [04]) Gebäudeentwurf auf Basis eines evolutionär erzeugten Raumprogramms: Grundbausteine (oben), zwei Varianten und ihre grobe Umsetzung (Mitte), detaillierter Ausführungsvorschlag. (Willems, 2006)

Abb. [05]) Sweet-Greens Projekt: Entwurfsidee, Eingabefenster für die Parameter der Hügellandschaft, computergenerierte und tatsächlich realisierte Landschaft. (Schein et al, 2004)







Ein Gegenbeispiel dazu ist das *Sweet-Greens*-Projekt *(Schein et al, 2004)* [11], bei dem durch einen parametrischen Designansatz der Entwurf einer Rasenlandschaft entwickelt und realisiert wurde, die teilweise mit pneumatisch gestützen Membranen überdacht ist. Dieses Projekt begann mit der groben Idee einer Umsetzung [Abb. 05 / oben / links], aus der dann schrittweise ein parametrisches Modell entwickelt wurde. Das Grundprinzip dieses Modells ist, in einen beliebigen Grundriss, der durch eine geschlossene Polylinie definiert wird, Kreise an zufälliger Stelle mit zufälligen Radien zu platzieren. Die Kreise werden dann geprüft, ob sie die definierten Rahmenparameter [Abb. 05 / Mitte] erfüllen – beispielsweise Mindestabstände zu anderen Kreisen, zu Hindernissen im Grundriss wie Bäume, etc. Entspricht ein Kreis allen Parametern, wird er beibehalten, wenn nicht, wird er wieder gelöscht. Dieser Prozess geht solange, bis entweder eine definierte Anzahl Kreise oder Anzahl von Versuchen, Kreise zu platzieren, erreicht ist. Danach wachsen die Kreise in die Höhen, wieder unter Einhaltung bestimmter Rahmenbedingungen. Dann entstehen aus der Geometrie der Hügel Dächer, auch hier wird eine Reihe von Prüfungen durchgeführt (Mindest-, Maximalgrößen, Überschneidungen mit anderen Dächern bzw. sonstigen Hindernissen, usw.). Es entsteht in jedem Durchlauf eine neue Rasenlandschaft. Gezeigt ist die letztlich umgesetzte Variante in virtueller [Abb. 05 / oben / rechts] sowie physischer Form [Abb. 05 / unten].

Im Grunde ist hier ein allgemeines Organisationsprinzip von 2D-Objekten auf verschiedenen Ebenen eines 3D-Raums entwickelt worden. Ausgehend von der Idee des

Kreises, basieren jedoch alle Berechnungen innerhalb des Programms auf der Kreisformel und sind damit für jede weitere Anwendung unbrauchbar oder wären nur mit vergleichsweise großem Aufwand entsprechend zu ändern. Dabei wäre es nicht allzu schwierig gewesen, diese Form der Raumorganisation von vorneweg auf verschiedene Geometrien (z.B. Rechtecke, Polylinienzüge, etc.) auszurichten und damit auch das Applizieren unterschiedlicher formaler Interpretationen (‚Stile') zu erleichtern. Das ‚Programm' wäre wesentlich universeller einsetzbar und losgelöst vom spezifischen Entwurfsgegenstand. Durch das Verweben einer Idee von konkreter Umsetzung mit einem allgemeinen Prinzip zur Organisation räumlicher Beziehungen, ist der *Sweet-Greens*-Generator festgelegt auf das, was er kann: Rasenlandschaften erzeugen in Hülle und Fülle.

4.2 Algorithmisch Denken?

Die Basis algorithmischer Entwurfsmethoden ist die Automation von Routineabläufen. Sie wird möglich durch die hohe Merkfähigkeit (Speicherkapazität) von Computern, die große Geschwindigkeit, mit der sie Designmodelle berechnen und darstellen können (Rechnerleistung) sowie die immens hohe Präzision (wiederherstellende Logik), mit der dies geschieht. Dazu kommt, dass mit solchen Methoden, mit der gleichen Leichtigkeit, mit der identische Designmodelle erzeugt auch eine fast beliebige Zahl ähnlicher Instanzen aus dem selben Lösungsraum generiert werden kann. Auf dem momentanen Entwicklungsstand geschieht dies zwar (noch) überwiegend innerhalb eines recht überschaubaren Grades an Differenzierung – es ist aber trotzdem eine Art von Variations- bzw. Anpassungsfähigkeit gegeben. Dies zusammen ist eine besondere Form der quantitativen Intelligenz, durch die sich neue Gestaltungsspielräume öffnen.

Ein Vergleich zu *Borges (1941)* [12] *„Die Bibliothek von Babel"* liegt hier nahe: Ein Wort in dieser gedachten Bibliothek besteht aus einer zufälligen Auswahl von 25 Zeichen, beliebig aneinandergereiht, ein Satz besteht aus einer zufälligen Auswahl von Wörtern, Kapitel ergeben sich aus zufälligen Sätzen, Bücher aus zufälligen Kapiteln. Damit enthält diese Bibliothek alle Bücher, die bereits geschrieben wurden, aber auch alle noch möglichen Bücher. In einer vergleichbaren Weise enthält der *Computer* die geometrischen Beschreibungen aller bereits realisierten und künftig noch möglichen Entwürfe. Aber im Unterschied zu *Borges* unendlicher Bibliothek, in der es in einem Menschenleben kaum möglich ist, ein sinnvolles Wort geschweige denn einen sinnvollen Satz zu entdecken, schafft die quantitative Intelligenz von Computern ebenfalls einen unendlichen Möglichkeitsraum. Es ist aber gleichzeitig genau wieder diese quantitative Leistungsfähigkeit, mit der sich diese Möglichkeitsräume auch erschließen lassen.

Es sind diese vier Eigenschaften – Merkfähigkeit, Geschwindigkeit, Präzision sowie Variations- und Anpassungsfähigkeit – die so zu denken sind, dass sie in generativen Designprozessen in irgendeiner Form zu einer besonderen gestalterischen Qualität führen: *„It is about using computers to think, as an extension of the brain." (Watanabe, 2002)* [13] Das Problem dabei ist nur, dass die Betrachtung von Design als ein Prozess der variantenreichen Automation nicht gerade der klassischen Vorstellung von Designprozessen entspricht. Auch die zusätzliche Komplexität solcher Prozesse, die dazu von einem hohen Abstraktionsgrad sind, gehen über das gewohnte Designverständnis hinaus. Entsprechend schwierig ist es auch, sich von gedanklichen Vorformatierungen zu lösen und einen adäquaten Umgang mit generativen Methoden zu experimentieren und zu finden. Dass dabei zunächst viel vom Gewohnten in das Neue transportiert wird, ist nur natürlich. ‚Neues Denken' geht nicht auf Knopfdruck, es muss sich erst allmählich, durch das Zusammenspiel mit dem Tun entwickeln. *DeLanda (2002)* [14] benennt, in Anlehnung an *Deleuze*, mit *„populational, intensive and topological thinking"* drei kognitive Stile, die von der Seite des rationalen Verständnisses einen Zugang zu einem veränderten Designhandeln bieten können. Er diskutiert sie im Kontext der genetischen Algorithmen, es sind damit aber gleichzeitig drei Themen angesprochen, die für das gesamte Feld des algorithmischen Designdenkens relevant sind.

4.2.1 Population Thinking – Denken in Lösungsräumen

Die Idee des *population thinking* geht auf die 1930er Jahre zurück, in die Zeit der Synthese von *Darwins* Evolutionstheorie und *Mendel'scher* Vererbungslehre zur modernen Evolutionsbiologie. Damit einher ging die Erkenntnis, dass nicht das Individuum sondern die Population das maßgebliche Element bei der Erzeugung von Formenvielfalt ist *(Mayr, 2001)* [15] – das Einzelne als konkrete Instanz eines Möglichkeitsraums.

Das eben diskutierte *Sweet Greens*-Projekt ist in Populationen, also Lösungsräumen gedacht, in zweifacher Hinsicht. Einmal ist es so konzipiert, dass jede erzeugte Rasenlandschaft einzigartig ist – eine Wiederholung ist auf Grund des enthaltenen Zufallselements von sehr geringer Wahrscheinlichkeit. Zum anderen kann das Programm mit verschiedenen Grundrissen arbeiten, erzeugt Variationen also auch unter verschiedenen Ausgangssituationen. Trotzdem ist es insofern unflexibel, als dass zwar innerhalb des konkreten Projektkontextes, nicht jedoch bei der Entwicklung der zu Grunde liegenden algorithmischen Struktur in Lösungsräumen gedacht wurde. Hier findet sich eine typische entwerferische Denkweise wieder, es wurde nur lösungsorientiert und nicht zusätzlich problemorientiert gearbeitet *(vgl. Lawson, 2006)* [16]. Mit anderen Worten gesagt, stand die Lösung des spezifischen Entwurfsproblems ‚Rasenlandschaft' im Vordergrund, nicht aber der Gedanke, wie hier das Problem einer bestimmten Systematik der räumlichen Organisation grundsätzlich gelöst werden kann. Dieses Entwurfsverhalten korrespondiert wieder mit der Unterscheidung, die im vorigen Kapitel zwischen generativem und algorithmischem Entwerfen getroffen wurde. Das Generative bezeichnet mehr das ‚Hemdsärmelige', ist mehr auf die Lösung einer konkret anliegenden Entwurfsaufgabe fokussiert, das Algorithmische mehr auf die problemorientierte, konzeptionelle Ebene des Algorithmendesigns ausgerichtet. Ein einzelnes generatives Programm repräsentiert eine Population möglicher Lösungen. Jede neue Lösung, die durch die Veränderung von Eingabe- und Verarbeitungsparameter erzeugt wird, ist einzigartig, entspricht aber trotzdem den Randbedingungen, die durch das Programm ausgedrückt sind. Entblättert man ein solches Programm von den Bedeutungszusammenhängen, in denen es verwendet wird, bleiben die Algorithmen in ihrer grundlegenden Konzeption übrig, dynamisierte Diagramme, die abstrakte Bedingungen setzten und logische Beziehungen organisieren. Diese Ebene der Algorithmen entspricht jetzt wieder eine Population möglicher generativer Programme.

Konventionelles Entwerfen arbeitet mit Lösungsräumen, denkt aber in der Kategorie solitärer Lösungen – das Ausloten von Lösungsräumen dient dazu, die einzelnen Elemente einer Lösung zu bestimmen und bis in jedes Detail auszuformulieren. Generatives Entwerfen arbeitet ebenfalls mit Lösungsräumen, seine Ergebnisse sind Lösungsräume – es werden projektbezogene Bedingungen gesetzt und Beziehungsgeflechte geknüpft, aus denen heraus sich eine gewisse Bandbreite an konkreten Lösungen erzeugen läßt. Algorithmisches Entwerfen schließlich arbeitet ebenfalls in Lösungsräumen, denkt jedoch (weitgehend) unabhängig vom konkreten Projektkontext. Es erzeugt abstrakte Lösungsräume, die sich in verschiedenen generativen Programmen zu neuen Entwurfsansätzen rekombinieren lassen.

---

15) „Schließlich, im Rahmen der Synthese der Evolutionsforschung […], schälte sich eine einheitliche Ansicht heraus: „Evolution ist die zeitliche Veränderung in den Eigenschaften der Populationen von Lebewesen." Mit anderen Worten: Die Population ist die sogenannte Einheit der Evolution. Gene, Individuen und biologische Arten (Spezies) sind zwar ebenfalls von Bedeutung, aber der Wandel der Populationen ist das charakteristische Kennzeichen für die Evolution des Lebendigen." (Mayr, 2001)

17) „If a quantity of matter is divided into two equal parts, each part will have the same value of the original and half the value of the extensive properties […] Two extensive properties add up in a simple way, intensive properties do not add up, but rather average." (deLanda, 2002)

4.2.2 Intensive Thinking – Denken in graduellen Veränderungen

Intensive und extensive Größen lassen sich ihrem Verhältnis nach definieren *(vgl. de Landa, 2002)* [17]. Letztere sind zum Beispiel Längen, Flächen oder Volumen, Größen, die Entwerfern vertraut sind und mit denen sie alltäglich umgehen. Das Kennzeichen solcher Größen ist, dass sie sich räumlich unterteilen lassen. Wird eine Strecke halbiert, entsteht zweimal die halbe Strecke. Symbolisiert die Strecke beispielsweise eine reale Knoten-Stab-Verbindung, werden aus einem Stab und zwei Knoten, drei Knoten und zwei Stäbe, werden beide halben Strecken wieder halbiert, entstehen entsprechend fünf Knoten und vier Stäbe, usw. Solche Arten von Verknüpfungen zwischen eindimensionalen Größen wie Anzahlen und Längen, zwei- und dreidimensionalen Größen wie Flächen und Volumen auf geometrischer Ebene sind die Bausteine parametrischer Designmethoden *(vgl. Kapitel 5.2)* und repräsentieren einen großen Teile der praktisch angewandten generativen Designansätze.

Intensive Größen sind nun solche, die bei Teilung unverändert bleiben bzw. sich nicht linear verändern. Werden 10 Liter Wasser mit 30° Temperatur in zweimal 5 Liter unterteilt, bleibt die Temperatur davon unberührt bei 30°, wird eine freistehende Säule vertikal geteilt, ändert sich nicht die Kraft, die durch die Säule abgeleitet werden muss. Wird Kraft (N) mit Fläche (mm^2) in Bezug gesetzt, ist dies die Spannung (N / mm^2) eine Bauteils. Verändert sich bei gleichbleibender Kraft der Querschnitt eines Bauteils, ändert sich auch der Spannungswert. Die maximal zulässige Spannung eines bestimmten Materials gibt somit Auskunft über die maximale Belastbarkeit eines Bauteils. Dies ist der Standardbereich der Formanalyse und -optimierung. Wird nun die maximal zulässige Spannung als intensive Größe begriffen, wird die Kraft zum formerzeugenden Parameter – die Änderung der Kraft bringt die Form hervor, nicht die erzeugte Form wird darauf geprüft, ob sie einer Kraft standhält.

Solche gedanklichen Schritte markieren sehr wichtige Wege hin zu einer größeren Relevanz generativer Designmethoden, die sie über die übliche Ebene der Geometrie hinaus tragen können. Diese Schritte scheinen klein zu sein, geradezu selbstverständlich, müssen aber doch gewaltige Hürden nehmen. Hier läuft immerhin eine seit langem etablierte Trennlinie in den entwerferischen Berufen, zwischen Designern und Konstrukteuren, Architekten und Ingenieuren. *Reiser* und *Umemoto (2006)* [18], die entlang dieser Trennlinie arbeiten, formulieren dies in sehr sympathischer Weise: *„Our treatment deals in a qualitative fashion with issues that ultimately require quantitative treatment: The province of our collaborators, engineers."* Physikalische Parameter als selbstverständliche Elemente eines generativen, geometrische Instanzen definierenden Prinzips zu begreifen, bedeutet für Entwerfer, die sich solche Ansätze entwickeln, auch ein größeres Maß an Verantwortung – zum Beispiel der Weg vom Architekten zum *„digital masterbuilder" (Kolarevic, 2003)* [19], der durch solche Prozesse in die Domänen von Bauingenieurwesen, Statik und Bauausführung eingreift. Diese Wunschvorstellung des digitalen Baumeisters könnten natürlich auch umgekehrt die Ingenieure und Konstrukteure für sich reklamieren. Weiterhin ist es auch geradezu der Kern der entwerferischen Disziplinen, dass hier die Hoheit über die Bestimmung von Formen liegt. Die Vorstellung, nur noch Moderator eines formerzeugenden Prozesses zu sein, der sich zwar im Groben, nicht mehr aber in jedem Detail kontrollieren läßt, wird das Selbstverständnis vieler Entwerfer strapazieren.

Der *Groningen-Twister* bietet ein gutes Beispiel für die Akzeptanz eines gewissen entwerferischen Kontrollverlusts: Bei diesem Projekt zur Neugestaltung des *Groningen Stadsbalkon, (KCAP/ARUP, 2003)* einem Platz vor dem Groninger Bahnhof, sollte eine massive Betonplatte von einem ‚Wald von Säulen' getragen werden – die gestalterische Leitidee Abb. 06 / oben. Bei der Plazierung der Säulen ergab sich folgendes Problem: *„There were too many degrees of freedom (column location, tilt angle, column size) and at the same time too many restricting rules (holes, incisions and paths to avoid, bearing capacities, optimum column distances) influencing each other and preventing to design a structurally and aesthetically working solution in reasonable time." (Scheurer, 2003)* [20] Die Lö-

sung war ein interaktives, javabasiertes Tool, bei der die einzelnen Säulen ähnlich wie Partikel eines Schwarms organisiert sind, die sich nach bestimmten Regeln zueinander verhalten, was in einem gewissen Grad durch das Verändern von Rahmenparametern beeinflusst werden kann Abb. 06 / unten. Die statischen Verhältnisse sind so nicht länger Größen, die durch gezielte, detaillierte Einflussnahme des Entwerfers manipuliert, verändert und kontrolliert werden. Sie werden zu einem akzeptierten, stabilen Element, in dessen Rahmen sich ein Prozess der Selbstorganisation vollzieht, der nur noch moderiert werden kann.

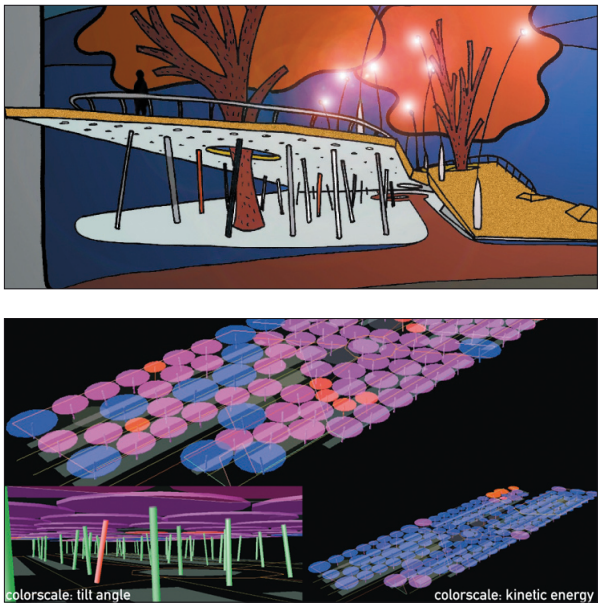


Abb. [06]) Groningen Stadsbalkon: Gestalterischer Leitgedanke (oben) (KCAP, 2003) und generatives Tool als Realisierungshilfe. (ARUP; Scheurer, 2003)

4.2.3 Topological Thinking – Denken in gleichbleibenden Eigenschaften

Topologische Eigenschaften sind solche, die sich unter Transformationen nicht verändern. Topologisches Denken bedeutet im Kontext generativer Verfahren, weniger in metrischen Dimensionen zu denken, sondern mehr die Beziehungen, durch die verschiedene Größen verbunden sind, als Mittel formerzeugender Prozesse zu nutzen. In konventionellen Designprozessen spielt der Umgang mit Eigenschaften, die unter Transformation unverändert bleiben, hauptsächlich als Kriterium der ökonomischen Rationalität eine Rolle: Je dichter das Bedingungsgeflecht eines Entwurfs und je weiter fortgeschritten ein Entwurfsprozess, umso mehr verlegt man sich darauf, eher Änderungen vorzunehmen, deren Auswirkungen begrenzt sind. Betrachtet man zum Beispiel einen klassischen Stuhl mit Sitzschale aus 3D-Formholz mit einem daran verschraubten Stahlgestell, so ist es völlig problemlos, die Schale etwa im Rückenbereich etwas zu modifizieren – das Stahlgestell verhält sich invariant gegenüber der Transformation der Sitzschale. Kommt man aber beispielsweise auf die Idee, die Befestigung des Stahlgestells in die Formholzschale zu integrieren, müssen sich sowohl Schale als auch Gestell ändern – als Folge wäre der ganze Entwurf zu revidieren Abb. 07.

Abb. [07]) Ameise 3100 als Dreibein- und Vierbein-Version, sowie ein Stuhl aus der 7er Serie (Jacobsen). Einmal verhält sich das Gestell invariant zu Änderungen der Sitzschalenform, einmal die Schalenform zum Gestell. Für die generative Transformation eines Entwurfs wie Taino (Gebert) müßten Schale und Gestell als eine topologische Einheit definiert werden.

Im generativen Kontext sind Eigenschaften, die bei Transformationsprozessen unverändert bleiben, ein wichtiges gestalterisches Element. Wie bereits besprochen, ist es

zwar nicht unmöglich, generative Designmodelle zu entwickeln, bei denen die Änderung einer Eigenschaft alle anderen Eigenschaften (strukturelle, funktionelle, etc.) eines Designmodells ebenfalls ändert, es ist aber auf jeden Fall eine äußerst anspruchsvolle Aufgabe und aus der Sicht der Programmierökonomie fragwürdig. Die Arbeit von *Werner (2003)* [21] gibt ein einfaches Beispiel, wie topologische Eigenschaften zur Basis eines Entwurfsraums werden können. Ausgehend von einer industriell gefertigten Sitzschale entwickelt er parametrische Variationen, die auf dem Beschnitt dieser Urschale basieren [Abb. 08]. Dieses formerzeugende Prinzip bedingt weder Änderungen für das Ausgangselement, noch für entsprechend konzipierte Stuhlgestelle. Ein weiteres Beispiel für das Spiel mit topologischen Eigenschaften ist die schon eingangs gezeigte Arbeit von *Menges (2005)* [22], der mit in sich verdrehten, abwickelbaren Bändern eine *NURBS*-Fläche interpretiert. Diese Konstruktion ist direkt abgeleitet aus dem uvn-Parameterraum einer solchen Fläche. Sofern bestimmte Sonderfälle ausgeschlossen werden, ist das Verhalten dieses Raums von der Form einer Fläche vollkommen unabhängig. Egal, wie die Ursprungsfläche verändert wird, die generative Logik ist in jedem Fall anwendbar und sorgt dafür, dass jedes einzelne Band die Veränderung durch eine Anpassung seiner Abmessungen nachvollzieht [Abb. 09].

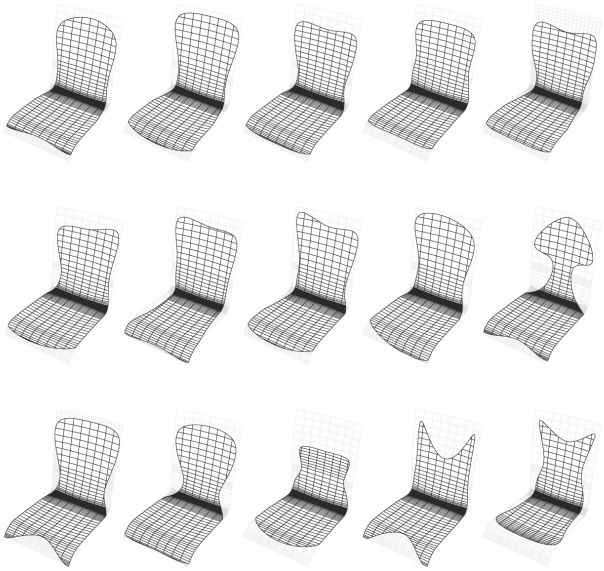


Abb. [08]) Parametrische Variationen einer Sitzschale. (Werner, 2003)

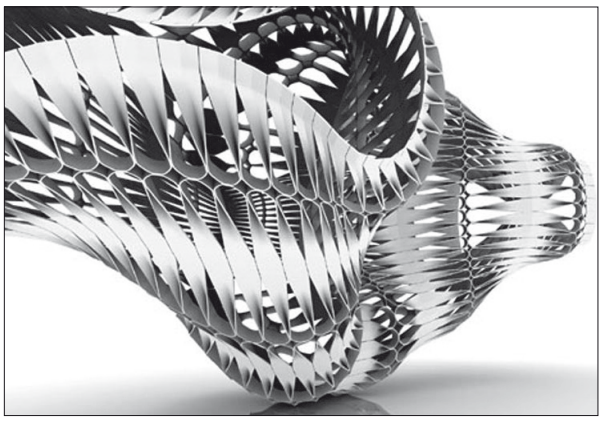


Abb. [09]) uvn-Parameterraum einer NURBS-Fläche als topologischer Raum. (Menges, 2005)

4.2.4 Akzeptanz von Constraints

Als abschließende Bemerkungen zum Bereich des algorithmischen Denkens, noch einmal eine Anmerkung zum Umgang mit den Randbedingungen. Man kann davon ausgehen, dass Randbedingungen, die von einem Entwerfer klaglos akzeptiert werden, entweder solche sind, die in einem Entwurfsprozess nicht als hinderlich empfunden werden oder gar solche, die sich als fruchtbar erweisen, entweder, weil sie sich als nützliche *Design Driver* verwenden lassen, oder zumindest den Nährboden für fruchtbare Konflikte bieten. Wenn ein Entwerfer Randbedinungen lautstark beklagt, sie aber trotzdem akzeptiert, ist dies sehr wahrscheinlich eine Bedingung der Kategorie *„Sachzwänge – Ausreden für Entscheidungsmüde"*, wie *Rittel (1976)* [23] sagt. Ansonsten werden *Constraints* formuliert und verändert, bis sich darüber ein geeigneter Bewertungsfilter ergibt, den eine Entwurfslösung insgesamt passieren kann. Beim algorithmischen Entwerfen verhält sich dies ein wenig anders. Es geht hier darum, Bedingungen vor allem dann in Frage zu stellen, wenn sie sich aus einem traditionellen Verständnis heraus stellen, wie und mit welchen Mitteln Design zu betreiben ist. Dagegen macht es wenig Sinn, während der Entwicklung eines generativen Programms ständig die gesetzten Randbedingungen zu hinterfragen und neu zu definieren, da hier eine eindeutige Kausalkette (das Programm) aufgebaut werden muss, die zwischen Bedingungen und Zielvor-

stellungen vermittelt *(vgl. Kapitel 3.1)*. Dies ist nun kein Rückfall in die systemtheoretischen Ansätze der ersten Generation, nach dem Motto: ‚Definiere die Mission, beschreibe das Problem, dann löse es'. Es ist nur so, dass Programmierung eben kein besonders geeignetes Mittel ist, um mit ständig wechselnden Bedingungen umzugehen. Es gilt hier als Gestaltungsprämisse, was *Fuller (1932)* [24] formuliert hat: *„Don't fight forces, use them."* Je schneller Bedingungen und Ziele festgelegt werden, desto effektiver kann auch die Entwicklung eines generativen Programms voran schreiten, umso schneller ist man auch in der Lage die Ergebnisse zu bewerten – um dann möglicherweise wieder neue Bedingungen und Ziele zu definieren, das Programm abzuändern oder neu zu schreiben. Dass die gestalterische Freiheit dabei Schaden nehmen könnte, ist auch nicht zu befürchten. Kreativität kann sich in eng begrenzten Lösungsräumen genauso austoben wie in weiter gefassten *(vgl. Lawson, 2006)* [25]. Zudem spielen frühe, zuerst realisierte Vorstellungen, wie ein Designproblem zu lösen sei, ohnehin eine wichtige Rolle als *Primary Generators*, als Gedankengebäude, an dem sich die folgenden Designentscheidungen orientieren.

4.3 Versioning durch algorithmische Entwurfsräume

Das Bedeutungsspektrum des Begriffs ‚*Versioning*' ist vielfältig. Es spannt von einfacher Variantenbildung über die Erzeugung von Varietät bis hin zum Ausdruck einer speziellen Entwurfshaltung, die Design auf der Ebene eines informationsverarbeitenden Prozesses begreift. Ein gewisses Anpassungs- bzw. Variationsvermögen ist allen generativen Designansätzen eigen, bedingt durch die parametrische Natur von Computerprogrammen. Gerade diese Eigenschaft ist für das Entwerfen äußerst reizvoll, aber entsprechend der vielfältigen Interpretation des *Versioning*, fallen auch die Erwartungen diesbezüglich sehr unterschiedlich aus.

4.3.1 Versionen des *Versioning*

1) Anpassung an sich verändernde Kontextbedingungen. Eine erste Ebene des *Versioning* ist, einen Entwurf während seiner Entwicklung in einem parametrischen Designmodell abzubilden, um ihn dadurch sich verändernden Kontextbedingungen schneller und leichter anpassen zu können. Diese Art, die Variationsfähigkeit parametrischer Designmodelle auszunutzen, ist kritisch zu betrachten: Je komplexer ein solches Modell wird, desto schwieriger wird es auch, seine Struktur so anzulegen, dass durch die Änderungen einzelner, vielfältig verknüpfter Bedingungen, das komplette Designmodell fehler- und konfliktfrei aktualisiert werden kann. Betreffen die Änderungen gar Bedingungen, die gar nicht im Modell ausgedrückt sind, kann die gesamte generative Entwurfsstrategie zusammenbrechen. Unter diesem Blickwinkel ist auch die Forderung *Kolarevics (2003)* [26] mit einem Fragezeichen zu versehen, nach der die Architektur durchgängige digitale Designketten nach dem Vorbild der Schiffs- oder Flugzeugbauindustrie anstreben sollte. Solche hochkomplexen Designprozesse sind nicht nur fehleranfällig, sie sind gleichzeitig auch Entscheidungskorsette. Dies hat sich beispielsweise am oben kurz skizzierten *Sweet-Greens*-Projekt sehr deutlich gezeigt, bei dem das parametrische Designmodell zunächst als willkommener Filter zur Beschleunigung von Entwurfsentscheidungen gedient hat, später aber die Bereitschaft, alternative Konzepte zu erproben, mit dem Fortschritt der Programmierung rapide abgenommen hat.

2) Variantenerzeugung als Ziel eines Entwurfsprozesses. Eine weitere Ebene des *Versioning* ist die Ausnutzung der Variationsfähigkeit eines generativen Programms nach Beendigung des eigentlichen Entwurfsprozesses. Das Stichwort zu diesem Anwendungskontext ist *Mass Customization* – also kundenindividuelle Massenproduktion. *Mass Customization* ist ein Konzept der Wirtschaftswissenschaft aus den 80er Jahren. Die Entwicklung von lokalen Märkten hin zu Massenmärkten, segmentierten Märkten und Nischenmärkten führte zu der Idee des individuellen Marktes – ein maßgeschneidertes Produkt für jeden einzelnen Kunden, eine Vorstellung, die über das Industriedesign mittlerweile auch Einzug in die Architektur gehalten hat. Eine wesentliche Hürde für dieses Konzept – entsprechend flexible Produktionsverfahren – ist durch *CNC*-gesteuerte Maschinen

bzw. *Rapid-Prototyping*-Verfahren zumindest im Prinzip genommen [Abb. 10]. Weiterhin stellt sich noch die Frage, wie Entwurfsprozesse der *Mass Customization* organisiert sein können, die im Grunde den Lösungsraum zum Produkt machen – die Domäne der generativen Designansätze. Üblicherweise wird bei Entwürfen, die auf individuelle Produkte abzielen, das Maß an Individualität bereits von den Entwerfern definiert [Abb. 11].

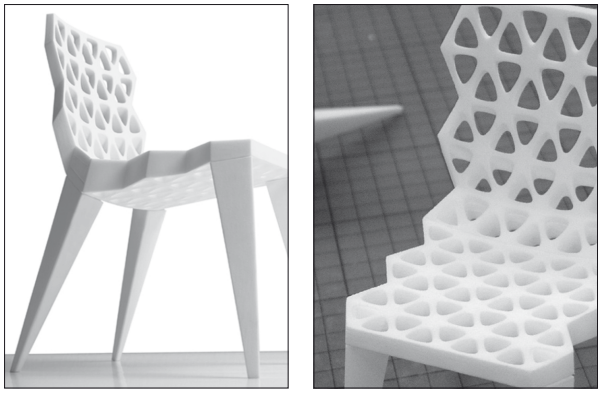


Abb. [10]) Ein Entwurf, ein Fertigungsverfahren zwei unterschiedliche Größendimensionen. Der Sinterstuhl im Original und im Maßstab 1:5. (Vogt und Weizenegger, 2002)

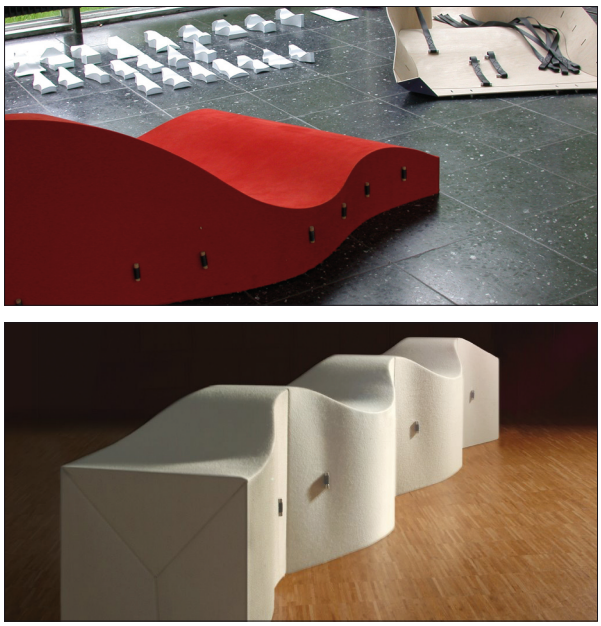


Abb. [11]) Wieviel Differenz brauchen Produkte, um als individuell angesehen zu werden? Parametrischer Entwurfsraum von Faltmöbeln, 1:10 Papiermodelle sowie drei realisierte Prototypen. (Schein, 2002)

Das Sinnbild schlechthin für die Schwelle zur Individualisierung der Märkte ist die *Swatch*-Uhr der Schweizer Firma *SMH*, die jahrelang, pünktlich zum Frühjahr, in neun Produktlinien organisiert, eine mittlerweile unüberschaubare Zahl an Armbanduhren-Varianten auf den Markt gebracht hat. Jeder Kunde wählt die ihm passende Uhr, allerdings ohne dass er dabei Einfluss auf deren Gestaltung nehmen kann.

Einen Schritt weiter Richtung *Mass Customization* gehen die etablierten kombinatorischen Konzepte der Automobilindustrie, bei welchen der Kunde bis kurz vor Auslieferung eine Vielzahl von Ausstattungsmerkmalen von der Lackierung über Sitzbezüge bis hin zum Bordcomputer selbst bestimmen kann. Die Neuauflage des *Rover Mini* ist theoretisch in rund 10 Millionen Varianten lieferbar, mehr als wohl jemals verkauft werden. Eine etwas überschaubarere Anzahl an Möglichkeiten bietet eine Sofaentwurf von *Benz* und *Reuter (2003)*, der sein Programm im Namen trägt: *„33.372 Varianten bietet das Modell EGO laut Pressemitteilung. […] Individuell zusammensetzen kann der Kunde das Sofa aus: Drei verschiedenen Grundtypen, acht Armlehnen, drei Rückenhöhen, fünf Sitzbreiten, drei Sitztiefen, drei Sitzhöhen, drei Sitzkomfortstufen, zwölf Füßen und 200 Ledern bzw. 600 Stoffen. Hinzu kommen noch Zusatzteile wie verstellbare Rückenkissen oder klappbare Seitenteile." (Werner, 2003)* [27]

Auf diesem Niveau der kundenindividuellen Massenproduktion sind generative Programme eine interessante Perspektive. Sie sind extrem leistungsfähig hinsichtlich der Anzahl an Varianten, die sie erzeugen können, sowie der Geschwindigkeit, in der dies geschieht. Kombinatorische Ansätze fügen sich sehr gut in eine Programmierlogik, die Einbindung in digitale Prozessketten vom *Interface* über den Kunden bis hin zur Produktion ist ebenfalls problemlos machbar. Vielleicht böte der Umgang mit einem anonymen Programm sogar einen besseren Rahmen, die eigene Individualität auszuleben, als der Umweg über einen menschlichen Kundenberater – das *Avatar* besiedelte 3D-Web, auch *Second Life* genannt, liefert hier Vorbilder. Grundsätzliche Probleme in der Idee der *Mass Customization* sind aber

auch durch noch so ausgeklügelte generative Prozessketten nicht lösbar, vor allem nicht dasjenige, das im eigentlichen Anspruch dieses Konzepts begründet liegt: Der Kunde sollte mindestens eine von mehreren Gestaltungseigenschaften ohne Vorgabe des Produzenten frei wählen können. Dieser Anspruch ist, wie *Piller*, einer der führenden Köpfe in diesem Gebiet einräumt, selten erfüllt: Bei vielen einschlägigen Angeboten handelt es sich eher um subtile Formen der Marktforschung *(vgl. Etscheit, 2002)* [28]. Bei echter *Mass Customization* stellt sich also die Frage, in welcher Weise ein Kunde an der Entwicklung (s)eines Produkts mitwirken kann, um über die übliche Baukastenlogik hinaus zu gehen. Bei den generativen Programmen würde dies bedeuten, dass er in irgendeiner Form an der Entwicklung der Programme beteiligt sein müsste.

3) Übertragung und Variation von formerzeugenden Konzepten. Die Variationsfähigkeit generativer Programme hat weiterhin eine Bedeutung, die quer zu verschiedenen Entwurfsprojekten liegt. Hier kommt die Eigenschaft solcher Programme als Speicher formerzeugender Prinzipien zum Tragen, wie in *Kapitel 2.3* besprochen. Diese Übertragung entwerferischer Ideen auf andere Kontexte kann direkt stattfinden, indem ein bereits entwickeltes generatives Programm für eine neue Entwurfsaufgabe eingesetzt wird. Dies ist aber nur dann möglich, wenn diese neue Aufgabe die selben Ausgangsbedingungen bietet wie die, für welche das Programm ursprünglich entwickelt wurde – und auch nur dann, wenn man als Entwerfer bereit ist, wieder die selben Bedingungen als gegeben zu akzeptieren. Dann können auch einzelne Programmbausteine für ein neues Projekt neu zusammengesetzt werden, z.B. entlang einer Strategie, wie unter der Überschrift ‚Programm und Stil' beschrieben. Dieses Vorgehen ist aber auch unter einer immer feiner werdenden Aufteilung möglich, die bis zu der Ebene reicht, wo algorithmische Konzeptionen zu einem neuen Bedeutungskontext zusammengesetzt werden: Das generative Programm selbst wird zum Gegenstand des *Versioning*.

4) Design als informationsverarbeitender Prozess. Neben diesen eher praktischen Aspekten ist das *Versioning* auch Ausdruck einer Betrachtungsweise, die Design als einen Prozess der Informationsverarbeitung begreift: Entwurfsmodelle als generative Systeme, die in einem Prozess der Informationsanreicherung entwickelt werden und die selbst durch Verarbeitung von Information ihren Zustand ändern: *„Versioning relies on the use of recombinant geometries that allow external influences to affect a system without loosing the precision of numerical control or the ability to translate these geometries using available construction technology. It advocates the use of vector-based information over pixel-based simulation and representation." (Speaks, 2002)* [29] Eine ganz ähnliche Haltung findet sich bei *Lynn (1999)* [30]: *„It is important for any parameter-based design that there be both, the unfolding of an internal system and the infolding of contextual information."*

Zu solchen Vorstellungen von Design sind einige Anmerkungen zu machen. Zunächst wird hier ‚Informationsverarbeitung' nicht als ein statisches Moment begriffen, bei dem ein Computerprogramm Designmodelle erzeugt und organisiert. Generative Programme entstehen allmählich, im Dialog zwischen dem Entwerfer und dem Modell ‚Code'. Sowohl das, was sich als internes System ausbildet, als auch das, was als Kontextinformation vom Programm verarbeitet wird und auf welche Weise dies geschieht, fällt in die Entscheidungshoheit des Entwerfers und entwickelt sich während der Programmierung. Weiterhin bedeutet die Idee der Informationsverarbeitung nicht, dass sie die pragmatische Dimension entwerferischer Tätigkeit negiert – sie eröffnet lediglich eine weitere Betrachtungs- und Bearbeitungebene für Designaufgaben, die sich aus einem konkreten Bedeutungszusammenhang lösen kann. Und schließlich meint Informationsverarbeitung, auch wenn sie in einem digitalen Modell abgebildet wird, keine Ausschließung des Materiellen. Diesen Aspekt wird auch von *Speaks* betont, wenn er von vektorbasierter anstatt pixelbasierter Information spricht. Er begegnet damit einer früh aufgekommenen Kritik am generativen (Architektur-)Entwurf als unbaubare, virtuelle Spielerei, bei der das Ergebnis lediglich auf ein 2D-Bild reduziert und nicht mehr baulich umsetzbar ist. Dies ist natürlich nur dann zu kritisieren, wenn die virtuelle Architektur auch den Anspruch erhebt, physisch umsetzbar zu

sein und nicht, wenn sie sich als Experiment oder ausdrücklich virtuelle Architektur versteht, wie beisielsweise bei die *Sound-Sculptures* von *dECOI* Abb. [12], Entwürfe von hoher sinnliche Qualität.

Abb. [12]) Paramorph II von dECOi: „This we began distorting through sound-modelling (entirely inexpertly) but giving birth to series and series of shell-forms which we began immediately to imagine as a sort of giant aural canal a listening device receptive to the passage of people and host to endlessly evolving sound-sculpture created by bodies in transit floated through the echoic space." (Goulthorpe, 2002) [30a]

4.3.2 Überraschungen und Sensationen

Die Freiheitsgrade, die in einem generativen Programm stecken können, führen Titel wie ‚Parametrik', ‚stochastisches Element', ‚Selbstorganisation' oder ‚Emergenz' – in Abhängigkeit von den verwendeten Algorithmentypen. Angesichts der vielen Möglichkeiten, diese Freiheitsgrade anzuwenden sowie ihrer unterschiedlichen Qualitäten, ist es auch nicht verwunderlich, dass viele Entwerfer eine ganz eigene Vorstellung entwickeln, was mit dieser Variationsfähigkeit erreicht werden kann.

Werner (2005) [31] ist beispielsweise der Ansicht, dass sich auch mit Hilfe einfacher parametrischer Designansätze wirklich überraschende Ergebnisse erzielen lassen. Überraschung braucht Differenz zum Gewohnten und diese sieht er einmal in der Möglichkeit der ausgiebigen systematischen Durchforstung von Lösungsräumen, die ein außergewöhnlich breites Spektrum an Entwurfsvarianten erzeugt – darunter auch solche, die der Entwerfer selbst nicht entwickelt hätte. Die gleiche Art von überraschenden Ergebnissen sieht er durch eine andere Quelle möglich, dem gezielten Einsatz von zufällig definierten Entwurfsparametern. Für *Soddu (2002)* [32] dagegegen spielt das Zufallselement kaum eine Rolle. Er sieht die Möglichkeit, unvorhersehbare Ergebnisse zu erzielen, darin begründet, dass in einem Programm die vielfältigen Interaktion unterschiedlicher Ebenen architektonischer und städtebaulicher Einflußgrößen abgebildet werden und sich über die Zeit entwickeln, aber durch strenge Regeln formal interpretiert werden. Es entstehen *„different results but belonging to the same identifiable architectural concept and town idea." Willems (2006)* [33] sieht im Zusammenhang mit seiner Arbeit zur Raumorganisation mit Hilfe von genetischen Algorithmen ebenfalls, dass mit generativen Programmen Ergebnisse zu erzielen sind, die ein Entwerfer so nicht hervorgebracht hätte. Er bezieht sich dabei aber nicht nur auf eine gewisse Unschärfe innerhalb generativer Programme. Er sieht das Potential für ungewöhnliche Lösungen auch darin, dass generative Entwurfsstrategien andere als die gewohnten Denkprozesse anstoßen und befördern. *DeLanda (2002)* [34] schließlich, ebenfalls bezogen auf die Evolution als Entwurfsmetapher, sieht hier gar die Möglichkeit, Ergebnisse zu erzielen, die nicht nur überraschen sondern gar schockierend sind – generative Prozesse als Multiplikatoren menschlicher Vorstellungskraft.

Solche Einschätzungen lassen sich im Detail hinterfragen, sie sind aus der Perspektive der einzelnen Autoren, ihrem individuellen Zugang zum Themenfeld des generativen Entwerfens nachvollziehbar. Sie – und viele andere Protagonisten im Feld der generativen Entwurfsmethoden, die sich jeweils wieder ihren eigenen Zugang zum Thema entwickelt haben – sind sich alle im Wesentlichen einig darüber, dass im generativen Entwerfen ein Potential liegt, das über das bisherige Verständnis von Entwerfen hinausgeht. Es ist auch klar, dass eines der wesentlichen Elemente darin besteht, die quantitative Leistungsfähigkeit von Computern adäquat einzusetzen und damit Lösungsräume – also Differenz – zu erzeugen. Was aber oft nicht beantwortet wird oder im Unklaren bleibt, ist die Frage, wo der eigentliche Schlüssel zu einer entwerferischen Qualität liegt, die über das Bestehende hinausgeht. Eine Perspektive hierzu:

4.3.3 Variation (Selbstähnlichkeit) und Varietät (Differenz)

Reiser und Umemoto (2006) [35] legen mit ihrem *„Atlas of Novel Tectonics"* eine vielschichtige und umfassende Betrachtung des Themenkomplexes ‚algorithmisches Entwerfen' vor und widmen sich auf einigen Seiten auch dem Thema der ganz besonderen, eigenen Qualität von Differenz, die solches Entwerfen hervorbringen kann. Diese Ideen sind in die folgenden Betrachtungen eingeflossen.

Auf der Ebene der Semantik ist die Frage nach Differenz zunächst eine Frage von Übereinkunft und damit Definition. Ein Stuhl ohne Rückenlehne ist keiner, sondern ein Hocker. Ein niedriger, breit ausladender, weich und opulent gepolsterter Stuhl ist ebenfalls kein solcher, sondern ein Sessel. Solange solche Regeln klar anwendbar sind, ist auch eine eindeutige Zuordnung möglich. An diesem Beispiel (Hocker-Stuhl-Tisch) zeigt *Wells* [36], dass solche Unterscheidungen nicht diskret sondern kontinuierlich sind und gerade in den Übergangsbereichen per Übereinkunft festgelegten Differenzen dahinschmelzen und immer mehr zur Ähnlichkeit werden. Während aber einerseits Ähnlichkeiten durch die Auflösung diskreter Unterscheidungsmerkmale erzeugt werden, ist es andererseits auch so, dass Ähnlichkeit Differenz hervorbringen, genauso wie aus Differenz Ähnlichkeit entstehen kann. *Deleuze* und *Guattari* verdeutlichen dies am Beispiel von Windhund, Ochse und Pferd. Obwohl ein Rennpferd der gleichen Spezies angehört wie ein Arbeitspferd, das vor den Pflug gespannt wird, ist nicht das Pferd dem Pferde ähnlich, sondern das Rennpferd liegt näher beim Windhund, das Arbeitspferd liegt näher beim pflügenden Ochsen. Eine Lektion für das algorithmische Entwerfen daraus ist, einen etablierten, an diskreten Unterscheidungen orientierten Umgang mit Organisationsprinzipien und formalen Interpretationen zu hinterfragen. Es ist ohnehin schon diskutiert worden *(vgl. Kapitel 3.2.4)*, dass diese Art des Entwurfsdenkens nicht besonders gut mit den Eigenheiten algorithmischen Entwerfens in Einklang zu bringen ist: Die vielen Bauteile eines Akkuschraubers etwa unterliegen unterschiedlichen funktionalen, strukturellen, ergonomischen, technischen etc. Anforderungen. Sie können zwar innerhalb eines parametrischen Modells beschrieben werden, wobei einzelne Komponenten austauschbar und in gewissen Grenzen auch transformierbar und unterteilbar sind [Abb 13]. Alle wesentlichen Gestaltungsparameter sind aber trotzdem festgelegt, Innovation ist auf diesem Weg schwer möglich.

Abb. [13]) Diskrete Unterscheidung einzelner Funktionsbereiche und kontinuierliche Übergänge durch das Prinzip der Selbstähnlichkeit.

Anders als diese konventionelle Entwurfslogik, die auf einem diskreten Baukastensystem basiert, sind die formerzeugenden Zusammenhänge in natürlichen Prozessen. Ein Baum z.B. ist in sich selbst ähnlich, von den Wurzeln über den Stamm zu den Ästen und Zweigen bis hin zur Blattstruktur. Trotzdem ändern sich zum Teil die Funktionen, von Nahrungsaufnahme und Verankerung, zu Nahrungstransport und Tragwerk, zu Tragwerk und einer Ausrichtung der Blät-

36) „Denken Sie an Stühle mit Armlehnen, an Lehnsessel, an Eßzimmerstühle, an Stühle die in Bänke übergehen, an Stühle die die Grenze überschreiten und zu Sofas werden, denken sie an Zahnarztstühle, Thronsessel, Opernsitze, an Sitze aller Art, an jene wundersamen pilzähnlichen Gewächse, die den Fußboden in den Kunst- und Kunsthandwerkaustellungen verbauen, und sie werden wahrnehmen, was für ein loses Bündel dieser einfache, unmittelbare Begriff in Wirklichkeit ist. Mit der Unterstützung eines intelligenten Tischlers würde ich mir die Mühe machen, jede Definition von Stuhl oder Stuhlhaftigkeit, die Sie mir geben, zu widerlegen." (Wells)

ter, die größtmögliche Lichtausbeute ermöglicht, die selbst wieder Tragwerk sind und Nahrung aufnehmen. Bei solchen Organisationsformen verliert das einzelne Teil seine Bedeutung und löst sich in der Gesamtstruktur auf [Abb. 13]. Es ist nicht mehr so wichtig, wie sich ein Teil zum Nächsten, zu einer Gruppe von Teilen oder der gesamten Struktur verhält, sondern wie sich die Teile in ihrer Gesamtheit verhalten – *„part to whole - relationships"*, und *„whole to whole - relationships"*, nennen *Reiser* und *Umemote (2006)* [37] diese beiden unterschiedlichen Arten von Beziehungen.

Während in gängigen Designverständnissen die Erzeugung von Varietät (Differenz) als Kern des Entwerfens gesehen wird, ist es beim algorithmisch geprägten Entwerfen eher die Zuwendung zur Variation (Selbst-Ähnlichkeit), die zu eigenständiger gestalterischer Qualität führen kann. Das Prinzip von Selbstähnlichkeit basiert auf Mustern, aber es sind keine Muster aus identischen Elementen, sondern solche, die leicht voneinander abweichen. Zur Darstellung solcher Muster ist die quantitative Intelligenz von *Computer* sehr gut geeignet. Diese Muster können informiert werden, also durch äußere Einflüsse geformt werden, etwa so, wie der Wind Strukturen im Sand erzeugt. Sie können aber gleichzeitig ebenso selbst informieren [Abb. 14].

Abb. [14]) Informierte und informierende Muster in der Natur: Dünen- und Fjällstrukturen.

Beispielsweise hat die Eiszeit die Fjälllandschaften Skandinaviens geschaffen. Diese besonderen Strukturen geben in Zusammenhang mit anderen Faktoren die Bedingungen vor, nach denen sich Flora und Fauna entwickeln können. Seen und Flüsse in den Niederungen gehen über in Sumpf- und Moorlandschaften, in Tannen- dann Birkenwälder, in Zonen mit Krüppelkiefern bis auf den Ebenen der Fjälls schließlich nur noch Flechten und Moose gedeihen können. Alle diese Bereiche markieren Biotope verschiedener Qualität, sind aber nicht diskret voneinander getrennt, sonder gehen kontinuierlich ineinander über.

Man kann sich daran anknüpfend Entwürfe vorstellen und mit Hilfe algorithmischer Entwurfsmethoden abbilden und umsetzen, bei denen etwa ein Muster verschiedener Dichte kontinuierlich übergeht von Fundament zu Tragwerk zu Lichtmodulation zu Öffnung und zurück. Es ist deutlich, dass sich hier der Begriff der Variation nicht mehr nur auf die Veränderung eines parametrischen Entwurfsmodells entlang der Zeit bezieht und auch nicht auf die Variationen zwischen Produkten im Sinne einer *Mass Customization,* wie etwa bei einigen der *FutureFactories* - Produkte [Abb. 15]. Diese beiden Arten der Anwendung generativer Formerzeugung fügen sich nahtlos an tradierte Designverständnisse an, hier geht es eher um Prozessoptimierung, Automation und Designproduktion. So werden von dieser Seite aus zwar einige wichtige Impulse für methodische und formale Innovationen gegeben und als Folge Produktionsverfahren weiterentwickelt. Weitergehende Schritte in Richtung einer gestalterischen Eigenständigkeit, einer neuen Typologie algorithmischen Entwerfens scheint von dieser Seite aus aber eher nicht zu erwarten.

Abb. [15]) Formale Variationen als Unterscheidung zwischen einzelnen Produkten einer Familie – Leuchtenentwürfe von FutureFactories. (Dean et al, 2005)

Vielversprechender ist hier das natürliche Prinzip kleiner Variationen innerhalb eines Entwurfs, das zum organisierenden und formbildenden Element [Abb. 16] wird. In diesem besonderen Umgang mit Variationen, über die sich kontinuierliche Veränderungen organisieren lassen und die ohne Hilfe der quantitativen Intelligenz von Computern kaum möglich sind, könnte das künftige Alleinstellungsmerkmal algorithmischer Designansätze liegen. Es wäre ein Entwurfsansatz, der sich stark an natürlichen Vorbildern orientiert, dabei aber weniger an der formalen Nachahmung biologischer Formen interessiert ist, sondern an deren ästhetischer Qualität, den Struktur bildenden Kräften und informierenden Mustern, die diese Formen hervorbringen, um sie als generative Algorithmen in Computerprogramme zu implementieren.

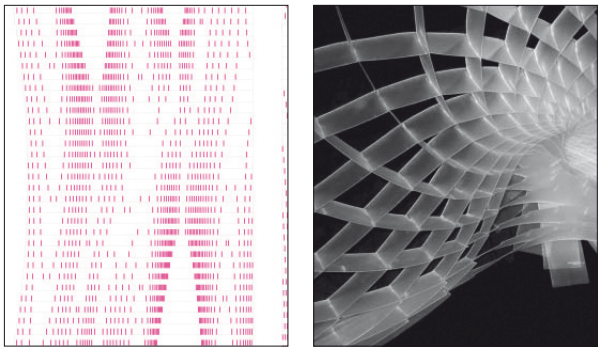


Abb. [16]) Selbstähnliches Muster als raumerzeugendes Prinzip.

(www.biothing.org, Stand: Januar 2006)

5. Beispiele algorithmischer Entwurfsräume aus den Bereichen Evolution, Parametrik, Optimierung

Die algorithmischen Entwurfsräume dieser Arbeit bedienen sich einzelner Bausteine aus den Bereichen der evolutionären Algorithmen, des parametrischen Designs und der Strukturanalyse bzw. -optimierung. Alle drei Gebiete können sowohl unter dem Gesichtspunkt der algorithmischen als auch dem der generativen Werkzeuge und Verfahren betrachtet werden. Dieses Kapitel skizziert aus dem Blickwinkel des Generischen einige der theoretischen Grundlagen zu diesen Bereichen, gibt eine Reihe von Anwendungsbeispielen und versucht jeweils eine kurze Bewertung ihrer Eignung als generative Entwurfswerkzeuge.

5.1 Evolution als Algorithmus

5.1.1 Evolutionäres Design

Die Begriffe ‚Design' und ‚Evolution' lassen sich auf verschiedenen Betrachtungsebenen miteinander verbinden. Die umfassendste Ebene ist die der Memetik *(vgl. Dawkins, 1977)* [01], nach der sich prinzipiell jede kulturelle Aktivität und somit auch das Entwerfen als ein Wettstreit von ‚Ideen' nach den Prinzipien der Evolution begreifen läßt. Die nächste, darin enthaltene Ebene ist die des *Evolutionary Design*. Hier geht es in erster Linie um die Entwicklung von theoretischen Modellen, bei denen sowohl die Veränderungen kultureller Artefakte über längere Zeiträume als auch einzelne Entwurfsprozesse als evolutionäre Vorgänge beschrieben werden. Die letzte Ebene schließlich ist die unmittelbare praktische Anwendung des Evolutionsprinzips im Designprozess in Form von evolutionären Algorithmen, ebenfalls unter dem Titel des *Evolutionary Design* geführt.

Die allgemeine Basis für eine Betrachtung des Design als evolutionäres Geschehen liefert *Dawkins (1983)* [02] mit seiner Theorie des *Universal Darwinism*, nach der jedes informationsverarbeitende System ein evolutionäres System sein kann, wenn es folgende Voraussetzungen erfüllt:

a) Transmission, bezeichnet zwei Aspekte: Reproduktion und Vererbung. Es muss eine Art von Vervielfältigung stattfinden, bei der Informationen weitergegeben werden. In welcher Form die Informationen vorliegen und wer oder was der Träger ist, ist dabei unerheblich: Eine Springmaus, die ihre Gene bei der sexuellen Fortpflanzung weitergibt, ein Computeralgorithmus, der eine Zeichenkette vervielfältigt, ein Architekt der einen Plan weiterreicht, usw.

b) Variation. Die übertragenen Informationen müssen beim Kopieren verändert werden: Perfekte Kopien können keine Basis für irgendeine Art von Evolutionsprozess sein, da jede Kopie die gleichen Eigenschaften hätte und somit auch keine einen evolutionären Vorteil gegenüber einer anderen erreichen könnte. Gleichzeitig dürfen die Änderungen aber auch nicht zu groß ausfallen, sonst entsteht ein System, bei dem es dem Zufall überlassen bleibt, einmal diese oder jene Eigenschaften zu bevorzugen.

c) Selection. Schließlich muss es noch Auswahlkriterien geben, die ausschlaggebend dafür sind, dass einige der gebildeten Varianten im Vorteil gegenüber anderen sind und so eine höhere Fortpflanzungs- oder Verbreitungswahrscheinlichkeit haben: Ein schärferes Auge, mit dem Feinde besser zu erkennen sind, eine neue Technologie, mit der sich ökonomisch günstiger produzieren lässt, ein Sofa das gekauft wird, weil es bequemer erscheint als andere, usw.

---

01) „The gene, the DNA molecule, happens to be the replicating entity that prevails on our own planet. (…) I think that a new kind of replicator has recently emerged on this very planet. (…) We need a name for the new replicator, a noun that conveys the idea of a unit of cultural transmission, or a unit of imitation. ‚Mimeme' comes from a suitable Greek root, but I want a monosyllable that sounds a bit like ‚gene'. (…) I abbreviate mimeme to meme. (…) Examples of memes are tunes, ideas, catch-phrases, clothes fashions, ways of making pots or of building arches." (Dawkins, 1977)

In diese Theorie eines umfassenden Evolutionsverständnisses, das über den biologischen Bedeutungszusammenhang hinausgeht, ist auch die Memetik eingebettet: Während die Gene die Replikatoren sind, mit denen die Baupläne des Lebendigen weitergegeben werden, sind Meme die Einheiten, die die Informationen jeder kulturellen Betätigung tragen: Ideen, Sprichwörter, ästhetische Konzeptionen, Vorlieben, Entwürfe für Gesetzestexte oder Zahnbürsten, usw. Unsere Körper sind Träger der Gene, unsere Hirne das Medium, in dem sich die Evolution der Meme abspielt. Aus der biologischen Evolution entsteht eine neue Ebene der kulturellen Evolution. An diese Vorstellung anknüpfend, formuliert *Gatherer (1999)* [03] die Grundlage für eine evolutionäre Designtheorie: *„If the design process begins with the production of novel ideas, generated from random combinations and mutations of existing ideas, continues with selection of those ideas for applicability to the problem in hand, and then proceeds to the (not necessarily accurate) transmission of those ideas, then the conditions necessary for an evolutionary process exist. That, in a nutshell, is the basis for an evolutionary theory of the design process."*

Dass die Vorstellung von Designprozessen als evolutionäre Vorgänge nicht unumstritten ist, liegt auf der Hand *(vgl. Gatherer, 1999)* [04]. Es ist zwar zweifellos so, dass Unwägbarkeiten, Zufälle und Unfälle bei jedem Entwurfsprozess eine gewisse Rolle spielen, ebenso natürlich Varietätserzeugung sowie Auswahlverfahren und -kriterien, nach denen diese Varietät bewertet wird. Trotzdem kann es als Konsens angesehen werden, dass Entwurfshandeln bewusstes, vom Entwerfer ausgehendes Agieren ist, das darauf abzielt, einen vorgefundenen Zustand mit Hilfe einer Designintervention in einen verbesserten Zustand zu überführen. Dem gegenüber steht die Evolution, bei der das *„survival of the fittest" (Darwin, 1859)* [05] kein besser oder schlechter in ethischen Kategorien bezeichnet, sondern schlicht eine höhere Fortpflanzungswahrscheinlichkeit. Zudem sind Evolutionsprozesse ‚blind', d.h. sie sind nicht zielgerichtet und kommen folglich auch ohne die Annahme bewusst handelnder Gestalterpersönlichkeiten aus.

Die Herausforderung für evolutionäre Designtheorien liegt darin zu klären, ob und wie sich bewusstes und zielgerichtetes Handeln mit einem unbewussten und ziellosen Prozess vereinbaren läßt. Ein möglicher Ansatz ist das Modell von *French (1999)* [06], bei dem die menschliche Erkenntnisfähigkeit ein Selektionsmechanismus ist, der auf den Pool der Meme wirkt: Ein grundlegender Auswahlmechanismus ist zunächst, ob eine Idee, die an uns herantransportiert wird, im Problemkontext eines Entwurfs überhaupt Sinn macht, ob sie also mit bereits vorhanden Denkmustern in Einklang zu bringen ist. Das Erkennen, wie sich bestimmte Ideen zu einer Neuheit kombinieren lassen, wäre dann ein weiterer Auswahlmechanismus, der die entsprechenden Meme (Ideen) bevorzugt und zu ihrer weiteren Verbreitung beiträgt. Ein anderer Ansatz, dieses Dilemma zu lösen, stammt von *Langrish (2006)* [07]. Er definiert verschiedene Arten von Memen: *„Recipemes, Selectemes, Explanemes"*. ***Recipemes*** sind Ideen, wie Dinge zu machen sind oder wie gehandelt werden soll – also grob gesagt Vorschriften und Methoden. ***Selectemes*** entsprechen Vorstellungen, was getan werden sollte oder was ‚besser' bedeutet, entsprechen also in etwa Wertvorstellungen und Entscheidungen. Die ***Explanemes*** schließlich, sind Ideen, die genutzt werden, um ‚Warum-Fragen' zu beantworten, sind also Begründungen und Motivationen. Durch diese differenzierte Unterteilung in drei Arten von Memen, die in unterschiedlichen Umgebungen miteinander konkurrieren, wird es auch hier möglich, dem ‚bewussten Gestalter' eine definierte Rolle in einer Evolutionstheorie des Entwerfens zuzuweisen.

---

04) „(…) it does seem as if thought always consists of a thinker-as-subject having thoughts about some object. A designer thus thinks about the artefact under construction and makes decisions regarding this structure etc. However, memetics insists that this intentionality, this subjectivity is an illusion produced by selection, just as the notion of design in nature is an illusion produced by selection. (…) The designer may protest: ‚But I solved it', but the memeticist would reply: ‚No, you were the brain/processing unit in which the cultural solution to the problem arranged itself." (Gatherer, 1999)

Sollte die Vorstellung, dass mit Designtätigkeit Dinge und Zustände bewusst und zielgerichtet zum Besseren verändert werden können, eine Illusion des menschlichen Geistes sein, ist es zumindest eine, die zur gestalterischen Arbeit motiviert und damit diese Veränderungen erst möglich macht – wenn vielleicht auch nicht auf dem direkten Weg, wie wir uns das denken und wünschen mögen *(vgl. Gatherer, 1999)* [08]. Unabhängig davon ist weder das Bild einer biologischen noch einer kulturellen Evolution geeignet, um daraus auf die direkte instrumentelle Anwendung evolutionärer Prinzipien in computerbasierten Designprozessen zu schließen. Die Ergebnisse, die mit evolutionären Algorithmen erreicht werden, sind bisher weit entfernt von der umfassenden Vielfältigkeit und den oftmals verblüffenden Lösungen, die biologische und kulturelle Evolutionsprozesse hervorbringen.

5.1.2 Evolutionäre Algorithmen

Die Arbeitsweise dieser Algorithmen beruht auf einer Nachahmung des Evolutionsprinzips. In der Algorithmentheorie werden sie den stochastischen Suchalgorithmen zugeordnet: Durch eine Akkumulation kleiner Verbesserungen von Generation zu Generation konvergieren Lösungen, die mit solchen Algorithmen erzeugt werden, allmählich zu einem Optimum, das über ein oder mehrere Zielkriterien beschrieben ist – evolutionäre Algorithmen eignen sich also auch für multimodale Optimierungsprobleme.

Man unterscheidet vier verschieden Arten solcher Algorithmen: ***Evolutionary Strategies (ES)*** *(Rechenberg, 1973)* [09], die von *Schwefel*, *Bienert* und *Rechenberg* entwickelt wurden, die sehr häufig verwendeten und robusten ***Genetic Algorithms (GA)*** *(Holland 1973, 1975)* [10/11] *(Goldberg, 1989)* [12], ***Evolutionary Programming (EP)*** *(Fogal, 1963) (Fogal, 1992)* [13/14] und das ***Genetic Programming (GP)*** *(Koza, 1992)* [15]. Die Unterschiede zwischen diesen einzelnen Typen haben mit der Entwicklungsgeschichte der Algorithmen zu tun, mit ihrer Leistungsfähigkeit und ihren Anwendungsfeldern. Die Besonderheit der *Evolutionary Strategies* beispielsweise besteht darin, dass nicht nur die Individuen einer Population verändert (mutiert) werden, sondern auch die Mutationsschrittweite selbst – bringen größere Veränderungen bessere Lösungen, steigt auch die Wahrscheinlichkeit, dass eine größere Mutationsschrittweite bevorzugt wird und umgekehrt. Die Spezialität des *Genetic Programming*, bei dem sich Computerprogramme nach dem evolutionären Prinzip selbst verändern, ist die Verarbeitung von Genotypen mit unterschiedlicher Länge. Unabhängig von solchen Unterschieden folgen diese Algorithmen im Wesentlichen alle dem selben Prinzip, das *Bentley (1999)* [16] in einer allgemeinen Architektur evolutionärer Algorithmen [Abb. 01] zusammengefasst hat, die hier vereinfacht wiedergegeben wird:

1) Initialisation. Als erster Schritt muss eine Startgeneration von Modellen möglicher Lösungen (Individuen) erzeugt werden. Will man beispielsweise eine Vase evolutionär optimieren, muss hier bereits definiert werden, welche Eigenschaften die Vase haben soll (etwa rotationssymmetrisch und nur aus Kreisen aufgebaut), welche davon veränderbar sein sollen (z.B. der Radius der Kreise) und welche nicht (z.B. die Position der Kreise in x- und y-Richtung). Jedes ‚Gen' eines Individuums (Modells) entspricht jetzt einem veränderbaren Parameter. Das ‚Chromosom' ist die Liste aller Gene, also veränderbaren Parameter eines Modells. (Anzahl Kreise, Radien, Positionen, usw.) Den Wert, den ein Gen aktuell hat, nennt man ‚Allel'. Bei der Initialisierung werden alle Gene aller Individuen der ersten Generation mit zufälligen Werten gefüllt.

---

08) „Those who are disturbed by the implication of this theory, that creativity is a a more random and messier process than we might care to admit, may console themselves with the thought that the diginity of the designer is not entirely forfeit. The meme pool is something to which all are capable of contributing, even if our contributions are more accidental than we may realise." (Gatherer, 1999)

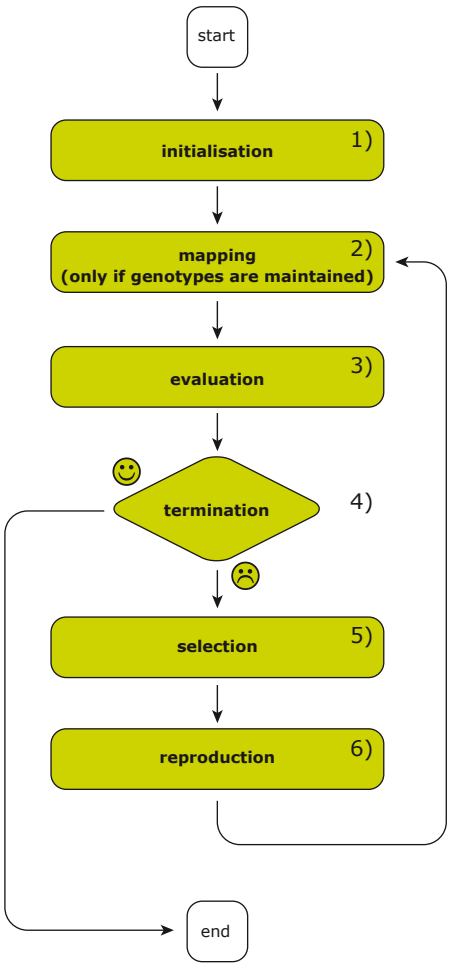


Abb. [01]) Vereinfachter, allgemeiner Aufbau evolutionärer Algorithmen. (nach: Bentley, 1999)

2) Mapping. Für das Vasenbeispiel bestünde ein Individuum aus einer Liste von Zahlen. Es ist auch möglich, diese Liste weiter zu codieren, üblicherweise in Binärcode, also in einer Zeichenkette aus Einsen und Nullen. Dieser Schritt ist allerdings nicht zwingend notwendig. Wird eine weitere Codierung vorgenommen, nennt man die codierte Form ‚Genotyp' und die zugrundeliegende Liste der Parameter ‚Phänotyp'. Ein Individuum kann also nur aus einem Phänotyp, oder aus einem Phänotyp und dem zugehörigen Genotyp bestehen. Die Übertragung eines Genotyps in den Phänotyp heißt *‚Mapping'* oder *‚Embryogenese'*.

3) Evaluation. Hier werden die Modelle nach vorab festgelegten Kriterien bewertet, für die Vase könnte das eine Mindestgröße für die Standfläche oder ein bestimmtes Fassungsvermögen sein. Die Rechenmodelle, mit denen die Phänotypen überprüft und bewertet werden, nennen sich ‚Fitnessfunktion' (Tests). Je nach Ergebnis des (der) Tests bekommt jedes Modell eine ‚Note' (Fitnesswert) zugewiesen. Diese Bewertung ist Grundlage für die nächsten Schritte.

4) Termination. An dieser Stelle wird geprüft, ob die Haltebedingungen erfüllt sind, also ob der evolutionäre Prozess beendet werden soll. Die Haltebedingungen, die hier normalerweise Verwendung finden, sind:

a) ein Individuum hat die geforderte Fitness erreicht,
b) eine bestimmte Anzahl an Generationen (Programmdurchläufen) ist erreicht,
c) eine bestimmte Zeit ist verstrichen oder
d) die Konvergenzrate (Verbesserungsrate) ist unter einen definierten Wert gefallen.

5) Selection. Auf Basis der Ergebnisse aus Schritt 3 werden nun die Individuen ausgewählt, aus denen eine neue Generation entstehen soll. Das Grundprinzip der Auswahl ist, eine größere Wahrscheinlichkeit dafür zu schaffen, dass sich besser bewertete Modelle fortpflanzen können, wobei aber eine gewisse Vielfalt erhalten bleiben muss. Um einen Pool von ‚Eltern' für die Folgegeneration zu schaffen, gibt es vier Standards: *Fitness Ranking* – je besser eine Vase umso mehr Kopien werden von ihr in den Elternpool gegeben; *Tournament Selection* – eine Vase wird mit zufällig ausgewählten, anderen Vasen aus der Population verglichen. Je mehr Vergleichsvasen sie übertreffen kann, desto größer die Wahrscheinlichkeit, dass sie gewählt wird; *Roulette Wheel Selection* – ebenfalls ein Wahrscheinlichkeitsverfahren, das über die Fitnesswerte gesteuert wird: eine fünfmal bessere Vase hat (beispielsweise) eine fünfmal höhere Wahrscheinlichkeit, als ‚Elter' ausgewählt zu werden. Und schließlich noch *Fertility*, bei der an Hand der Fitnesswerte festgelegt wird, wieviel ‚Nachkommen' ein Elter haben darf.

6) Reproduction. Als abschließender Schritt müssen jetzt aus den Eltern neue Nachkommen produziert, also (leicht) veränderte Modelle der Vasen erzeugt werden. Dazu werden die sogenannten genetischen Operatoren verwendet, die Gängigsten sind: *Mutation*, was bedeutet, dass der

Wert eines zufällig gewählten Parameters zufällig geändert wird, also z.B. der Radius eines Kreises vergrößert oder verkleinert wird, u.ä. *Crossover* bedeutet, dass die Parameterlisten zweier Individuen an einer zufälligen Stelle geschnitten und kreuzweise wieder zusammengesetzt werden. Von *Multiple Crossover* spricht man, wenn Chromosomen in mehrere Stücke zerlegt und daraus kreuzweise wieder neue gebildet werden. *Inversion* schließlich bedeutet, dass das Chromosom umgedreht, also in umgekehrter Reihenfolge interpretiert wird. Wenn alle ‚Kinder' *(offspring)*, also alle neuen, jetzt mehr oder minder leicht veränderte Vasenmodelle erzeugt sind, beginnt wieder Schritt 2 (oder 3).

Die üblichen Anwendungsfelder für evolutionäre Algorithmen liegen in Bereichen, die das Alltagsgeschäft von Architekten und Designern mehr am Rande tangieren: Sie werden etwa genutzt, um günstige Anordnung von Leiterbahnen für analoge Schaltkreise zu ermitteln *(Koza et al, 1997)* [17], die Anzahl und Verteilung von Schweißpunkten und -nähten an Karosserieverbindungen von Automobilen zu optimieren *(Bruns et al, 2006)* [18] oder Flugzeugtragflächen so zu verändern, dass sie günstigere aerodynamische Eigenschaften aufweisen als zuvor, *(Eby et al, 1999)* [19] – evolutionäre Algorithmen zählen heute zu den Standards unter den Optimierungsverfahren im Ingenieursbereich. Einen etwas kreativeren Umgang mit evolutionären Verfahren zeigt das Beispiel von *Sims (1994)* [20] ***Virtual Creatures***: In verschiedenen, simulierten physischen Umwelten entwickeln sich virtuellen Wesen, die darauf bewertet werden wie gut sie Schwimmen, Laufen oder Hüpfen können. Bei den ersten Beiden ist das Bewertungskriterium, grob gesagt, wie schnell sie vorwärts kommen, beim Hüpfen geht es um die Höhe. Die entstehenden Wesen werden durch Quader visualisiert, die den einzelnen Körperteilen entsprechen. Die Quader sind – was nicht visualisiert wird – verbunden durch jeweils eine aus sieben möglichen Kombinationen aus Gelenkart und zugehöriger Muskelbewegung. Das Verhalten dieser Bewegungsmechanismen in einer simulierten physischen Umwelt wird durch ein externes Kontrollsystem gesteuert. Im Ergebnis entsteht eine erstaunliche Vielfalt virtueller Kreaturen, die im simulierten Überlebenskampf die besten Eigenschaften gezeigt haben [Abb. 02]. Etwas näher am entwerferischen Bereich liegen Arbeiten wie die von *Kilian (2003)* [21], der genetische Algorithmen nutzt, um Freiformflächen in abwickelbare Bänder zu unterteilen, von *Willems (2006)* [22], der evolutionäre Algorithmen in Kombination mit *Pathfinding*-Algorithmen einsetzt, um Raumstrukturen nach funktionalen Kriterien zu organisieren [Abb. 03] oder wie eine Arbeit von *Menges (2006)* [23], der komplexe Formen mit Hilfe eines genetischen Algorithmus so kontrolliert, dass bestimmte Anforderungen an die Geometrie der Formen, die aus Herstellungsbedingungen abgeleitet sind, im Fortlauf des evolutionären Prozesses erhalten bleiben [Abb. 04].

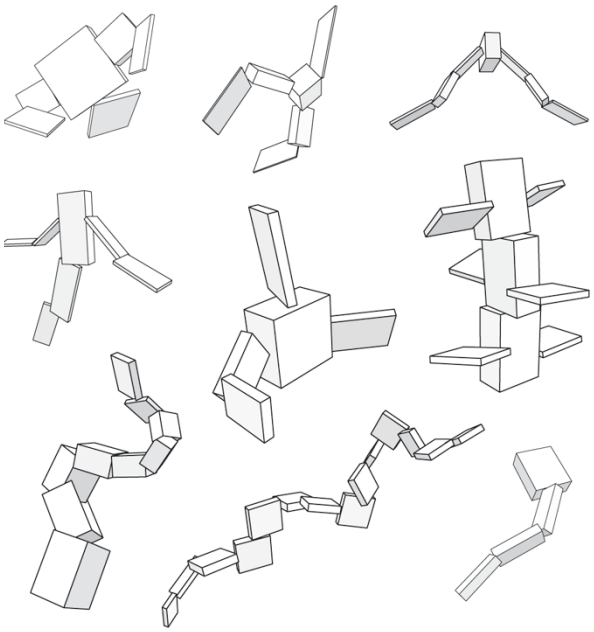


Abb. [02]) Virtual Creatures: Beispiele für evolutionär hervorgebrachte, ‚schwimmfähige' virtuelle Kreaturen. (Sims, 1994)

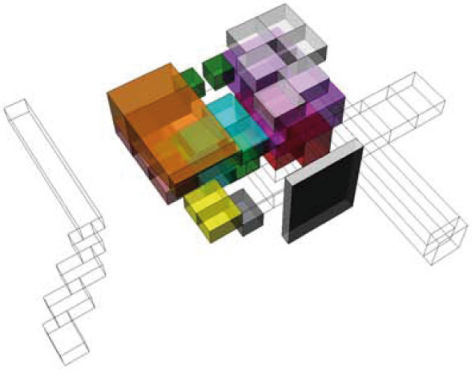


Abb. [03]) Schema zur Evolution von Raumstrukturen. (Willems, 2006)

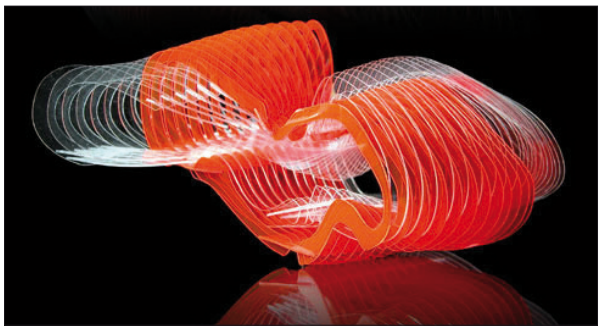


Abb. [04]) Kontrolle komplexer Formen mit Hilfe genetischer Algorithmen. (Menges, 2006)

Eines der nach wie vor raren Beispiele von Ansätzen, bei denen ein evolutionärer Algorithmus nicht nur einen Teilaspekt einer Designaufgabe bearbeitet, sondern ein komplettes Designprodukt hervorbringt, ist der *Coffee Table* von *Bentley (1999)* [24]. Die Idee hinter diesem Experiment war, einen Tisch zu erzeugen, der mit eingeschränkten handwerklichen Fähigkeiten mit fertig zugesägten Platten aus dem Baumarkt gebaut werden kann. ‚Tisch' wird durch fünf Auswahlkriterien definiert: *„size, mass, flatt upper surface, supportiveness and unfragmented" (Bentley, 1999)* [25]. Der evolutionäre Prozess beginnt mit einer zufälligen Anzahl an Blöcken, die jeweils eine zufällige Länge, Breite, Dicke sowie Position haben Abb. 05 / oben, wobei die Dicke bei dem abgebildeten Ergebnis auf 20mm festgelegt war – das gewünschte Standard-Plattenmaß. Nach rund 500 Generationen mit ca. 180 Individuen war jeweils ein Tisch erzeugt. Die Ergebnisse bezeichnet *Bentley (1999)* [26] selber als *„acceptable designs (as judged by humans)"* Abb. 05 / unten. Ein im Ergebnis ähnliches Projekt erzeugt ebenfalls tischähnliche Strukturen Abb. 06, Basis hier ist die Evolution eines 3D-*Lindenmayer*-Systems, das die Anordnung von Voxeln im Raum steuert *(Hornby et al, 2001)* [27]. Obwohl in beiden Fällen mit methodisch sehr interessanten Ansätzen gearbeitet wird und die Ergebnisse mit etwas Wohlwollen vielleicht auch als Tisch verwendet werden können, sind ihre Qualitäten aus entwerferischer Perspektive doch eher etwas fragwürdig – hier ist noch einiges an Experimentierarbeit zu leisten.

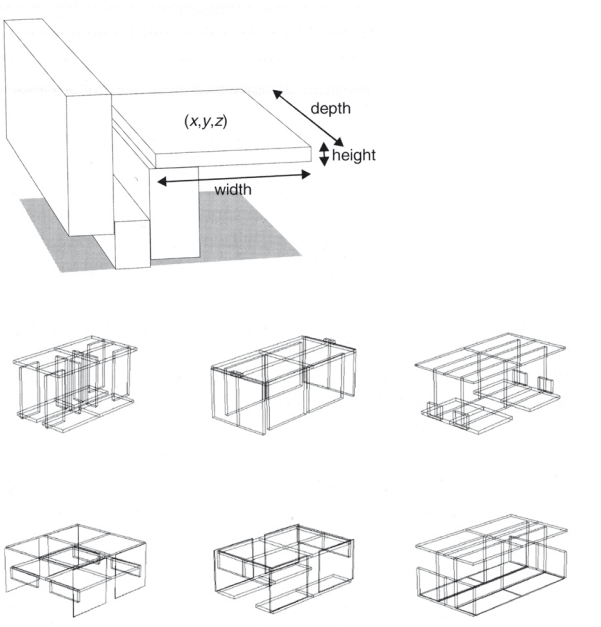


Abb. [05]) Coffee Table – Schema und verschiedene Ergebnisse. (Bentley, 1999)

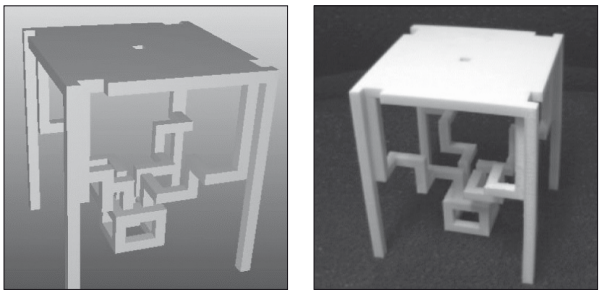


Abb. [06]) Evolutionär entwickelte Tische: 3D-Datei und entsprechendes Rapid-Prototyping-Modell. (Hornby et al, 2001)

5.1.3 Interaktive evolutionäre Algorithmen

Eine weitere Spielart sind die interaktiven evolutionären Algorithmen, bei denen der Mensch die Bewertung und Auswahl der erzeugten Individuen – ganz oder zum Teil – vornimmt. Diese Ansätze werden auch mit *„Artificial Evolution" (Frazer, 1995)* [28] oder *„Artificial Selection" (Sims, 1991)* [29] bezeichnet und zielen darauf ab, Spontanurteile und ästhetische Bewertungen in den Evolutionsprozess mit einzubeziehen – was bei den eben geschilderten Verfahren nicht möglich ist. Der Urvater dieses Algorithmentyps ist der *Blind Watchmaker*, den *Dawkins (1986)* [30] entwickelt hat, um das evolutionäre Prinzip der Anhäufung kleiner Änderungen über längere Zeiträume anschaulich und erfahrbar zu

machen. Über 16 Gene wird die Erzeugung der sogenannten *biomorphs* gesteuert, kleine, zweidimensionale Strichwesen, die Assoziationen zu biologischen Lebensformen wecken. In der Standardanwendung des *Blind Watchmaker* wird ein bereits erzeugter *biomorph* geladen, eine Reihe von Nachkommen erzeugt und auf dem Bildschirm dargestellt: Der Nutzer wählt dann eine der Figuren aus, neue Nachkommen werden erzeugt und so weiter. Auf diesem Wege ist es möglich, mehr oder minder zielgerichtet, eine große Bandbreite von Strichwesen mit erkennbarer Identität zu züchten: ‚Spinnen', ‚Farne', ‚Bäume', usw. Abb. 07

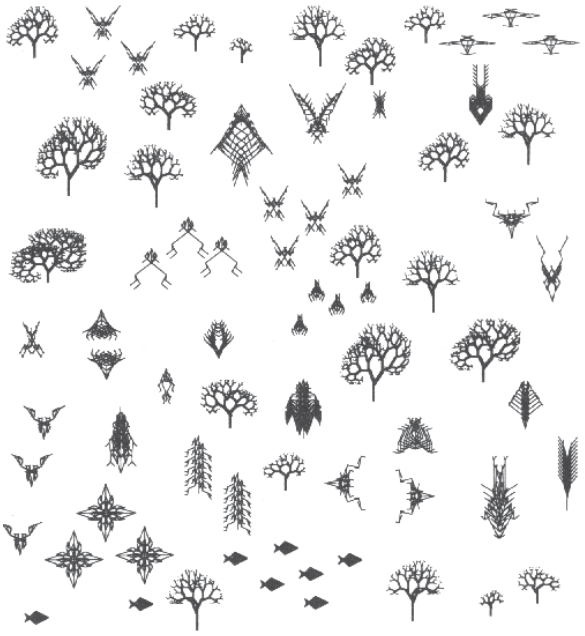


Abb. 07) „Safari Park" aus biomorphs. (Dawkins, 1986)

Die Konzeption des *Blind Watchmaker* hat auch Einzug in den entwerferischen Bereich gehalten, wo versucht wird, dieses generative Prinzip zu nutzen um damit zielgerichtet, aber evolutionär gestützt, Entwürfe zu entwickeln. Im bereits gezeigten Beispiel der Leuchten von *FutureFactories (Dean et al, 2005)* [31] wird eine Kombination aus genetischem Algorithmus und interaktiver Evolution verwendet. Der genetische Algorithmus kontrolliert hauptsächlich die geometrischen *Constraints* der Leuchten, etwa bezüglich der technischen Installationen. Die Bewertung des Nutzer soll ästhetisch ansprechende Formen hervorbringen. Allerdings ist hier die Identität der Lampen durch den vorgegebenen geometrischen Aufbau schon weitgehend festgelegt, ein Nutzer kann entsprechend nur Variationen aber keine neuen Typen von Lampen erzeugen Abb. 08.

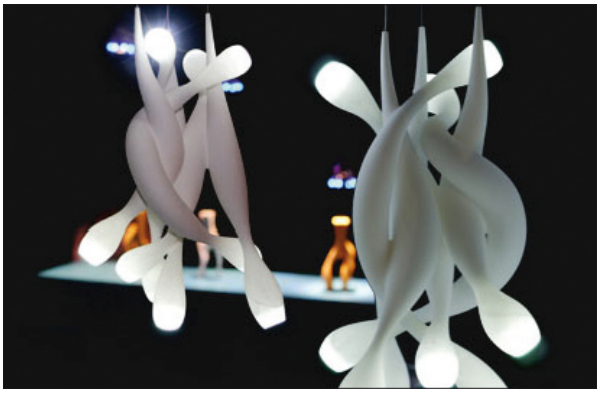


Abb. 08) Evolutionär gestützte Leuchtenentwürfe von FutureFactories. Trotz der freien Formen und obwohl jedes Stück ein Unikat ist, führt die generative Logik zu einer klar identifizierbaren Formensprache. (www.futurefactories.com, Stand: April 2006)

Abb. 09) Weingläser, entwickelt über ein Computerprogramm mit interaktiver Evolution. (Hung et, 2002)

Einen umfassenderen Ansatz verfolgt ein Projekt, bei dem *„a computational system framework for enhancing design in an evolutionary manner" (Hung et al, 2002)* [32] entstanden ist. Das Hauptaugenmerk in diesem Beispiel war, eine Software zu entwickeln, die während des gesamten Designprozesses einsetzbar ist und gleichzeitig nicht nur für einen speziellen sondern eine breite Palette von Entwurfsgegenständen nutzbar sein soll. Das System arbeitet ebenfalls mit einem

genetischen Algorithmus, der die Möglichkeit der ästhetischen Bewertung über die Auswahl einzelner Ereignisse bietet. Zusätzlich werden direkte Interaktionsmöglichkeiten auf verschiedenen Repräsentationsebenen geboten: Parametereinstellungen für den genetischen Algorithmus, genetische Repräsentation der Designmodelle, die direkt verändert werden kann *(„genetic engineering", (Dawkins 1986)* [33]*)*, 2D-Linien- und 3D-Volumenkörper-Darstellungen. Die Autoren versuchen die Relevanz und praktische Anwendbarkeit ihres System am Beispiel des Entwurfs von Weingläsern zu zeigen. Abb. 09

5.1.4 Evolutionäre Algorithmen als generative Entwurfswerkzeuge

Die scheinbare Nähe zwischen den beiden fundamentalen Prozessen des Entwerfens (Erzeugung und Einschränkung von Varietät) auf der einen Seite sowie grundlegender Elemente evolutionärer Prozesse (Mutation und Selektion) andererseits, legt den Gedanken nahe, dass es möglich sein müßte, mit evolutionären Algorithmen genauso komplexe und vielfältige Lösungen zu erzeugen, wie es in biologischen oder kulturellen Evolutionsprozessen geschieht. Diese Annahme ist kritisch zu betrachten. So haben die interaktiven evolutionären Algorithmen eine ganze Reihe ungelöster Probleme, vor allem, wenn sie mit dem Anspruch entwickelt werden, über die gesamte Schöpfungskette eines Designprozesses einsetzbar zu sein:

a) Gerade zu Beginn ist Entwerfen ein sprunghafter Prozess, der in kurzer Zeit eine breite Palette verschiedener Entwurfsansätze hervorbringen kann, mit unterschiedlichen formalen, strukturellen, formalen, etc. Eigenschaften. Solche Entwurfsphasen können über evolutionäre Prozesse, deren Basis die Akkumulation von langsamen, graduellen Veränderungen ist, nicht nachvollzogen werden – sie folgen einem ganz anderen Zeitrhythmus als das Entwerfen.

b) Ein weiteres Problem besteht in der Bewertung der erzeugten Alternativen. *Sims (1991)* [34] sieht, aus einer frühen 90er Jahre Perspektive von Computerleistung, eine sehr schnelle Geschwindigkeit in der Berechnung und Darstellung von Entwurfsvarianten als Voraussetzung für einen sinnvollen Einsatz interaktiver evolutionärer Algorithmen. Gerade dadurch aber, dass innerhalb kurzer Zeit eine sehr große Zahl an Ereignissen erzeugt werden kann, die sich in Kategorien der menschlichen Wahrnehmung kaum mehr unterscheiden lassen, wird eine sinnvolle Bewertung erschwert oder gar unmöglich.

c) Beide Probleme verstärken sich durch wachsende Komplexität eines Entwurfs, schon allein auf der rein geometrischen Ebene: Je mehr Parameter in einem Entwurf verändert werden können und umso mehr Freiheitsgrade im evolutionären Prozess vorhanden sein sollen, desto mehr Unsinn, sprich Formen ohne erkennbare Identität werden auch erzeugt. Nun ist es natürlich möglich, die Freiheitsgrade einzuschränken und Bedingungen einzuführen, die von vorneweg zur Erzeugung identifizierbarer Formen führen. Dadurch wird der evolutionäre Prozess aber zu einem Prozess der Variantenbildung anstatt der Varietätserzeugung. Man kann dieses Problem recht gut an den oben dargestellten Gläsern Abb. 09 nachvollziehen. Obwohl einfachere, rotationssymmetrische Körper ohnehin immer irgendwie an Blumenvasen, Gläser oder andere Gefäße erinnern und die erzeugten Geometrien auch als solche zu erkennen sind, ist kaum eines dabei, das wenigstens als formal ausdifferenziert zu bezeichnen wäre. Dieses Problem verschärft sich

---

34) „An alternative approach is to sample randomly in the neighborhood of a currently existing parameter set by making random alterations to a parameter or several parameters, then inspect and select the best sample or samples of those presented. This allows exploration through the parameter space in incremental arbitrary directions without requiring knowledge of the specific effects of each parameter. This is artificial evolution in which the genotype is the parameter set, and the phenotype is the resulting structure. Selection is performed by the user picking preferred phenotypes from groups of samples, and as long as the samples can be generated and displayed quickly enough, it can be a useful technique." (Sims, 1991)

drastisch, wenn die Geometrie eines Entwurfs komplexer wird und weitere, nicht-geometrische Parameter in das evolutionäre Modell einbezogen werden.

Der Autor dieser Arbeit hat das Experiment mit den Weingläsern in einem vergleichbaren Ansatz nachempfunden Abb. 10. Trotz einiger instrumenteller Hilfen, um die beschriebenen Probleme zu entschärfen (händisches Verändern der Gläser (Phänotypen) mit automatischer Aktualisierung des Genotypen, Evolution von Teilgeometrien, etc.) ist das Arbeiten mit dieser Methode näher an einer Übung in geistiger Disziplinierung als an einem Entwurfsprozess. Es ist denkbar, dass solche Ansätze eine gewisse Rolle als Werkzeug zur Variantenbildung bei einer weitgehend ausformulierten Entwurfslösung spielen könnten. Dass sie sich als umfassende generativen Entwurfswerkzeuge eignen, die in allen Phasen eines Entwurfsprozesses einsetzbar sind, ist bislang nicht mehr als eine Hypothese, die im praktischen Kontext noch keine Bestätigung gefunden hat.

Abb. 10) Klick and evolve – verschiedene Zwischenschritte und zwei Ergebnisse von Außenformen von Weingläsern, erzeugt mit Hilfe eines interaktiven evolutionären Algorithmus. (Schein, 2006)

Steigende Komplexität eines Entwurfs ist auch eine Schwierigkeit für die üblichen evolutionären Algorithmen, die automatisch, ohne Eingriffe des Nutzers ablaufen. Auch hier ist die Vorstellung, dass die computerbasierte Evolution überraschende, gar schockierend neue Ergebnisse hervorbringen könnte *(vgl. deLanda, 2002)* [35], bislang eine theoretische Vermutung, die erst noch zu belegen wäre. Kulturelle und biologische Evolutionsprozesse bilden hochgradig komplexe Umgebungen, deren vielfältige Wechselwirkungen, wenn sie denn alle bekannt und eindeutig quantifizierbar wären, sich trotzdem kaum als Computermodell umsetzen ließen. Bei allen gezeigten Beispielen findet nicht innerhalb eines bunten Zoos ein Konkurrenzkampf von Phänotypen mit unterschiedlichsten Eigenschaften statt. Ganz im Gegenteil, in einem computerbasierten Evolutionsprozess konvergieren die Ergebnisse sehr schnell zu einem Optimum und die einzelnen Individuen sind entsprechen auch kaum mehr voneinander zu unterschieden. Die Darstellung der *Virtual Creatures* beispielsweise mag den Eindruck erwecken, dass alle diese Individuen ‚auf einen Rutsch' entstanden sind, tatsächlich hat sich jedes von ihnen innerhalb separater Durchläufe des genetischen Algorithmus entwickelt.

Das Phänomen der Konvergenz ist kein unerwünschter Nebeneffekt, sondern logische Folge des Evolutionsprinzips – einmal gewonnene Vorteile werden beibehalten und nur dann aufgegeben, wenn sie durch andere, höher bewertete Vorteile aufgewogen werden. Die Wahrscheinlichkeit, dass dies geschieht, ist zu Beginn eines künstlichen Evolutionsprozesses recht hoch und führt zu einer schnellen Verbesserung des globalen Fitnesswerts, nimmt aber dann auch sehr schnell ab, bis nur noch marginale oder gar keine Veränderungen mehr stattfinden Abb. 11. Da die Konvergenz in Richtung der Bewertungskriterien führt, die definiert wurden, weiß man auch schon in etwa, wie das erzeugte Ergebnis sein wird. Als Optimierungsalgorithmen werden evolutionäre Algorithmen (im entwurfsnahen Kontext) vor allem zur Lösung geometrischer Probleme (Form und Anordnung geometrischer Instanzen), zur strukturellen Optimierung von Tragwerken oder von einzelnen Bauteilen eingesetzt. Ihre Anwendung ist vor allem dann sinnvoll, wenn:

a) keine andere Methode der Optimierung bekannt ist,
b) eine andere Optimierungsmethode zu zeitaufwendig wäre,
c) mehrere Optimierungsziele gleichzeitig erreicht werden sollen und
d) um zu experimentieren, ob mit der evolutionären Methode bessere Ergebnisse zu erzielen sind als mit anderen Verfahren.

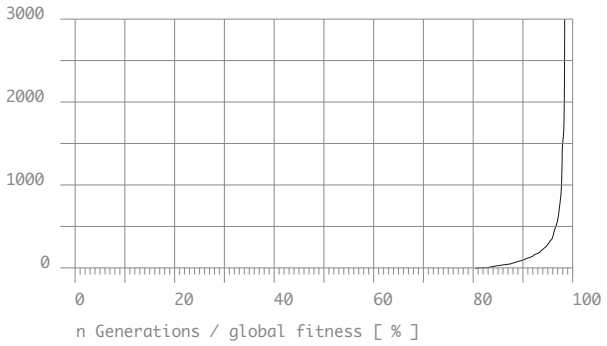


Abb. [11]) Eine typische Kurve des Evolutionsfortschritts für genetische Algorithmen – von links nach rechts steigt der Fitnesswert, von unten nach oben die Anzahl Generationen.

Als Optimierungswerkzeuge werden evolutionäre Algorithmen auch im praktischen Teil der Arbeit eingesetzt. Dass sie als generative Werkzeuge trotz einer prinzipiellen Vorhersagbarkeit dann doch für die ein oder andere kleinere Überraschung gut sein können, liegt an einer besonderen Eigenschaften dieser Algorithmen: Bei vielen generativen Methoden ist ein Ausgangszustand *A* eines Designmodells, ein gewünschter Endzustand *B* und entsprechende Operatoren, die das Modell von *A* nach *B* überführen, zu definieren, entsprechend dem Konzept der Designoperatoren, das *Mitchell (1990)* [36] formuliert hat. In diesen Operatoren drücken sich die Denkstrukturen eines Entwerfers aus, seine Vorstellungen, was getan werden muss, um von *A* nach *B* zu gelangen. Demgegenüber genügt es bei den evolutionären Algorithmen einen Ausgangszustand und Zielkriterien für einen gewünschten Endzustand zu formulieren. Den Weg von *A* nach *B* findet der Algorithmus von selbst, ist dadurch vom Denken des Entwerfers etwas unabhängiger und erzeugt auch Lösungen, die der Designer auf Grund seiner gedanklichen Formatierung so nicht entwickelt hätte.

Zusammenfassend kann man sagen, dass evolutionäre Algorithmen besonders für Projekte geeignet sind, in denen komplexe Strukturen entwickelt werden sollen, die durch einige wenige Parameter kontrolliert werden, deren Wechselwirkungen aber nicht eindeutig bestimmbar sind. Gestalterisch besonders interessant ist der Unschärferaum, der durch die stochastischen Elemente in dieser Art von Algorithmen entsteht, da dadurch auf dem Weg von der Initialisierung bis zur Annäherung an unterschiedliche Zielkriterien immer wieder neue, nicht genau vorhersehbare Lösungen entstehen.

5.2 Parametrisches Design

In Designprozessen werden Werte für geometrische, mechanische, ökonomische, funktionale, aber auch soziale, ökologische etc. Einflußfaktoren bestimmt, miteinander in Beziehung gesetzt und verändert. Generative Algorithmen haben mindestens einen, meist aber eine größere Zahl von Parametern, mit denen ihre Ausführung beeinflußt werden kann. Parametrisches Design wäre demnach nicht nur ein Oberbegriff für generatives Design, sondern für Design insgesamt. Wenn heute allerdings von parametrischem Design gesprochen wird, bezeichnet dies üblicherweise Methoden

---

36) operatorX (State1) = (State2)

„State1 and State2 are variables. By specifiying the range of State1 (the argument of the operator), we can establish the circumstances under which the operator can properly be applied. And by specifying how to compute State2 for any value of State1, we can define the effect of the operator. An operation (instance of the operator) is performed whenever the operator is applied to a particular state in the range of State1." (Mitchell, 1990)

der Verknüpfung geometrischer Parameter eines *CAD*-Modells. Darüber wird das Modell so definiert, dass die Manipulation eines geometrischen Parameters automatisch eine Aktualisierung der assoziierten Parameter und damit des gesamten Modells nach sich zieht.

5.2.1 Assoziative Geometrie

Burry (2003) [37] definiert parametrisches Design über die Schritte *„explicit geometry modeling, parametric modeling, associative geometry modeling"*. Explizites Modellieren bzw. Konstruieren ist der nach wie vor gebräuchlichste Weg zum Erstellen virtueller Designmodelle am *CAD*-System. Geht man z.B. von einem Rechteck *R*, einer Linie *L* und einem Kreis *K* aus, wie in der Abbildung [Abb. 12] grün dargestellt, und verschiebt man *K* nach *K'*, bleibt der Rest davon unberührt. Soll die Konnektivität der Linie von Rechtecksecke zu Kreismittelpunkt erhalten bleiben, muss die Linie in einem weiteren Schritt manuell entsprechend angepasst werden. Die orange Darstellung gibt ein Beispiel für einfache parametrische Modellierung. Hier ist der Startpunkt *P1* der Linie *L* mit einer Ecke des Rechtecks verknüpft, der Endpunkt *P2* mit dem Mittelpunkt des Kreises *K*. Wird das Rechteck *R* zu *R'* verändert und / oder der Kreis *K* zu *K'* verschoben, wird dadurch auch die Linie automatisch angepasst. Dann ist es schließlich noch möglich, nicht nur einzelne Parameter geometrischer Instanzen direkt miteinander in Beziehung zu setzen, sondern auch die Beziehung selbst untereinander zu assoziieren, wie im blauen Teil der Abbildung [Abb. 12] gezeigt. Hier wird für die Kante *a* des Rechtecks *R* eine bestimmte Länge, im Beispiel die Einheit 20, festgelegt. Die Kante *b* soll die Hälfte der Länge von *a*, die Länge *c* der Linie *L* soll die Hälfte von *a* + *b* sein, der Radius *d* des Kreises *K* soll der Hälfte von *a* - *b* entsprechen, *P1* und P2 sollen weiterhin mit der Rechtecksecke und dem Kreismittelpunkt verknüpft sein. Sind die Parameter und Beziehungen eines geometrischen Modells in dieser Weise miteinander verbunden, genügt die Änderung eines Parameters oder einer Beziehung, um das komplette Modell zu aktualisieren. Wird, wie im Beispiel, der Wert für *a* auf 10 gesetzt, passt sich das gesamte Modell ohne weiteres Zutun von Aussen entsprechend an.

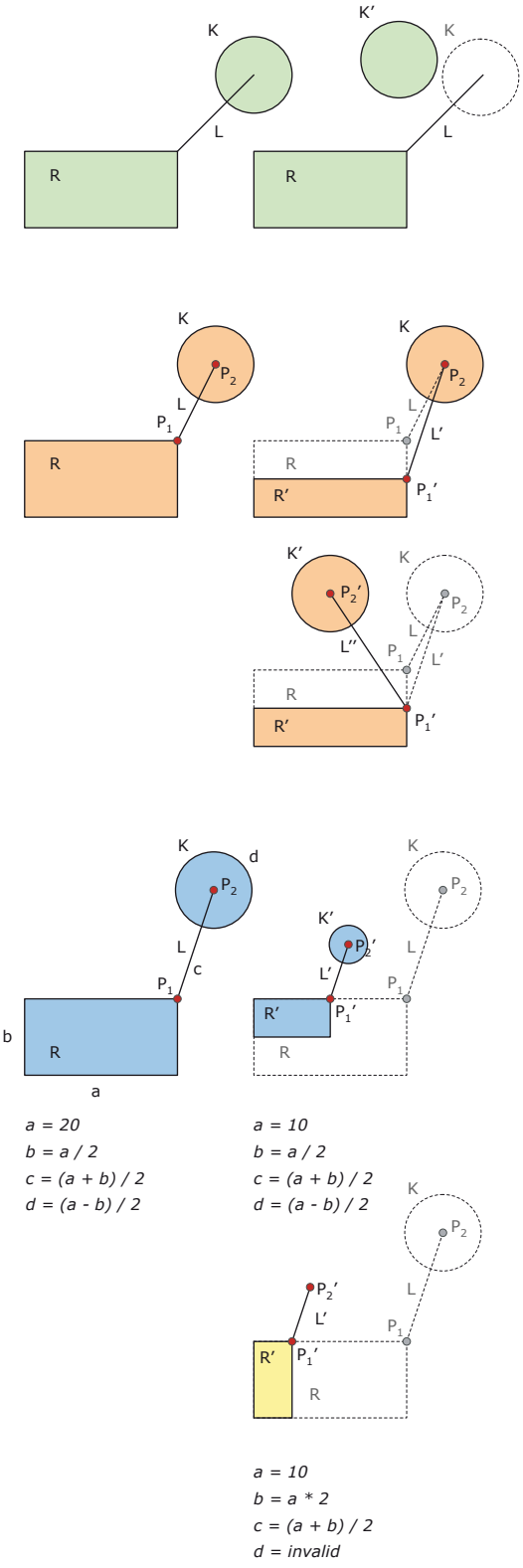


Abb. [12]) Von der CAD-Modellierung (grün) über parametrisches Modellieren (orange) zur assoziativen Geometrie. (nach Burry, 2003, verändert)

Von parametrischem Design spricht man, wenn parametrische Modellierung oder assoziative geometrische Modellierung als Methode des Entwerfens angewendet wird. Im generativen Kontext wird ein Beziehungsgeflecht von Parametern mit Hilfe von Programmierung entwickelt und abgebildet, viele *CAD*-Anwendungen stellen auch – mit mehr oder minder großem Leistungsumfang – einen Historienbaum zur Verfügung, mit dem ebenfalls parametrische Designmodelle entwickelt werden können. Von der Arbeit mit solchen Historienbäumen werden viele Entwerfer bereits Erfahrungen mit den zwei hauptsächlichen Problemen parametrischer Designmodelle kennen: *Over-Constraining* und *Under-Constraining*. Mit *Over-Constraining* ist der Effekt bezeichnet, dass ein Designmodell durch zu viele Parameter oder auch falsch assoziierte Parameter beschrieben wird, die zu Widersprüchen bei der Aktualisierung eines Modells oder ungültigen, nicht interpretierbaren Werten führen. In der Abbildung [Abb. 12] gelb dargestellt, wird beispielsweise die Regel, das die Kante *b* die Hälfte der Länge von Kante *a* haben soll, dahin gehend geändert, dass *b* nun 2 mal *a* ist. Die Linie kann nach dieser Änderung noch gezeichnet werden, der Kreis hingegen bekommt einen negativen Radius und kann nicht mehr gezeichnet werden. Wären jetzt mit dem Kreis weitere Instanzen assoziiert, könnten auch diese nicht mehr umgesetzt werden. Ein ähnliches Problem tritt auf, wenn zu wenige Bedingungen *(Under-Constraining)* gesetzt werden. So sind etwa im blauen Teil der Abbildung [Abb. 12] keine Angaben darüber gemacht, wie sich der Winkel verändern soll, mit dem die Linie *L* an Rechteck *R* anliegt, wenn die Kantelänge *a* verändert wird. In der Anschauung scheint es logisch, dass der Winkel gleich bleibt, für ein parametrisches Modell wäre hier zu wenig Information vorhanden. Weder die Line, noch der Kreis und eventuelle weitere, damit verbunden geometrischen Instanzen könnten erzeugt werden, da die Beziehung zwischen Winkel und Rechteck nicht eindeutig definiert ist.

Aus der Perspektive eines computerbasierten parametrischen Modells sind zu wenige Bedingungen ein geringes Problem, da sie sich in einem Abbruch des Programmablaufs äußern und so relativ einfach zu identifizieren und zu beheben sind. Schwieriger ist es, wenn zuviele oder falsche Bedingungen gesetzt sind, da diese unter Umständen nur bei ganz bestimmten Eingabeparameter zu einem Programmabbruch oder einer nicht beabsichtigten Interpretation der Eingabedaten führen. Das Problem von zu vielen, falschen oder zu wenigen Parametern ist aber nicht nur für die Ausführbarkeit eines parametrischen Programms relevant, sondern auch für die Interpretation der damit erzeugten Modelle. Es ist durchaus möglich, dass ein Designmodell problemlos mit allen möglichen Eingabeparametern erzeugt werden kann, das Ergebnis in der Interpretation seiner Aussage trotzdem falsch ist. Wenn beispielsweise die Höhe einer Treppe mit der Geschosshöhe eines Gebäudes assoziiert ist, nicht aber die Anzahl der Stufen, Tritthöhe und Tritttiefe, kann eine Änderung der Geschosshöhe zu einer Stufengeometrie führen, die durch das Raster der DIN-Norm fällt – es waren zu wenige Bedingungen gesetzt. Ähnliche Probleme treten auch auf, wenn ein Modell *over-constrained ist*: Die Treppe ist richtig assoziiert, eine Vergrößerung der Geschosshöhe führt zu einer korrekten, neuen Stufenanordnung, damit auch zu einer größeren Treppe. Diese braucht wieder mehr Freiraum vor dem ersten Tritt, die mit dem Freiraum verbundene Wand wird verschoben, der dahinterliegende Raum ebenfalls ... bis letztlich die eine Treppe den ganzen Entwurf so verändert hat, dass er nicht mehr auf das geplante Grundstück passt. Ein parametrisches Modell sollte also weder zuviele noch zuwenige Bedingungen haben. Dies gilt für die Ebene der Programmierung als auch der Konzeption und Interpretation der erzeugten Modelle. Dies zu erreichen ist keine triviale Aufgabe, besonders dann, wenn komplexere Beziehungsgeflechte ausgebildet werden sollen. Ob sich dies in der projektbezogenen Anwendung lohnt, ist wiederum eine Frage der Ökonomie, wie in *Kapitel 3.2.4* diskutiert.

5.2.2 Kategorien parametrischen Designs

Da jedes gegenständliche Design in irgendeiner Form mit der Organisation geometrischer Instanzen zu tun hat, sind auch die möglichen Anwendungsfelder parametrischen Designs entsprechend weit gespannt. Entwurfsansätze, die

sich der geometrischen parametrischen Modellierung bedienen, fassen *deLuca und Nardini (2002)* [38] unter der Kategorie *„Algorithmic Parametric Approach"* zusammen, worunter auch die größte Zahl bisher realisierter Arbeiten zum parametrischen Design fällt. Ein populäres, wenn auch schon etwas älteres Beispiel dafür, ist die *Waterloo Station* von *Grimshaw (1993)* Abb. 13.

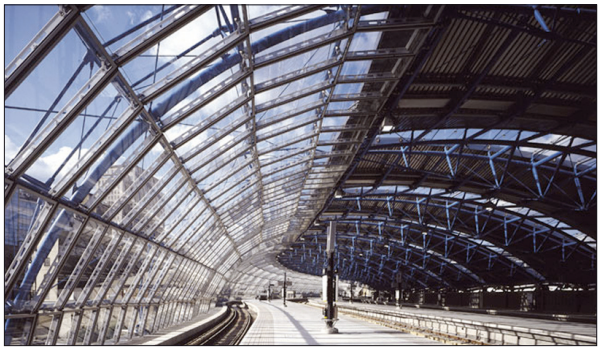


Abb. [13]) Ausschnitt der Überdachung der Waterloo-Station in London.
(Grimshaw, 1993)

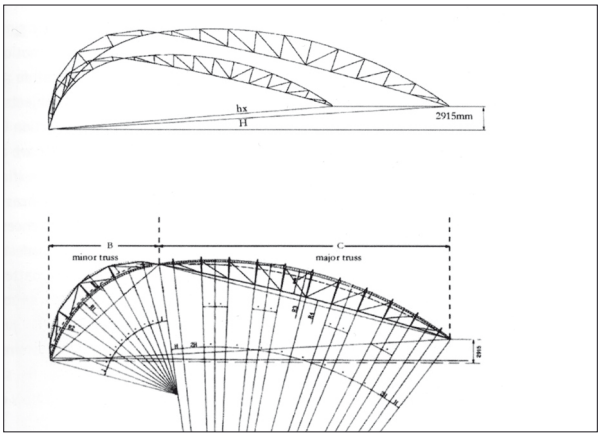


Abb. [14]) Parametrische Beschreibung des Skalierungsfaktors der Trägergeometrie: hx = ((2915 2 + (B + C) 2) 1/2, wobei B die Spannweite des kleineren und C die Spannweite des größeren Trägers ist.
(vgl. Kolarevic, 2003)

Ein kennzeichnendes Element dieses Entwurfs ist die Überdachung des Bahnhofs, die durch die Anordnung der Gleise und die Vorgaben des Grundstücks aus einem gekrümmten, sich verjüngenden Grundriss entwickelt werden musste. Das Dach setzt sich aus 36 Trägern zusammen, wobei jeder Träger aus zwei Bogenträgern besteht. Von einem dieser Träger wurde ein parametrisches Modell entwickelt, so dass sich die Trägergeometrie den unterschiedlichen Spannweiten anpassen konnte Abb. 14. Diese parametrische Beschreibung wurde dann noch auf die einzelnen Bauteile der Träger sowie die Bauelemente der Bedeckung ausgedehnt. Im Endergebnis entstand so ein parametrisches Modell des Dachs, mit dessen Hilfe der Entwurf im Laufe des Entwurfsprozesses immer weiter verfeinert und sich verändernden Gegebenheiten angepasst werden konnte *(vgl. dazu: Kolarevic, 2003)* [39].

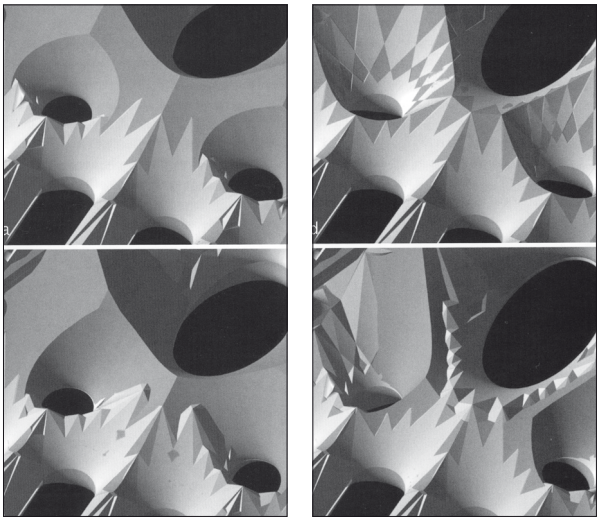


Abb. [15]) Parametrische Variationen eines Fensterdetails der Sagrade Familia.
(Burry, 2003)

Zunächst weniger aus praktischem, sondern mehr aus Forschungsinteresse heraus, hat sich ein anderes parametrisches Designprojekt entwickelt, das seit einigen Jahren im Kontext der Fertigstellung von *Gaudis Sagrada Familia* durchgeführt wird. Angesichts der Tatsache, das *Gaudis* Entwürfe im überwiegenden Teil aus skalierten Modellen bestehen, die nur einen Teil der Kathedrale beschreiben, dass weiterhin viele der Modelle im spanischen Bürgerkrieg verloren gingen oder beschädigt wurden und Gaudi keine Angaben gemacht hat, nach welchen Prinzipien er seine besondere Formensprache entwickelt hat, ging es hier zunächst darum, die generative Logik der *Sagrada Familia* aufzudecken. Als Schlüsselelement von *Gaudis* Sprache wurden schließlich Regelflächen identifiziert, also Flächen, die entstehen, wenn eine Gerade eine Raumkurve entlang geführt

wird. Auf Basis dieser Erkenntnisse war es möglich, detaillierte parametrische Modelle zu entwickeln und durch Änderung der Parameter der Regelflächen [Abb. 15] einen systematischen Abgleich mit *Gaudis* Gipsmodellen vorzunehmen, bis die digitale Modellen diesen entsprachen. Hier spielte parametrisches Design zunächst die Rolle eines Analyseinstruments, mittlerweile wird es bei der *Sagrada Familia* auch als Methode zur weiteren Planung der Kathedrale sowie als Mittel, Produktionsdaten zu verteilen, genutzt *(vgl. Burry, 2003)* [40].

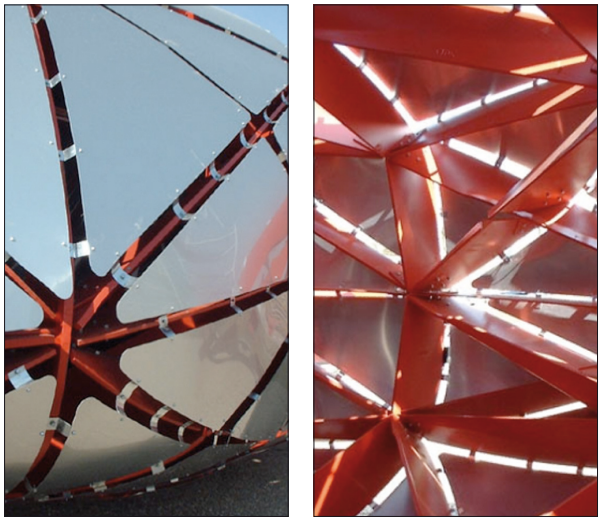


Abb. [16]) Detailansichten des parametrisch erzeugten Web of North Holland Pavilion. (Oosterhuis, 2002)

Eine zweite Kategorie parametrischer Entwurfsansätze ist der *„Variational Approach" (deLuca et al, 2002)* [41], bei dem geometrische Parameter mit nicht-geometrischen Parametern (Volumen, Gewicht, Krafteinwirkung, etc.) verbunden werden. Ein Beispiel für diesen Ansatz ist *Oosterhuis' (2002) Web of North Holland Pavillon* [Abb. 16]. Basis dieses Entwurfs ist ein Flächennetz aus Dreiecken mit sechswertigen Knoten. Als hauptsächlicher formgebender Parameter in einem virtuellen Environment diente simulierte Schwerkraft, die genutzt wurde, um die Grundgeometrie aus einem sechswertigen Dreiecksnetz zu transformieren. Aus dem so veränderten Netz wurden dann über verschiedene parametrische Skripte die Zuschnitte der stützenden Stahlstruktur, die Verbinderteile für diese Struktur sowie der Zuschnitt für die Aluminiumpaneele errechnet und die benötigten Fertigungsdaten ausgegeben. Eine Änderung des simulierten Schwerkrafteinflusses auf das ursprüngliche Raster verändert entsprechend alle Bauteile mit. Beim *Web of North Holland Pavilion* ist jedes Bauteil ein Unikat, seine Fertigung folgt der Logik des *File-To-Factory*, bei der die Produktionsdaten direkt, ohne Umweg über den Plan, an *CNC*-gesteuerte Fertigungsmaschinen weitergegeben werden.

Abb. [17]) Dynaform während der Aufbauphase (oben) und wave. (Franken, 2001 / 2000)

Ebenfalls der Kategorie *Variational Approach* zuzuordnen sind Arbeiten von *Franken* wie *Dynaform (2001)* [Abb. 17 / oben], *Brandscape (2000)* oder *Wave (2000)* [Abb. 17 / unten], alle drei Messearchitekturen, für Auftritte von BMW entwickelt. Das generative Prinzip aller drei Entwürfe ist ähnlich und gleicht dem, das *Oosterhuis* für den *Web of North Holland Pavilion* verwendet hat. Auch diese Messearchitekturen basieren auf Grundgeometrien, die durch simulierte Kräftefelder (Lichteinwirkung, Gravitation, Windlasten u.ä.) transformiert werden. Ebenso ist Programmierung eine wichtige Methode, mit der aus der erzeugten und variierten Grundstruktur alle Bauteile eines Entwurfs abgeleitet, ausdifferenziert und schließlich als *CNC*-geeignete Produktionsdaten weitergegeben werden. Eine wichtige Erkenntnis für die Verknüpfung geometrischer und physikalischer Parameter, die *Franken* im Kontext dieser Projekte formuliert, ist, dass die im virtuellen Environment simulierten Kräfte nur

bedingt aussagekräftig für die in der physischen Realität wirkenden Kräfte sind. Obwohl Freiformen, die durch virtuellen Krafteinfluss entstandenen sind, oft günstige statische Eigenschaften aufweisen, ist eine weitere Verfeinerung der Tragwerksgeometrie in Zusammenarbeit mit Bauingenieuren und Statikern nötig *(vgl. Franken, 2003)* [42]. Diese Formen tragen durch den generativen Prozess eine Idee von struktureller Optimierung in sich, sind aber keine statisch optimierten Geometrien.

Beide Arten von parametrischem Design (*Parametric Algorithmic Approach* sowie *Variational Approach*) werden im praktischen Teil dieser Arbeit experimentiert. Eine letzte Kategorie, der *„Artificial Intelligence Approach" (deLuca et al, 2002)* [43], ist eine Kombination von Expertensystemen und parametrischem Design, der im praktischen Designkontext bisher weniger eine Rolle spielt und hier nicht weiter besprochen werden soll.

5.2.3 Parametrisches Design als generatives Entwurfswerkzeug

Im praxisnahen Kontext kommen Methoden des parametrischen Design aufgrund ihrer klaren Programmstrukturen und ihrer vielfältigen Verwendbarkeit häufig zur Anwendung. Im Gegensatz zu den evolutionären Algorithmen ist hier allerdings jeder einzelne Transformationsschritt explizit zu definieren, wobei genau darauf zu achten ist, welche Bedingungen wie miteinander verknüpft werden. Parametrische Designmethoden können einerseits ein selbstverständliches Bindeglied eines durchgängigen digitalen Designprozesses sein, gerade im Hinblick auf *File-To-Factory* - Abläufe. Sie erzeugen auch wieder eine eindeutig identifizierbare Entwurfssprache, im Sinne eines klar abgegrenzten algorithmischen Entwurfsraums. Parametrische Designmethoden werden allerdings auch umso unflexibler, je komplexer und spezifischer das Regelsystem ist, das mit ihrer Hilfe definiert und implementiert wird. Diese mangelnde Flexibilität kann abgemildert werden, in dem man parametrische Algorithmen, entsprechend der in *Kapitel 4.1.2* beschriebenen Logik, konzeptionell in generische und generative Bestandteile unterteilt. Die beiden abschließenden Beispielen aus zwei Studienprojekten zum Thema *‚Parametrisches Design'* *(Schein et al, 2004, 2005)* [44/45] verdeutlichen dieses Prinzip nochmals.

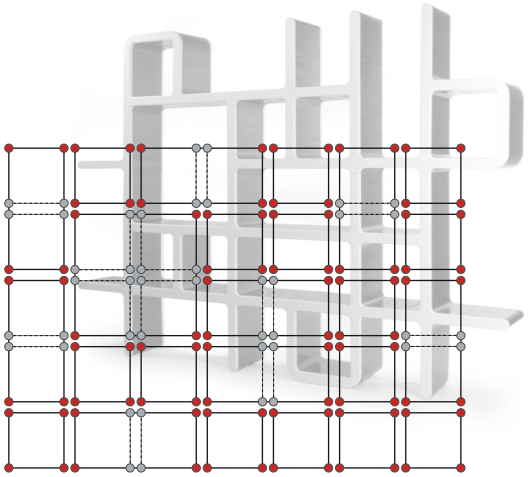


Abb. [18]) Rasterstruktur eines Regals, entwickelt in einem parametrischen Designprozess nach einem abgewandelten Tetris-Prinzip. Die formale Interpretation (Prototyp-Hintergrund) wurde konventionell, von Hand entwickelt und ausgeführt. (Bayer, Galle, 2004)

Die Grundstruktur des Regalentwurfs, der in der Abbildung [Abb. 18] gezeigt ist, basiert auf einem einfachen orthogonalen Raster, aufgebaut aus Quadraten einer definierten Kantenlänge und einem definierten Abstand. Nach dem Prinzip des Computerspieleklassikers *Tetris* werden aus einer Bibliothek vordefinierter Elemente zufällig ausgewählt und das Raster damit aufgefüllt. Die so entstehenden, zweidimensionalen Strukturen wurden dann händisch weiter interpretiert und eine konkrete Umsetzung in Form eines Regals entwickelt – der parametrische Ansatz zur Erzeugung der geometrischen Grundstruktur ist von der detaillierten konstruktiven Interpretation abgekoppelt. Damit ist eine klare Schnittstelle definiert und die generative Logik kann, ohne weitere Eingriffe in die Programmstruktur, in einen anderen Kontext übertragen werden und beispielsweise als Gebäude- oder Raumgrundriss interpretiert und weiterentwickelt werden. Das zweite Beispiel ist ein parametrisch definierter Baustein aus *UHPC*, einem stahlfaserarmierten Hochleistungsbeton. Dieser Baustein wurde als solitäres parametrisches Element entwickelt und program-

miert, und läßt sich in verschiedenen Entwurfskontexten interpretieren, etwa als Fassadenelement, als Bestandteil eines Raumteilers oder einer tragenden Struktur [Abb. 19].

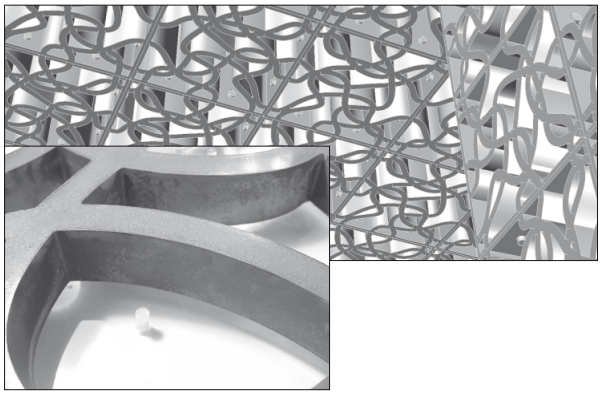


Abb. [19]) Detail eines parametrisch definerten Betonbausteins (Prototyp) und dessen Integration in einer Pavilionstruktur (Rendering). (Ennecker, 2005)

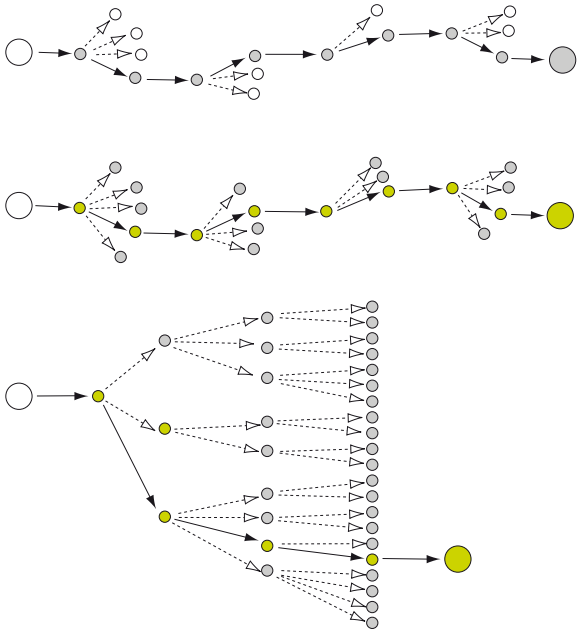


Abb. [20]) Nicht-optimierende, einstufig und mehrstufig optimierende Suchstrategien. (nach: Becker et al, 1994)

5.3 Design aus Optimierung

Ähnlich wie das Thema ‚Parametrik', hat auch das der ‚Optimierung' eine umfassende Bedeutung für das Entwerfen, man kann Design auch als reine Optimierungsaufgabe verstehen – die Überführung eines Soll- in einen verbesserten Ist-Zustand. Dies ist im Kontext dieser Arbeit von Belang, weil die Optimierungsverfahren, die genutzt werden, der Domäne der Ingenieure entnommen sind, denen die obige Auffassung näher liegt als Entwerfern. Solche Verfahren werden hier aber nicht nur zu Optimierungs-, sondern darüber hinaus zu Gestaltungszwecken verwendet, was im Folgenden erläutert wird.

5.3.1 Optimierung und Suche

Optimierungsstrategien zählen allgemein zu den Suchstrategien, von denen drei Hauptgruppen [Abb. 20] unterschieden werden *(vgl. Becker et al, 1994)* [46]:

1) nicht-optimierende Suchstrategie
2) einstufig optimierende Suchstrategie
3) mehrstufig optimierende Suchstrategie

Die nicht-optimierenden Suchstrategien sind dadurch gekennzeichnet, dass in jedem Iterationsschritt nicht nach einer besten, sondern lediglich nach einer funktionstüchtigen Lösung gesucht wird, d.h. nach einer Solchen, von der gesagt werden kann, dass sie die gegebenen Kriterien gerade so erfüllt. Das hört sich ein wenig nach einer Strategie für Faule an, es gibt aber genügend Entwurfsaufgaben, bei denen es durchaus nicht einfach ist, überhaupt eine gültige Lösung zu finden. Die einstufig-optimierende Suchstrategie betrachtet nur funktionierende Lösungen und wählt in jedem Schritt die jeweils Beste aus, um daraus neue Varianten zu bilden. Die mehrstufig-optimierende schließlich behält in jedem Iterationsschritt zwei- oder mehr funktionstüchtige Lösungen, um von jeder dieser Lösungen dann weitere Varianten zu erzeugen, bis ein globales Optimum erreicht ist. Diese Art der Suche ist nur selten vollständig durchführbar, wenn überhaupt, ist sie nur auf Teilprobleme anwendbar, da sie schon nach wenigen Durchläufen eine große Anzahl an Lösungen erzeugt, die bewertet werden müssen. Im günstigsten Fall verdoppeln sich die Lösungen von Schritt zur

Schritt ungefähr, wohin das führt, weiß man vom König, der den Erfinder des Schachspiels mit Reiskörnern ausbezahlen sollte. Diese Wachstumsraten sind so groß, dass auch der großzügige Einsatz von Rechenleistung nur eine gewissen Anzahl weiterer Iterationen möglich macht.

Von Entwerferseite aus ordnet man ingenieurstechnische Optimierungsverfahren mehr der 2. bis 1. Kategorie zu, während sich entwerferische Suchstrategien eher innerhalb der Kategorien 2 bis 3 bewegen. Man muss mit solchen allgemeinen Einteilungen allerdings vorsichtig sein – idealtypische Entwurfsprobleme, die sich mit einem Vorgehen entlang einer der drei Kategorien erschlagen lassen, werden sich kaum finden lassen. Dazu kommt noch eine mehr oder minder große, persönliche Färbung solcher Suchstrategien. *Rittel (1970)* [47] verweist auf beides, wenn er etwa vom *„Routinier"*, vom *„großen Meister"* [Abb. 21] spricht, der keine Probleme kennt, ein Vertreter des *Straight-Forward-Designs par excellence*, oder vom *Scanning Process* [Abb. 21], der darauf beruht, dass bewährte Lösungsstrategien zuerst versucht werden: Erst wenn man in einer Sackgasse gelandet ist, kehrt man zu einer früheren Stelle im Entwurfsprozess zurück und zieht einen neuen Lösungsstrang auf.

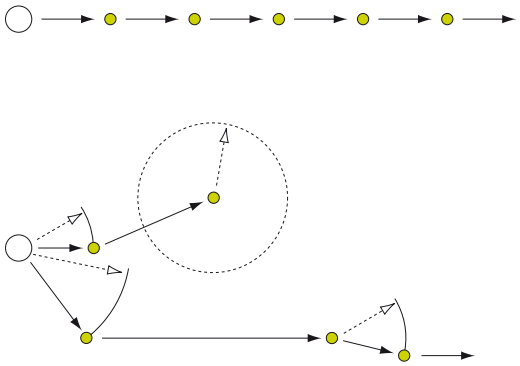


Abb. [21)] „Großer Meister" ohne Probleme sowie Scanning-Process – Suchstrategien. (nach: Rittel, 1970)

Man erkennt deutlich, dass die genannten Kategorien eher abstrakte Idealtypen sind und nicht definierte Vorgehensweisen, die auch zwingend zum Erfolg führen. Interessant ist auch der Wechsel in der Terminologie. Während bei *Becker et al (1994)* von *„mehrstufig-optimierenden Suchstrategien"* gesprochen wird, nennt *Rittel (1970)* das gleiche Verfahren *„mehrstufige Alternativenbildung"*. Hieran läßt sich ein unterschiedliches Verständnis eines Entwurfsprozesses festmachen. Der Gedanke einer optimierenden Suchstrategie geht davon aus, dass über die Analyse eines Problems zunächst

a) die Zielvorstellungen genau festgelegt werden können, dann
b) diese Vorstellungen auch in Form klarer Zielkriterien beschreibbar sind,
c) es ein geeignetes Verfahren zum Bemessen der Güte gibt (Evaluationsmethode)
d) und ebenfalls eine Methode existiert, wie auf Grundlage der Ergebnisse der Evaluation ein verbessertes Designmodell erzeugt werden kann (Synthese).

Etwas überspitzt kann man sagen, dass hier Design so verstanden wird, dass die Evaluationsergebnisse in eine Funktion einfließen, mit der ein Designmodell *M1* in ein verbessertes Modell *M1'* überführt werden kann:

Evaluation M1 -> Funktion (M1) -> M1'

Der Begriff der Alternativenbildung ist hier vorsichtiger. Er entspringt einem Entwurfsverständnis, das zunächst davon ausgeht, dass nicht immer klar ist, worauf der Entwurf eigentlich abzielt. Aber auch wenn Zielvorstellungen deutlich werden, können möglicherweise die Kriterien, die diese Ziele beschreiben, nicht genau definiert werden. Gibt es solche Zielkriterien, existiert vielleicht kein Verfahren, sie zu messen. Liegen Messergebnisse vor, bedeutet dies noch nicht, dass daraus zwangsläufig abgeleitet werden kann, wie das Designmodell verändert werden soll. Und selbst wenn die Ergebnisse eindeutig nahe legen, welche Designentscheidungen getroffen werden sollten, bedeutet dies noch nicht, dass dies tatsächlich so sein soll oder vom Entwerfer gewollt wird. Wenn dies in obige Funktionslogik überführt wird, liegt ein Evaluationsergebnis vor, das – möglicherweise – darauf Einfluss nimmt, wie ein Modell *M1* in

M1‘ überführt wird, vielleicht soll aber auch ein ganz anderer Weg beschritten werden, mit ganz anderen Modellen.

Evaluation M1 -> (?) Funktion (M1) -> (?) M1‘ -> (?) M2 -> (?) M17 …

Auch diese Darstellung mag überzogen sein, aber sie zeigt, dass Vorstellungen, wie die Zyklen von Analyse, Synthese und Evaluation in Designprozessen miteinander in Beziehung stehen, verschieden sein können. Solche Zyklen, die sich in vielen Modellen von Designprozessen wiederfinden, implizieren auf Grund ihrer Darstellungsform immer mehr oder minder stark, dass Design ein Prozess wäre, der vom Allgemeinen zur konkreten Lösung führt. Diese entspricht der Kategorie ‚Optimierung‘: Sobald eine Lösung einen gewissen Grad an Konkretheit erreicht hat, kann sie über deduktive Lösungsstrategien schrittweise in eine funktionstüchtige oder optimale Lösung überführt werden. Entsprechend klar getrennt sind auch die Abschnitte von Analyse, Synthese und Bewertung. Bei der Kategorie ‚Alternativen‘ führt der Weg dagegen mehr vom Speziellen zum Konkreten, es kommen induktive Lösungsstrategien zum Tragen, bei denen sich ein Endergebnis nicht linear ergibt, sondern sich vielmehr wie ein *Patchwork* aus verschiedenen Einflüssen zusammensetzt. Hier verschwimmen die Unterschiede und Bezüge zwischen Analyse, Synthese, Bewertung, Problemstellung und Lösung.

Diese beiden unterschiedlichen Verständnisse von Entwurfsprozessen haben disziplinäre Ursachen: Ingenieure und Konstrukteure sind mit anderen Problemstellungen konfrontiert als Designer oder Architekten. Sie entsprechen aber auch einem wachsenden Verständnis über das Wesen entwerferischer Tätigkeit. Während bei *Asimov (1962)* [48], auf den die Unterscheidung Analyse-Synthese-Evaluation zurückgeht *(vgl. Gero, 1999)* [49], ein solcher Zyklus auf den nächsten folgt [Abb. 22], fügt sich dies bei *Lawson (2006)* [50] zu einem Bild, bei dem Design ein Verhandlungsprozess zwischen Problem und Lösung ist, bei dem Analyse, Synthese und Evaluation eine Rolle spielen, aber nicht mehr in einen eindeutigen Ablaufzusammenhang gestellt werden können [Abb. 23].

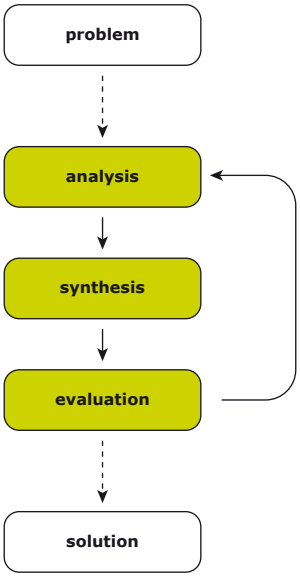


Abb. [22]) Klassische Interpretation des Analysis-Synthesis-Evaluation - Schemas als Modell für Designprozesse. (nach: Asimow, 1962)

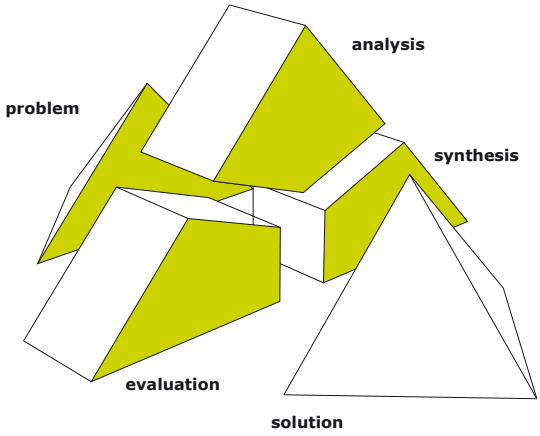


Abb. [23]) Moderne Interpretation des Analysis-Synthesis-Evaluation - Schemas als Modell für Designprozesse. (nach: Lawson, 2006)

Bei einem solchen Designverständnis gibt es keinen klar definierten Startpunkt mehr – der Entwurfsprozess kann beim Allgemeinen beginnen, er kann aber genauso gut bei einem Detail starten, um den gesamten Prozess zu informieren, oder eben irgendwo sonst, beispielsweise mit einem Optimierungsvorgang, wie bei den Entwurfsexperimenten dieser Arbeit: Es wird, flapsig gesagt, ‚in‘s Blaue hinein‘ optimiert und in der Folge versucht, aus den Optimierungsergebnissen gestalterische Ergebnisse abzuleiten: Deduktive Lösungsansätze werden in einen induktiven Zusammenhang gebracht.

5.3.2 Topologieoptimierung und Analyse von Stabtragwerken

Sowohl die Topologieoptimierung einer Fläche als auch die Analyse von Stabtragwerken sind Verfahren, die auf der Finite-Elemente-Methode (FEM) basieren. Bei dieser Methode wird ein Berechnungsproblem in eine große, aber endliche (finite) Anzahl kleiner Elemente aufgelöst und durch partielle Differentialgleichungen näherungsweise gelöst. Die FE-Methode gehört zum Standard der ingenieurstechnischen Berechnungsverfahren und ist in einer Vielzahl von Softwareanwendungen (*ANSYS®, RSTAB®, Nastran®, Unigrafics®, ProEngineer®, CATIA®*, u.a.) implementiert. Die Auflösung einer Ursprungsgeometrie kann über ein-, zwei- oder dreidimensionale, fallabhängig auch über höherdimensionale Elemente erfolgen. Für Stabtragwerke werden üblicherweise eindimensionale Elemente verwendet (Knoten-Stab-Knoten), für Schalentragwerke zwei- oder dreidimensionale.

Das Verfahren der Topologioptimierung dient dazu, die Materialverteilung einer Struktur so zu gestalten, dass in den statisch aktiven Bereichen mehr Material, in den statisch weniger oder gar nicht aktiven Bereichen entsprechend weniger oder gar keine Material vorhanden ist. Die Struktur des *D-Tower (Spuybroek, 1999)* Abb. 24, eine 12m hohe, interaktive Skulptur aus glasfaserverstärktem Kunststoff, wurde beispielsweise auf diesem Weg entwickelt: Aus einer ganzen Reihe von Berechnungen unter verschiedenen Randbedingungen (Lastfälle, Materialdicken) wurde die optimale Materialverteilung ermittelt, die sich in den einzelnen Bauteilen der Struktur gemäß dem Grad der statischen Aktivität kontinuierlich ändert. Der Wert, der eine Aussage über den Grad dieser Aktivität einer Struktur macht, ist der sogenannte Dichtewerte, in der Regel normiert von 0 bis 1: 0 bedeutet statisch nicht aktiv, 1 entspricht höchster Aktivität. Dieser Wert wird jedem einzelnen finiten Element zugeordnet. Wenn eine Topologieoptimierung in mehreren Iterationsschritten durchgeführt wird, werden die Berechnungsergebnisse des vorherigen immer mit in den aktuellen Schritt einbezogen – über die Zeit entsteht eine Art ideale statische Struktur. Dieses Verfahren nutzt *Zimmermann (2007)* [51], um bei einer entsprechend bearbeiteten Kuppelgeometrie finite Elemente über einem bestimmten Schwellenwert auszuwählen. Aus den Eckpunkten dieser Elemente wird dann über ein Voronoi-Diagramm ein Betongitterschalen-Tragwerk erzeugt Abb. 25.

Abb. 24) D-Tower: Topologieoptimierung und umgesetzte Struktur. (Spuybroek, 1999)

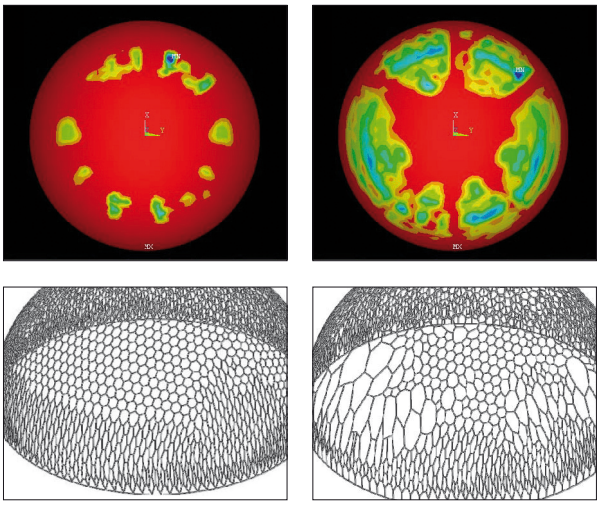


Abb. 25) Beispiele für sequentielle Topologieoptimierung (oben) und daraus abgeleitete Betongitterschalen-Tragwerke: Von grob bzw. vollflächig zu feiner bzw. optimierter Struktur. (Zimmermann, 2007)

Die beiden Beispiele interpretieren die Ergebnisse einer Topologieoptimierung also in unterschiedlicher Weise. Während beim *D-Tower* eine Differenzierung der Materialstärke über die gesamte Struktur vorgenommen wird, arbeitet das Beispiel der Betongitterschalen-Tragwerke mit dem Prinzip der Auslassung statisch weniger aktiver Bereiche,

wobei die Dimensionierung der Bauteile nicht weiter differenziert wird. Beiden Ansätzen gleich ist allerdings, dass Menge und Verteilung des Materials insgesamt von den Ergebnissen der Topologieoptimierung abgeleitet sind.

Die Methode der strukturellen Analyse von Stabtragwerken erlaubt es, sowohl Aussagen über das Gesamtverhalten einer Struktur, als auch über das einzelner Bauteile zu treffen. Ein Stabwerktragwerk kann unter bestimmten Rahmenbedinungen (Materialien, Lastfälle, Auflager, etc.) berechnet und aus den Ergebnissen beispielsweise die maximale Knoten- sowie Stabverschiebung betrachtet werden. Die maximale Knotenverschiebung darf in Abhängigkeit vom Tragwerkstyp eine bestimmte Größe nicht überschreiten – grundsätzlich gilt: Je kleiner, umso besser. Dieser Wert bezeichnet die maximale Verschiebung eines Knotens in x-, y- oder z-Richtung des Koordinatensystems oder in Richtung der Normalen zu einer Ursprungsfläche. Gleiches gilt für die maximale Stabverschiebung. Da die Knotenverschiebung bereits eine Aussage über das Verhalten der Stäbe beinhaltet (Knoten = Endpunkte der Stäbe), ist sie überwiegend dann relevant, wenn sehr lange Stäbe verwendet werden bzw. die Stablängen stark unterschiedlich sind. Dann kann es vorkommen, dass sich ein Stab an irgendeiner Stelle stärker durchbiegt als sich seine Endknoten verschieben und er dabei eventuell sogar knickt.

In dem hierzu gezeigten Beispiel Abb. 26 wurde diese Art der Betrachtung als Fitnessfunktion in einem genetischen Algorithmus dazu genutzt, um die Anordnung von Stahlringen einer großen Skulptur zu organisieren. Diese Skulptur soll zwischen zwei Hochhäusern eingespannt und von zwei Brücken durchdrungen werden. Die Besonderheit dieser Aufgabenstellung war, dass die Anmutung des Entwurfs zwar als Bild visualisiert ist: Eine 16m große Kugel, die aus einer zufällig erscheinenden Anordnung einer Vielzahl von Stahlbändern gebildet wird. Es war aber weder klar, wieviele dieser Bänder verwendet werden sollten, wie ihre Querschnitte und ihre Anordnung sein sollte noch wie die geometrische Anbindung der einzelnen Bänder untereinander beschaffen sein sollte.

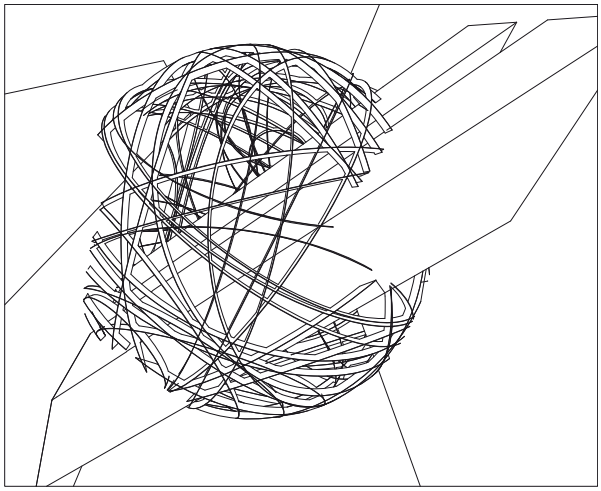


Abb. 26) WiredSphere-Projekt. Beispiel für eine Instanz eines algorithmischen Entwurfsraums, der über Methoden der evolutionären Optimierung, der Analyse von Strabtragwerken und des parametrischen Designs gebildet wurde. (Entwurf: Bellini, 2005; Programmierung: Schein; Tessmann, 2006 / 2007)

Die geforderte Lösung hier war also nicht, ein solitäres Ergebnis zu entwickeln sondern gleich einen Lösungsraum, dessen Instanzen alle bestimmten konstruktiven, geometrischen, statischen und ökonomischen Bedingungen gleichermaßen entsprachen. Erst dieser algorithmische Entwurfsraum ermöglichte es, letzendlich eine Lösung zu finden, die dem eingangs vorgestellten und visualisierten Bild am ehesten entsprach. Anders als bei den beiden Beispielen zur Topologieoptimierung konnte hier die Menge des verwendeten Materials zuvor bestimmt werden, der genetische Algorithmus sorgt zuverlässig für eine statisch günstige Verteilung der darüber definierten Bauteile.

5.3.3 Optimierung als generatives Entwurfswerkzeug

Wenn solche Optimierungsverfahren aus ihrem ursprünglichen Zusammenhang genommen werden und wie hier in den letzten beiden Beispielen gezeigt, als Generatoren dienen, um einen Designprozess zu informieren, ist sorgfältig darauf zu achten, dass die Ergebnisse nicht missinterpretiert werden, wie auch *Zimmermann (2007)* [52] in seiner Arbeit deutlich macht. Wenn beispielsweise die Verschie-

bung einer Gesamtstruktur unter einem asymmetrischen Lastfall berechnet werden soll, müssen die finiten Elemente an ihren Übergängen bestimmte Stetigkeitsbedinungen aufweisen, damit das Ergebnis aussagekräftig wird – die Wahl der Elemente, der Elementgröße sowie des *Meshing*-Verfahrens haben darauf Einfluss. Weiterhin ist zu berücksichtigen, dass eine optimierte Struktur nicht aus einer einzelnen Konfiguration, sondern einer Überlagerung verschiedener Rahmenbedingungen (z.B. Winddruck aus verschiedenen Richtungen, asymmetrische Schneelasten – also wechselnde Belastungen) ermittelt wird.

Wenn also Optimierungsergebnisse wie hier und in den folgenden Experimenten gezeigt, in konstruktive Interpretationen überführt werden, kann man zu Recht davon sprechen, dass eine Logik von Optimierung, Kraftfluss oder Spannungsverteilung mit in die Gestaltung eingeflossen ist. Man kann allerdings nicht zwangsläufig von einer statisch optimierten Struktur sprechen. Dafür müssten die Ergebnisse in einem weiteren Schritt einer klassischen statischen Überprüfung und gegebenfalls Weiterentwicklung zugeführt werden.

Oder aber der *Feedback*-Prozess zwischen Geometrie und informierenden Größen aus der Simulation physischer Kräfte wird so detailliert und aufwändig entworfen, dass im Resultat bereits ein statisch gültiges Designmodell entsteht. Genau an dieser Stelle ist das wohl größte Potential algorithmischer Entwurfsräume verortet. Es ermöglicht die entwerferisch bisher getrennt bearbeiteten Bereiche der Geometrie und der Simulation physikalischer Größen in einen algorithmischen Designprozess zusammenzufassen. Dadurch werden nicht nur verschiedene disziplinäre Grenzen herausgefordert und überwunden (Architekt – Ingenieur; Designer – Konstrukteur). Es bietet sich außerdem die Möglichkeit, unter Einbeziehung komplexer mathematischer Hilfsmittel auch differenziertere und komplexere Designmodelle zu entwickeln.

Angewandter Teil – Entwurfsexperimente

6. Themenfeld

Der experimentelle Teil dieser Arbeit entwickelt generative Entwurfswerkzeuge und Methoden, die sich mit der allgemeinen Problematik auseinandersetzen, wie aus virtuell modellierten Freiformflächen *(vgl. Glaeser, 2004)* [01] konstruktive, physisch umsetzbare Strukturen abgeleitet werden können. Im Folgenden werden aus praktischer Sicht die wichtigsten Aspekte dieses Problemfelds skizziert und für die hier bearbeiteten Themen eingegrenzt.

6.1 Konstruieren und Modellieren virtueller Formen

6.1.1 Konstruieren von Form

Mit den meisten *CAD*-Systemen ist es problemlos möglich, zweisinnig gekrümmte, komplexe Flächenformen zu modellieren. ‚Modellieren' als virtuelles Ziehen, Zerren, Kneten, Drücken und im Gegensatz zu ‚Konstruieren' – vom Lateinischen *con struere*: Aufbauen, Zusammenfügen. Der konstruktive Umgang mit Formen ist das, was in industriellen Entwurfskontexten seit langer Zeit gewohnt ist und in Funktionsumfang, Organisation und Struktur von *CAD*-Anwendungen seine Entsprechung gefunden hat. Solche Programme stellen eine Reihe vorgefertigter geometrischer Elemente bereit, wie Linie, Rechteck, Kreis und darauf aufbauend, im Bereich der Flächen- und Flächenkörper, etwa Planen, Zylinder, Kegelstümpfe, Quader. Aus Instanzen solcher und anderer geometrischer Objekte werden in sequentiellen Schritten allmählich komplexere Formen aufgebaut. Diese Schritte beinhalten typischerweise das ständige Hinzufügen weiterer Instanzen geometrischer Objekte, die durch eine Reihe unterschiedlicher Operationen verändert und räumlich organisiert werden. Hierzu zählt beispielsweise die Anwendung der *Bool'schen* Operationen um aus *Solids* durch Addition, Subtraktion oder das Bilden von Schnitt- oder Teilmengen neue Formen zu definieren. Dazu kommen verschiedene erzeugende Operationen für Rotations- oder Extrusionskörper, Transformationsoperationen wie Skalieren oder Verzerren, Befehle um erzeugte Instanzen zu ordnen und dergleichen mehr. Unabhängig davon, welche Operationen angewendet werden und wie detailliert ein virtuelles Designmodell auch konstruiert wird, bleibt ein Charakteristikum solcher Modelle immer erhalten: Es gibt diskrete Punkte, die für eine Orientierung genutzt werden können, wie bei dem einfachen Beispiel einer Linie, die an genau einer Stelle in einen Kreisbogen mündet [Abb.01].

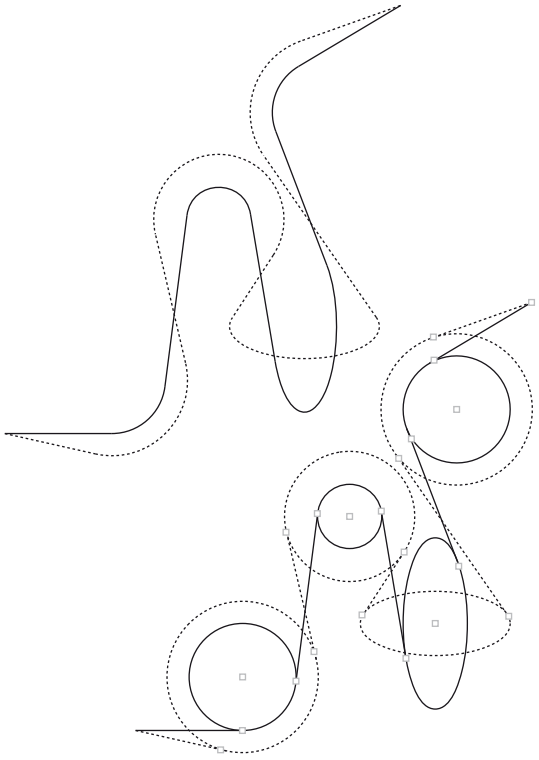


Abb. [01]) Eine über ihre Radien variierte Kurve mit diskreten Orientierungspunkten.

01) „Eine Fläche kann als Erzeugnis einer Linie gesehen werden, die sich durch den Raum bewegt. Wird eine gekrümmte Linie mit kontinuierlich wechselndem Radius entlang einer anderen Linie, ebenfalls mit kontinuierlich wechselndem Radius geführt, entsteht eine Fläche mit lokal unterschiedlichen Eigenschaften. Solche Flächen sind nicht mehr eindeutig elliptisch, hyperbolisch oder parabolisch sondern können mehrere solcher Eigenschaften aufweisen. Diese Art von Flächen werden als Freiformflächen bezeichnet." (Glaeser, 2004)

‚Orientierung' wird dabei in dreifacher Bedeutung angesprochen:

1) Geometrische Ordnung. Die diskreten Stellen sind zunächst wertvolle Hilfen, um bei der Konstruktion eines virtuellen Modells Proportionen abzuschätzen, räumliche Bezüge nachzuvollziehen und zu organisieren. Die additive Logik, durch die eine zunehmende geometrische Detaillierung erreicht wird, gibt dabei schon eine Richtung vor, an der sich die Entwicklung eines Designmodells ausrichtet – vom Einfachen zum Komplexen, wobei die zunehmende Komplexität eines Modells in aller Regel aus den Bedingungen einiger weniger geometrischer Instanzen entwickelt wird.

2) Physische Bezüge. Durch den konstruktiven Umgang mit geometrischen Elementen sind in gewisser Weise auch schon Vorstellungen von Material, Statik und Fertigung repräsentiert. Ein Rotationskörper etwa kann direkt als Drehteil, ein Extrusionskörper als Strangpressprofil interpretiert werden. Die Art der räumlichen Organisation geometrischer Instanzen kann bereits auf ein statisches Modell verweisen. Durch beides zusammen verdichtet sich auch eine Idee von Materialität oder umgekehrt, das gedachte Material eines Objekts legt gewisse statische Modelle und Fertigungsverfahren nahe und damit auch die Verwendung entsprechender geometrischer Elemente. Auf der Seite der Analyse steht dann wieder eine ganze Reihe verschiedener *CAD-Tools* bereit, mit denen sich prüfen läßt, ob die materielle Hypothese, die durch Geometrie ausgedrückt wurde, korrekt ist: Kollisionsprüfungen, Berechnung von Materialspannungen und - verformungen und Ähnliches.

3) Gestalterisches Leitbild. Die gedankliche Verbindung von Geometrie und Materialität und ihre instrumentelle Verknüpfung durch *CAD*-Systeme repräsentieren damit grundlegende Bedingungen, die prägend wirken, wenn Form mit Inhalt assoziiert wird, wenn also Produkte oder Architekturen gestaltet werden. Vereinfacht gesagt entspricht der konstruktive Umgang mit Formen damit einer Wissenstradition, einer Art kollektiver Entwurfsphilosophie, die ausgeht von der Komposition einfacher Grundformen [Abb. 02], wie sie schon die Architektur der Antike benutzt und die sich bis in die Gegenwart fortgesetzt hat: Es ist eine bewährte Entwurfstradition, aber auch ein selbstreferenzierendes System, in intellektueller und ästhetischer Hinsicht, wie bereits in *Kapitel 2.4.2* angesprochen.

Abb. [02]) Protoypen konstruktiver Entwürfe. Der Ulmer Hocker von Bill und Gugelot (1954) sowie das Smith-House von Meier (1966).

6.1.2 Modellieren von Form

Als Erweiterung des konstruktiven Umgangs mit Formen steht das Arbeiten mit Freiformkurven und -flächen, die in fast allen *CAD*-Anwendungen über die sogenannten *NURBS (Non-Uniform Rational B-Splines)* beschrieben und modelliert werden. Erweiterung deshalb, weil mit Hilfe von *NURBS* auf der einen Seite komplexe, weich und amorph anmutende, naturhafte Formen modelliert werden können, sie aber auf der anderen Seite auch das gewohnte Formenspektrum mit beinhalten. Eine Polylinie etwa stellt lediglich einen Sonderfall in den Parametern der mathematischen Beschreibung einer *NURBS*-Kurve dar *(vgl. Glaeser, 2005)* [02]. Dadurch ist es möglich, kontinuierliche Transformationen in

einem sehr großen Spektrum unterschiedlicher Formen vorzunehmen: Eine einfache Rechtecksfläche kann so schrittweise in eine völlig zerbeult scheinende Form überführt werden oder *vice versa* Abb. 03.

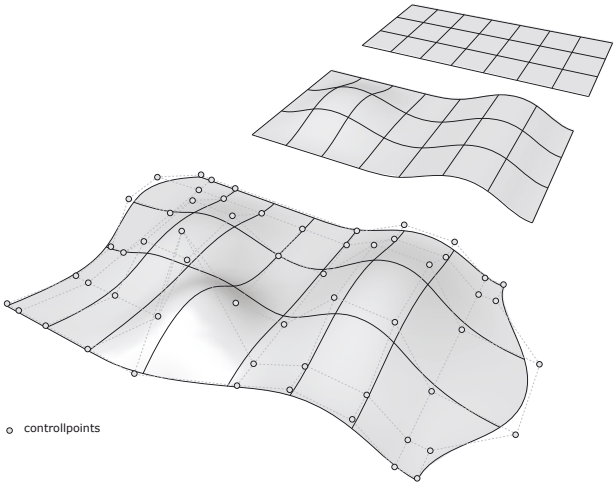


Abb. 03) Wesentlich für Freiformflächen ist, dass „(…) deren Aussehen nahezu beliebig variiert werden kann, indem gewisse Parameter und erzeugende geometrische Elemente variiert werden." (Glaeser, 2005) 02

Die Form einer *NURBS*-Kurve wird im *CAD*-System über Kontrollpunkte, Gewichtungen, die diesen Kontrollpunkten zugeordnet sind, Knoten sowie den Grad der Kurve definiert *(controll points, knots, weights, degree)* Abb.04. Die Kontrollpunkte entsprechen den Ecken der Polygon-Züge, über die eine *NURBS*-Kurve repräsentiert wird. Positionen und Anzahl dieser Kontrollpunkte, die bis auf den Anfangs- und Endpunkt nicht auf der Kurve liegen müssen, bestimmen somit den groben Rahmen des Kurvenverlaufs. Die genau Ausprägung der Kurvenform errechnet sich aus den Verhältnissen der Polynom-Funktionen die mit den Kontrollpunkten assoziiert sind. Das Gewicht eines jeden Kontrollpunkts legt fest, wieviel Einfluss ein Kontrollpunkt über seinen Kurvenabschnitt hat: Je höher das Gewicht, desto näher wird der entsprechende Kurvenabschnitt an den Kontrollpunkt gezogen. Der Grad einer *NURBS*-Kurve, der höchste Exponent in der Polynominal-Funktion der Kurve, ist ein weiterer formgebender Parameter: Der Grad entspricht der Anzahl der Segmente des Polygons, die einen Kurvenabschnitt zwischen zwei Knoten definieren. Je höher der Grad, umso feiner kann eine Kurve auch modelliert werden. Ist der Grad 1, besteht die ‚Kurve' aus Geraden, die durch die Kontrollpunkte gehen. Die Knoten schließlich, die in Abhängigkeit vom Kurvengrad und der gesamten Anzahl der Kontrollpunkte bestimmten Kontrollpunkten zugeordnet sind, bestimmten dadurch über den Anteil an der Kurve, den ein Kontrollpunkt beeinflusst. *NURBS*-Flächen werden entsprechend durch ein Netz aus Kontrollpunkten und Knoten repräsentiert.

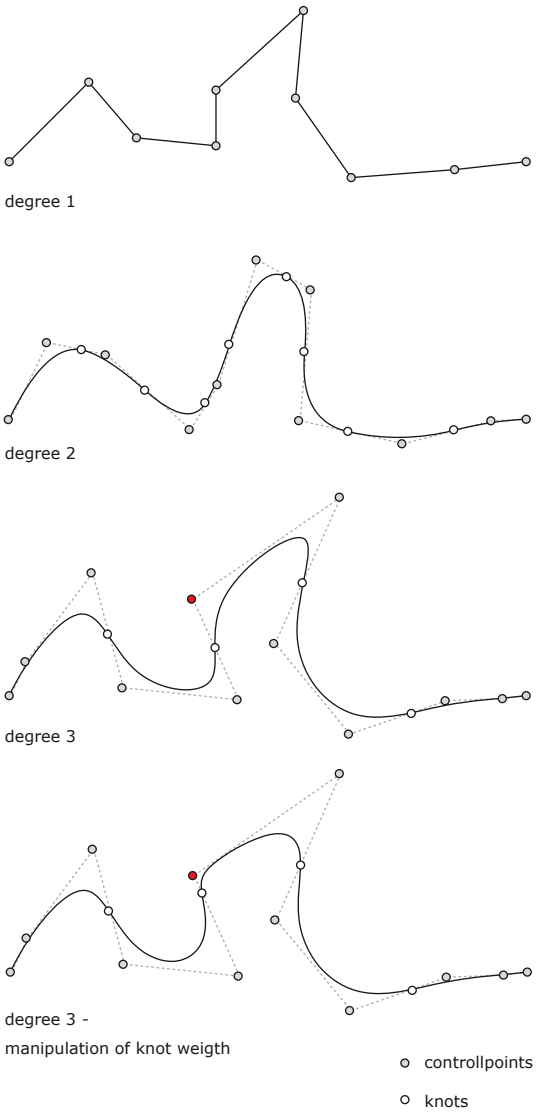


Abb. 04) Parametrik von NURBS-Kurven, Beispiele.

Sie sind damit einerseits ein äußerst flexibles sowie sehr präzises Mittel, um komplexe Formen zu modellieren. Andererseits überschreiten sie aber auch das eben beschriebene, tradierte und bewährte Orientierungssystem, das dem konstruktiven Umgang mit Formen innewohnt Abb. 05.

1) Geometrische Komplexität. *NURBS*-Flächen sind, genauso wie eine Linie, ein Kreis oder ein Würfel, geometrische Elemente, von denen Instanzen erzeugt werden. Durch die Vielzahl formerzeugender Parameter ist aber schon jede Instanz für sich von einer hohen Komplexität. Diese große Zahl an Parametern, die weichen Übergänge, die kontinuierlich sich verändernden Krümmungen machen es schwer, sich beim Modellieren von freien Formen zu orientieren. Es gibt kaum diskrete Stellen, an denen sich die Wahrnehmung ‚festhalten' kann. Sobald *NURBS*-Flächen in geometrische Zusammenhänge eingebunden werden, wird es sehr schwierig und aufwändig, Änderungen an einer Fläche auf verknüpfte Instanzen zu übertragen. Dies ist ein Grund dafür, dass solche Flächen oft als separate Elemente behandelt werden, die im Modellierungsprozess erst spät eine Detaillierung erfahren.

2) Abstraktion vom Materiellen. Freiformen wecken Assoziationen zur Formenproduktion der Natur, im handwerklichen Kontext ist ein Wissen über die Herstellung solcher Formen erhalten geblieben. In aller Regel werden sie heute mit Hilfe rechnergesteuerter Prozesse realisiert. Dabei entsprechen weder biologische Wachstumsprozesse, noch die handwerklichen oder *CNC*-Fertigungsprozesse in ihrer Logik dem Modellierungsprozess, mit dem komplexe, zweisinnig gekrümmte Geometrien im *CAD*-System erzeugt und geformt werden. Insofern ist es hier schwieriger, direkte Bezüge zu Materialeigenschaften, statischen Modellen und Fertigungsverfahren herzustellen: Das Arbeiten mit *NURBS*-Modellen beinhaltet eine gewisse Abstraktion vom materiellen Kontext.

3) Gestalterische Neuorientierung. Man kann im Zusammenhang von Freiformflächen also davon sprechen, dass die gedankliche und instrumentelle Verbindung von Geometrie, Material, Statik und Fertigung bei weitem noch nicht so selbstverständlich ist, wie beim konstruktiven Umgang mit den gewohnten geometrischen Elementen. Es gibt zwar im Bereich des Produktdesign schon eine längere Tradition im Umgang mit solchen Formen. Das hier etablierte Wissen steht aber im Zusammenhang mit großen Stückzahlen und kleiner Bauteilgröße. Ein gestalterischer Umgang mit größer dimensionierten Freiformflächen wird in vielen verschiedenen Ansätzen experimentiert, die sich erst noch zu selbstverständlichen Entwurfshaltungen bündeln werden. Deutlich ist aber bereits, dass generative Verfahren dabei eine sehr wichtige Rolle übernehmen, um die große Menge an unterschiedlichen Informationen (formgebende Parameter) zu bearbeiten.

Abb. [05]) Beispiele für Entwürfe mit Freiformflächen. Hara von Gurioli (2003) sowie das Kunsthaus in Graz von Cook und Fournier (2003).

6.2 Problembeschreibung – Vom Virtuellen zum Materiellen

6.2.1 Struktur zu Form, Form zu Struktur

Ein generativer Prozess, der in einer frei geformten, physisch umgesetzten Struktur mündet, kann im Wesentlichen als ein Prozess der Informationsanreicherung betrachtet werden, bei dem die geometrischen Bedingungen von Formen mit den Bedingungen, die sich aus Materialeigenschaften, strukturellen Anforderungen und Fertigungsverfahren ergeben, verhandelt werden, um ein detailliertes

Designmodell zu erzeugen. Unter den verschiedenen generativen Ansätzen, die hier Verwendung finden, lassen sich zwei Grundtypen ausmachen.

Typ 1 – Über die Struktur zur Form – benutzt unterschiedliche Algorithmen, um aus den Bedingungen einer Struktur Formen abzuleiten. Dabei können eine Vielzahl einzelner geometrische Instanzen zu einer konstruktiven Struktur entwickelt werden, ein Beispiel hierfür ist der *Tiling*-Algorithmus, den *Aranda und Lasch (2006)* [03] für ihr *Grotto*-Projekt [Abb. 06] benutzen: Hier wird mit Hilfe des Voronoi-Diagramms eine Punktewolke erzeugt und aus daraus entwickelten Körpern ein höhlenähnliches Gebilde zusammengesetzt. Dieser Prozess kann aber auch so durchgeführt werden, dass nur eine Großform nach bestimmten Regeln erzeugt wird und daraus dann eine detaillierte konkrete Umsetzung abgeleitet wird. Die *10 Mile Spiral* [Abb. 07], ebenfalls von *Aranda und Lasch (2006)* [04], ist ein Beispiel für diese Art generativer Ansätze. Bei beiden Beispielen ist die Idee einer räumlichen Struktur bereits im generativen, formerzeugenden Prinzip enthalten.

Abb. [06]) Tiling – Grotto-Projekt von Aranda / Lasch (2005).

Abb. [07]) Spiraling – 10 Mile Spiral-Projekt von Aranda / Lasch (2004).

Typ 2 – Von der Form zur Struktur – kann als ein Prozess der nachträglichen Rationalisierung verstanden werden. Hier liegt als Ausgangspunkt eine von Hand modellierte, virtuelle Flächen als Großform vor, die dann nach bestimmten Regeln unterteilt und in ein konstruktives System überführt wird. Hier kommen häufig Algorithmen zur Flächentriangulation zum Einsatz, auch das *Slicen*, das Aufteilen einer Fläche durch planare Ebenen, wird oft angewendet, wie beim *Loewy Bookshop* von *Jakob* und *McFarlane (2001)* [Abb. 08] oder der *wetGrid-Installation* im Kunstmuseum von Nantes *(Spuybroek, 2000)* [Abb. 09] zu sehen.

Abb. [08]) Slicing – Loewy Bookshop von Jakob und McFarlane (2001).

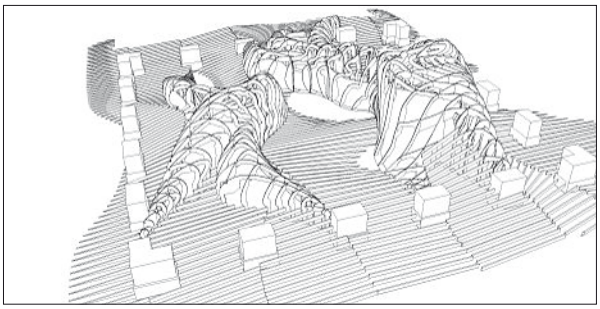


Abb. [09]) Slicing – wetGrid-Installation im Kunstmuseum von Nantes. (Spuybroek, 2000).

Natürlich können und werden auch beide Typen von Ansätzen in vielfältiger Weise miteinander kombiniert, diese Arbeit betrachtet aber lediglich generative Verfahren, die dem zweiten Typ zugeordnet werden können. Für die Entwurfsexperimente wurden daraus folgende *Constraints* als Ausgangspunkte festgelegt:

1) Die Modellierung einer Fläche erfolgt auf konventionelle Weise, per Hand, über ein *CAD*-System.

2) Die Entwicklung einer räumlichen Struktur führt von der Fläche zur konstruktiven Umsetzung, nicht umgekehrt.
3) Die Ableitung einer konstruktiven Umsetzung basiert auf einer einzelnen Fläche.
4) Eine Fläche wird dabei als abstrakte Großform begriffen und nicht als konkrete Form eines Bauteils.

6.2.2 Von der virtuellen Fläche zur Structural Shape

Diese Einschränkungen erlauben es, den Prozess der baulichen Umsetzung einer freien Form als eine Transformation von einer virtuell modellierten Fläche zu einer *Structural Shape* zu begreifen. Das heißt, das symbolische Modell einer Fläche, die dreidimensional im virtuellen Raum aufgespannt ist, aber über keine Dicke verfügt, wird mit zusätzlichen Informationen versehen, um daraus eine strukturierte Bauform zu entwickeln. Bei diesem Prozess ist eine Balance zweier Aspekte zu finden, wie in den Arbeiten *Frei Ottos* Abb. 10 in vielfältiger Weise demonstriert:

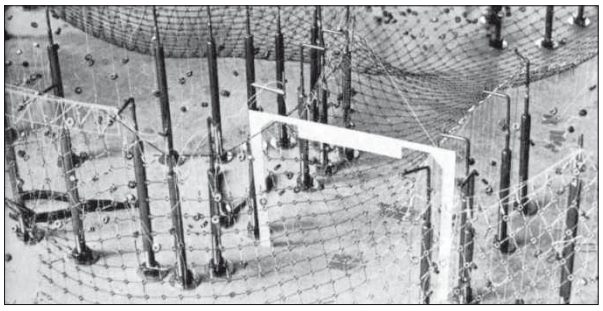


Abb. 10) Multihalle in Mannheim (Otto, 1974): Hängemodell (unten) und Innenansicht während der Aufbauphase – Einklang von Geometrie und Mechanik.

Dem mechanischen Verhalten des Konstruktionssystems, das eine Fläche interpretiert, sowie der Geometrie der Ursprungsfläche und der daraus abgeleiteten Bauteile: *„In general, a structural shape consists of it's geometry and it's mechanical behavior. However, the interrelationship between them is inseperable; if a structure's geometry is changed, it's mechanical behavior will also change." (Sasaki, 2004)* [05]

Bei der Verwendung generativer Verfahren werden die zusätzlichen Informationen, die nötig sind, um eine Ausgangsfläche in eine baubare Form zu überführen, nicht direkt über *CAD*-Modelle entwickelt und visualisiert, sondern über Computerprogramme hinzugefügt und organisiert. Ein wesentlicher erster Schritt, der hier in fast allen Fällen enthalten ist, besteht in der Anwendung eines Elementierungsverfahrens, nach der eine Ursprungsfläche in kleinere Elemente zerlegt wird, z.B. durch das schon erwähnte *Slicen*, durch die Zerlegung einer Fläche in abwickelbare Flächen, durch Triangulation oder Quadrangulation, also das Unterteilen einer Fläche in Drei- oder Viereckmaschen. Schon dieser Schritt setzt wechselweise geometrische Bedingungen: Nicht jeder Flächentyp läßt sich durch jede Art von Aufteilungsverfahren bearbeiten.

Aus den geometrischen Parametern der erzeugten Flächenelemente werden die einzelnen Bestandteile einer räumlichen Struktur detailliert entwickelt. Spätestens bei diesem Transformationsschritt sind die geometrischen Bedingungen mit denen in Einklang zu bringen, die sich aus den Materialeigenschaften einzelner Bauteile, dem Verhalten der gesamten Struktur und den gewählten Fertigungsverfahren ergeben. Die *Constraints* der materiellen Welt sind aber nicht nur maßgebend für die konkrete Form von Bauteilen, sondern können darüber hinaus bereits einen Einfluss auf die Elementierungssystematik und dadurch wieder auf die Form der Ausgangsfläche haben. Die vielfältigen Wechselwirkungen, die hier entstehen können, sollen hier nicht systematisiert, sondern an Hand der nachfolgenden Beispiele illustriert werden.

6.3 Beispiele

6.3.1 Structural Shapes in der Praxis

Dass größere Flächen bei ihrer Realisierung direkt, ohne größere Abweichung von der Ursprungsform abgebildet werden, ist eher die Ausnahme. Häufig werden sie in kleinere Elemente aufgelöst, die in vielen Fällen wirtschaftlich günstiger herstellbar sind. Ein limitierender Faktor ist hierbei einmal die Stückzahl identischer Teile, die darüber bestimmt, ab wann sich die Fertigungskosten, beispielsweise für das Erstellen einer Produktionsform, wieder amortisieren. Eine weitere Faktor ist natürlich die Größe eines Bauteils, die nicht nur ökonomisch, sondern auch durch die Verwendung bestimmter Fertigungstechniken und Maschinen begrenzt ist. Und schließlich spielt auch die geometrische Komplexität eines Bauteils und seine konkrete Form eine große Rolle, sowohl aus ökonomischer Sicht als auch bezüglich der prinzipiellen Machbarkeit. Die Sitzschale des *Rocking-Chair von Eames (1950)* Abb. 11 ist ein Beispiel für ein schon relativ großes und komplexes Freiform-Bauteil. Die ursprüngliche Variante aus glasfaserverstärktem Kunststoff ist heute nicht mehr wirtschaftlich herstellbar, so dass sie in der aktuellen Fassung von Vitra aus Polypropylen gefertigt wird. Das Entwurfskonzept – eine Schale für verschiedene Sitzgestelle – sorgt für größere Stückzahlen und damit eine ökonomisch günstige Perspektive in konventionellen Herstellungsprozessen.

Abb. 11) Der Rocking-Arm-Chair in der von Vitra produzierten Fassung mit einer Sitzschale aus Polypropylen. (Charles and Ray Eames, 1950)

Die beiden gezeigten Entwürfe von *MGX* Abb. 12 kommen ebenfalls noch ohne eine Elementierung aus, allerdings sind die Formen hier so komplex, dass Standard-Herstellungsverfahren kaum mehr in Betracht kommen. Die Idee der *Mass Customization*, die Bestandteil der Entwurfsphilosophie vom *MGX* ist, macht den Gedanken an konventionelle Fertigungstechniken vollends obsolet. Beide Entwürfe sind mit Hilfe von *Rapid-Prototyping*-Technologien wie Laser-Sintern oder Stereolitografie hergestellt. Limitiert sind solche 3D-Verfahren einmal durch die relativ hohen Materialkosten sowie die recht langen Bauzeiten, so dass sie (zur Zeit) trotz der technischen Möglichkeiten realistischerweise kaum für Bauteile in Frage kommen, die etwa Stuhlgröße überschreiten.

Abb. 12) Eine Leuchte und eine Vase, hergestellt mit Rapid-Manufacturing-Technologien. (Materialise.MGX, 2006)

Eine einfache Art, eine Freiformfläche zu zerlegen, ist sie durch planare Ebenen zu schneiden. Bei dieser Art der Flächenelementierung entstehen Elemente, deren Aussenkonturen exakt der Form der Ursprungsfläche folgen. Die Anmutung des Ergebnisses ist maßgeblich von der Anzahl Schnitte, der weiteren Interpretation dieser Schnitte sowie dem Maßstab der Ursprungsform abhängig. Beim Beispiel des *BMW-Bubble* von *Bernhard Franken* Abb. 13, wird die Ursprungsform, die von zwei ineinander fließenden Wassertropfen abgeleitet ist, durch orthogonale Schnitte zerlegt. Diese Schnitte sind außen als Fugenbild der zweisinnig gekrümmten Acrylglasflächen zu sehen, die die Haut der Form bilden – jede Scheibe ist ein Unikat und entsprechend aufwändig durch thermische Verformung über eigens gefräste Formen hergestellt. Entlang der Schnitte entwickelt sich

nach innen hin die Tragstruktur des Pavilions, deren einzelne Bauteile aus Aluminiumplatten über Wasserstrahlschneiden hergestellt wurden. Beim *metropol parasol*, ein Wettbewerbsbeitrag für die Gestaltung der *Plaza de la Encarnaçion* in Sevilla von *Jürgen Mayer (2004)* [Abb. 14], wurde eine ähnliche Elementierungstechnik verwendet. Die Basis des Entwurfs sind sechs enorm große ‚Sonnenschirme', die begehbar sein sollen und in die ein Restaurant integriert werden wird. Die gesamte Struktur dieser Schirme wird aus planaren Holzplatten hergestellt, auch dies ist, wie beim *BMW-Bubble*, nur noch über *CNC*-Verfahren möglich.

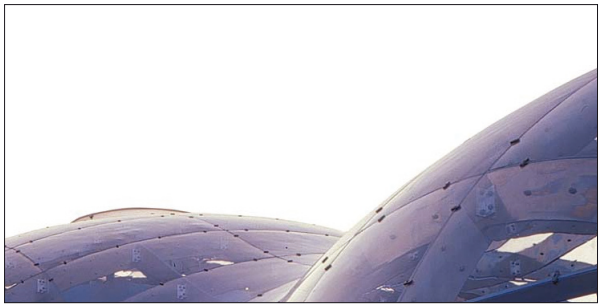


Abb. [13]) Ausschnitt des BMW-Bubbles, ein Ausstellungspavilion für die IAA. (Franken, 1999)

Abb. [14]) metropol parasol für die Plaza de la Encarnaçion in Sevilla. (Mayer, 2004)

Einsinnig gekrümmte Flächen lassen sich durch planare Vierecke (Rechtecke oder Trapeze) abbilden, ebenso wie rotationssymmetrische Kuppeln (durch Trapeze, außer im Pol). Dies hat den Vorteil, dass die flächigen Elemente durch die (gegenüber den 3D-*CNC*-Verfahren) kostengünstigeren 2D-*CNC*-Verfahren herstellbar sind. Je nach Ausgangsgeometrie können auch identische Bauteile für serielle Herstellungsprozesse entstehen. Einsinnig gekrümmte Elemente lassen sich auch verwenden, um zwischen zwei geeigneten Raumkurven zu vermitteln, ein typisches Markenzeichen vieler Projekte *Gehrys*, hier gezeigt am Beispiel der *Walt Disney Concert Hall* [Abb. 15]. Die einzelnen Elemente lassen sich dabei ohne Stauchung und Dehnung abwickeln und werden durch die Ränder der Ursprungsfläche beschnitten. Tori lassen sich durch planare, viereckige Flächen aufteilen. Segmente aus Tori kann man wiederum verwenden, um Flächen zu erzeugen, die sich in ihrer Anmutung der frei geformter Flächen annähern. Abweichungen von einer Ursprungsfläche kommen dabei immer dann zu Stande, wenn diese kontinuierliche Krümmungswechsel aufweist, was durch Torus-Segmente nicht abbildbar ist. Diese Technik der Flächenunterteilung ist von *Foster and Partners* entwickelt worden, das Beispiel zeigt *The Sage, Gateshead* [Abb. 16].

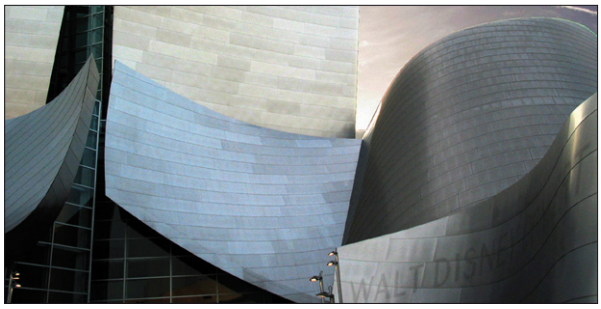


Abb. [15]) Walt Disney Concert Hall (Ausschnitt). (Gehry, 2003)

Abb. [16]) The Sage, Gateshead. (Foster and Partners, 2004)

Eine weitere Möglichkeit ist Kuppeln (oder Ellipsoide) durch planare Vielecke aufzuteilen. Berühmt sind die geodätischen Kuppeln, die *Richard Buckminister Fuller* seit den 1940er Jahren weiter entwickelt hat. Solche Kuppeln entstehen über geometrische Transformationen aus Dodekaedern oder Ikosaedern. Das Beispiel hierfür zeigt einen Entwurf

von *Grimshaw*, bei dem durch Sechsecke abgebildete Kuppeln miteinander verschnitten sind [Abb. 17].

Abb. [17]) Gewächshaus des Eden Projekts, Cornwall. (Grimshaw, 1997)

Die Aufteilung einer Fläche in planare Einheiten ist dann nötig, wenn das verwendete Material nicht gekrümmt herstellbar ist oder dies zu teuer wäre. Triangulation – also die Aufteilung einer Fläche in Dreiecksmaschen, ist die einzige Möglichkeit, eine Fläche unabhängig von ihrer Form so zu unterteilen, dass immer planare Elemente entstehen. *„In der Regel lassen sich solche Flächen [Freiformflächen] nicht mehr mit ebenen Viereccksscheiben belegen, sodass man genötigt ist, die wirtschaftlich und konstruktiv ungünstigeren Dreiecksscheiben zu verwenden." (Schlaich et al, 2002)* [06]

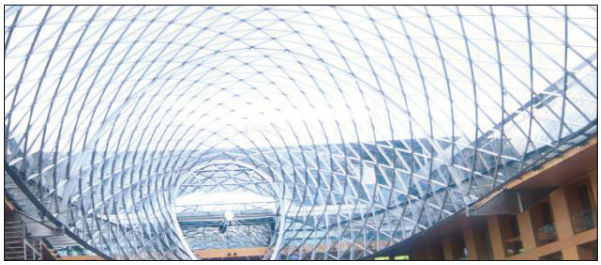


Abb. 18/19) Zwei Beispiel für Glasüberdachungen, die über Flächentriangulation erzeugt wurden. Oben: Atrium-Überdachung DG-Bank, Berlin (Gehry, 1997) sowie die Überdachung für die Mailänder Messe (Fuksas, 2004): Während sich die Glaskonstruktion des Gehry-Entwurfs selbst trägt, benötigt logo & vela eine Reihe von ‚Krücken' (zusätzliche Stützen).

Die hierzu angeführten Beispiele sind zwei Glas-Stahl Bauten, einmal die skulptural ausgeformte und frei tragende Innenhofüberdachung der DG-Bank am Pariser Platz in Berlin [Abb. 18], und zum anderen die Glasüberdachung *logo & vela* der Mailänder Messe [Abb. 19], die mit rund 50.000 m^2 Dachfläche, rund 41.800 Stäben und 18.000 Knoten und einer komplexen, zweisinnig gekrümmten Form wohl eines der ambitioniertesten Projekte ist, das solche Strukturen benutzt. Dabei geht es selbstverständlich nicht nur darum, eine möglichst wirtschaftlich günstige Triangulierung einer Fläche vorzunehmen, sondern gleichzeitig auch eine Anordnung des Netzes zu finden, die entlang der Kraftflüsse orientiert ist.

Dreiecke in einem ganz anderen Kontext werden beim *Federation Square* - Projekt [Abb. 20] von *Lab-Architects* verwendet: Hier wir das *„pinwheel" (Radin, 1994)* [07] benutzt, eine Methode zur nicht-periodischen Flächenaufteilung, um Flächen in einzelne Dreieckselemente zu zerlegen und (teilweise) wieder zu unregelmäßigen Vielecken zusammenzufassen. Diese Elementierung ist offensichtlich an der Fassade des Gebäudes, findet sich aber teilweise auch in seiner Tragstruktur wieder.

Abb. [20]) Federation Square, Melbourne – eine Methode der Ebenenparkettierung als prägendes Element. (Lab Architects, 1997)

Bei Translationsflächen entsteht die Fläche, indem eine Erzeugende (Profilkurve) entlang einer Leitlinie verschoben wird. Die Profilkurve kann durch parallele Vektoren vereinfacht abgebildet werden. Diese Vektoren repräsentieren die Längskanten der Vierecke, die Verbindungen zwischen den beiden Start- bzw. Endpunkten der Vektoren die

Querkanten. Dadurch, dass die Vektoren verschiedene Längen und Steigungen (orientiert an der Leitlinie) haben können, sind somit auch relativ freie, zweisinnig gekrümmte Geometrien abbildbar, sofern sie als Translationsflächen erzeugt wurden – *„Den Dreh mit den Geraden"* nennt *Schober (2002)* [08] dieses von ihm entwickelte Verfahren. Die Glas-Stahl-Konstruktion der Überdachung des Flusspferdehauses in Berlin von *Gribl* ist nach diesem Prinzip umgesetzt [Abb. 21].

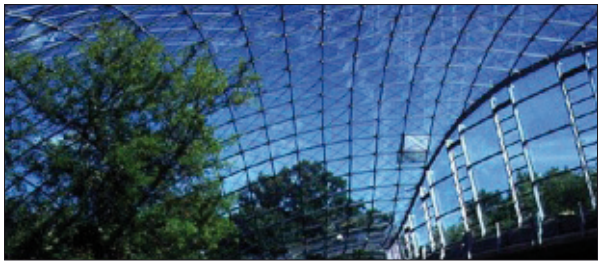


Abb. [21]) Flusspferdehaus Berlin Zoo. (Gribl, 1996)

Einen Flächentyp, der sich ebenfalls durch planare Elemente abbilden läßt, verwenden *dECOi (1995)* in ihrem *Ether/1* - Projekt [Abb. 22], eine Skulptur, die zum 50jährigen Bestehen der vereinten Nationen in Genf realisiert wurde. Hier wurde die Form der Ausgangsfläche aus eingefrorenen Bewegungssequenzen von Tänzern erzeugt, die *Forsythes* Quintett interpretieren. Jede Bewegungssequenz entspricht einem vertikal ausgerichteten Querschnitt. Werden diese Querschnitte durch eine *Loft*-Operation in eine Fläche überführt, bei der die einzelnen Segmente zwischen den Querschnittskurven gerade sind, können diese Flächensegmente ohne Stauchung oder Dehnung abgewickelt werden. Im gezeigten Beispiel von *Ether/1* wurden diese Segmente allerdings nicht flächig, sondern über einzelne Aluminiumstäbe interpretiert.

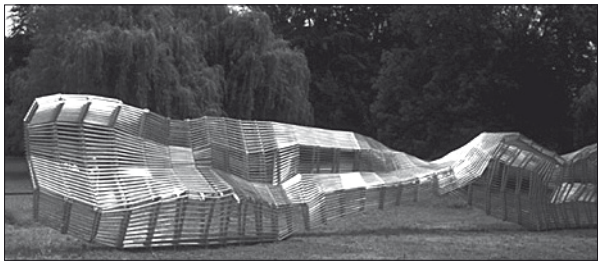


Abb. [22]) Ether/1 – Skulptur zum 50jährigen Bestehen der UN. (dECOi, 1995)

Das Problem, eine beliebige Freiformfläche in abwickelbare Bänder zu zerlegen, untersucht auch *Kilian (o.A.)* [09] in einem Experiment auf rein geometrischer Ebene. Er entwickelt dabei einen Ansatz, bei dem eine *NURBS*-Fläche durch einen genetischen Algorithmus dahingehend evaluiert wird, dass möglichst große Bereiche der Fläche in abwickelbare Elemente *(developable stripes)* unterteilt werden können [Abb. 23].

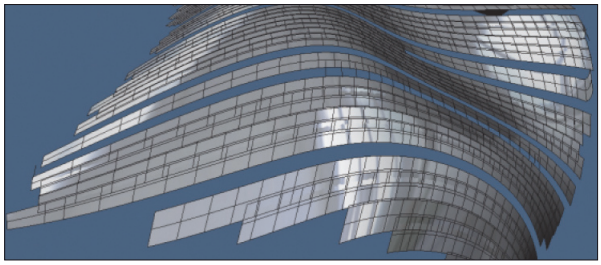


Abb. [23]) Abwickelbare Bänder mit Hilfe eines GA aus einer Freiformfläche erzeugt. (Killian, o.A.)

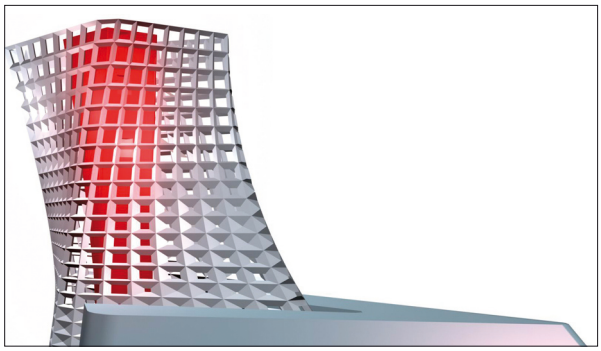


Abb. [24]) augmentedFRAME. (f-u-r, 2004)

Ebenfalls auf unterschiedliche Freiformflächen anwendbar ist der geometrische Kniff, den *Becker* und *Tessman* in ihrem *augmentedFRAME* - Projekt [Abb. 24] entwickeln, ein Wettbewerbsbeitrag für einen *Landmark-Tower* anläßlich der Pariser Olympiabewerbung für 2012. Hier wird eine Fläche in einem ersten Schritt in ein Viereckraster zerlegt. Dann wird jede einzelne der resultierenden Viereckmaschen durch jeweils vier Trapeze nach außen und jeweils vier nach innen abgebildet. Jedes Trapez ist dabei planar, da es immer auf einer Ebene liegt, die über je zwei Eckpunkte der Viereckmaschen, die entlang der Normalen der Ursprungsfläche nach aussen verschoben sind, sowie dem zugehörigen

Mittelpunkt einer Viereckmasche aufgespannt ist. Durch diesen Trick ist es möglich, alle Bauteile der Struktur durch flächige Elemente zu erzeugen, im gezeigten Beispiel sind gefaltete Blechelemente vorgesehen. Der Verlauf, der auf der so erzeugten Flächenstruktur durch die unterschiedlichen Öffnungen sichtbar wird, ist über die Ergebnisse einer Topologieanalyse erzeugt: Je höher die lokale Belastung an einer Viereckmasche, umso weiter schließen sich die Trapezbleche Richtung Mittelpunkt.

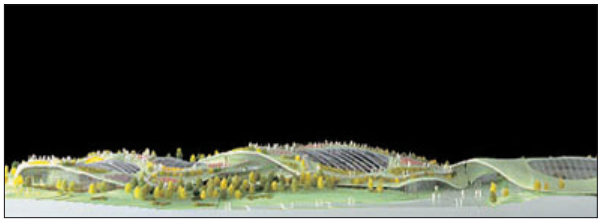


Abb. [25]) Modell I project. (Ito, 2005)

Das letzte Beispiel und die Ausnahme von der Regel, nach der größere Flächen nicht direkt 1:1 umgesetzt werden, ist das *I project*, eine Arbeit von *Toyo Ito (2005)* Abb. 25. Ein wesentlicher Bestandteil dieses Entwurfs ist eine frei gekrümmte Fläche, die aus einer durchgängigen, dünnen Betonmembran realisiert wird. Die Form dieser Fläche wurde zunächst grob modelliert und dann über eine spezielle Art der Topologieoptimierung verfeinert: In einem rekursiven Prozess wurden die Spannungsverläufe der Fläche über die FE-Methode berechnet. Es ergaben sich Bereiche mit unterschiedlicher Beanspruchung Abb. 26. In der folgenden Iteration wurden die erzeugenden Kurvenprofile entsprechend dieser Ergebisse verändert und die resultierende Fläche erneut berechnet. Dieser Prozess wurde solange fortgesetzt, bis die Fläche eine Form eingenommen hat, bei der keine Spannungsspitzen mehr auftraten. Diese Art der Strukturoptimierung nennt sich *„Evolutionary Structural Optimization (ESO)"* bzw. *„Extended Evolutionary Structural Optimization (EESO)" (Cui et al, 2003)* [11]. Durch die Anwendung dieses Verfahrens im *I project* entsteht eine statisch optimierte, dünnwandige Betonschale, die ohne Elementierung umgesetzt wird – was nicht für die Schalung gilt, die aus einzelnen, 3D-gefrästen Elementen zusammengesetzt werden muss.

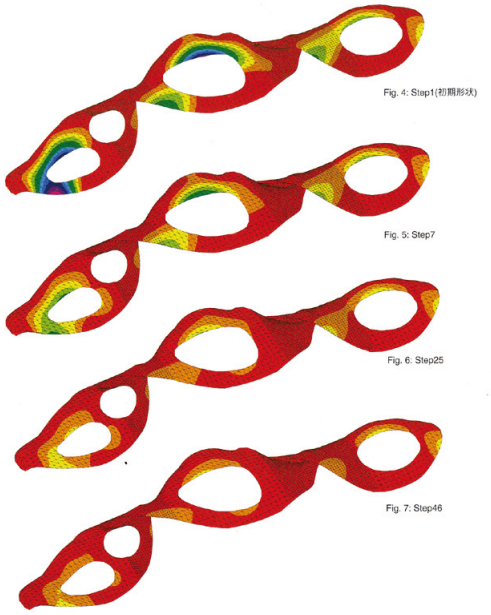


Abb. [26]) Evolutionäre Flächenstrukturoptimierung: FEM-Modelle I project. (vgl. Sasaki, 2005) [10]

6.3.2 Generative Logik von Structural Shapes

Diese Beispiele sollen genügen um zu verdeutlichen, dass die konstruktive Interpretation einer virtuellen Fläche ein vielfältiges Wechselspiel zwischen geometrischen und mechanischen Parametern ist, für das verschiedenste Lösungsansätze möglich sind Abb. 27.

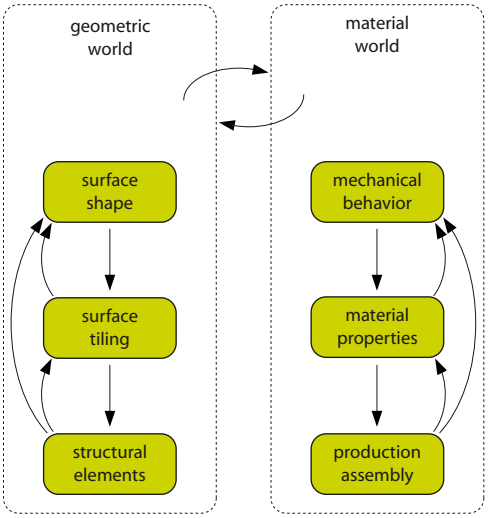


Abb. [27]) Einflussgrößen bei der Entwicklung einer konstruktiven Interpretation aus einer virtuellen Fläche.

Von der Seite generativer und algorithmischer Transformationsprozesse aus gedacht entstehen, allgemein gesprochen, auf der geometrischen Ebene wechselseitige Bedingungen aus der Form einer Ursprungsfläche, dem gewählten Aufteilungsverfahren sowie den Formen der einzelnen Bauteile, mit denen die Konstruktion letztlich realisiert werden soll. Diese Bedingungen sind entsprechend zu identifizieren, festzulegen und im Rahmen einer formerzeugenden Logik miteinander in Beziehung zu setzten. In einigen Fällen kann eine rein geometrische Betrachtung dieses Transformationsprozesses genügen: Im Produktdesign ist es beispielsweise nicht ungewöhnlich, Fragen der Statik oder von Materialeigenschaften implizit zu behandeln. Sie fließen bei der Entwicklung eines Entwurfs ‚aus dem Bauch heraus' mit in die Form ein. Dieses Gefühl für eine adäquate Proportionierung, das die Anforderungen der materiellen Welt mit beinhaltet, kann aber trügerisch sein. Vor allem bei größeren konstruktiven Strukturen bedarf es ohnehin einer Überprüfung. Insofern ist es sinnvoll, die Bedingungen, die sich aus dem Verhalten einer Struktur, den Materialeigenschaften sowie den gewählten Fertigungsverfahren ergeben, zumindest optional als Größen zu formulieren, die ihrerseits auf das Gefüge der geometrischen Parameter Einfluß nehmen können. Die Bedingungen der materiellen Welt, die selbst in wechselseitiger Abhängigkeit zueinander stehen, können auf verschiedene Weise mit der Welt der geometrischen Parameter interagieren. Sie können bereits die Form der Ursprungsfläche mit bestimmen, sie haben Einfluss auf die Art des Elementierungsverfahrens und auch auf die konkrete Dimensionierung der einzelnen Bauteile.

6.3.3 Structural Shapes und andere Flächeninterpretationen

Aus geometrischer Sicht ist eine Fläche ein unendlich dünnes, im dreidimensionalen Raum aufgespanntes Konstrukt. Jedes der genannten Beispiele ließe sich – unabhängig von seiner tatsächlichen Entstehungsgeschichte – auf eine oder mehrerer solcher Flächen zurückführen. Eine Fläche als *Structural Shape,* als eine Bauform zu begreifen, ist bereits eine weitere, wenn auch sehr allgemeine Interpretation des Begriffs ‚Fläche'. In dieser Interpretation ist Raum für weitere: So kann die Sitzfläche des *Eames-Chairs* als Geste begriffen werden, bei den Entwürfen von *Materialize-MGX* haben die Flächen skulpturalen Charakter, ebenso bei *ether/I* von *dECOi*. Bei *logo & vela*, der Überdachung für die Mailänder Messe ist die Fläche Tragwerk, ebenso beim *I project* von *Ito*, bei *Gehry's Walt Disney Concert Hall* sowie beim gezeigten Projekt von *LAB-Architects* haben die Flächen stark ornamentalen Charakter. Neben solchen gibt es noch eine Reihe weiterer Interpretationsmöglichkeiten für den Begriff ‚Fläche' *(vgl. Kilian, 2006)* [12]: Als gegenständliches Objekt, wie bereits in *Kapitel 2.4.3* diskutiert, als Bild, als Begrenzung, als Körper, als Baugruppe, als performative Fläche oder als parametrisches Objekt, um nur einige Weiteren zu nennen. Im Kontext generativer Methoden ist vor allem die parametrische Beschreibung einer Fläche von Interesse. Zusammen mit der in *Kapitel 2.3.2* diskutierten Eigenschaft von generativen Verfahren, Entwurfswissen zur Formerzeugung explizit und wiederabrufbar zu machen, erlaubt sie es, Methoden zur Überführung einer Fläche in eine Bauform unabhängig vom konkreten Entwurfsgegenstand zu betrachten. Wenn das Beziehungsgeflecht aus wechselseitigen geometrischen und mechanischen Bedingungen entsprechend mit den geometrischen Parametern einer Fläche assoziiert wird, ist es möglich, die Form einer Ursprungsfläche zu ändern und sie dadurch in einen anderen Kontext zu transportieren. Die transformierende Logik, die in den generativen Algorithmen ausgedrückt ist, bleibt anwendbar, sie passt sich der neuen Form an – eine Fläche läßt sich somit auch als ein Transportmedium von Designlösungen, als Designträger interpretieren.

6.4 Methodik der Experimente

6.4.1 Flächen als Träger von Designlösungen

Die Idee, Flächen in Kombination mit generativen Verfahren als Designträger zu nutzen, bedeutet also, dass die Form einer Fläche geändert wird und die entwerferische

Idee der konstruktiven Interpretation der Fläche weitgehend automatisch an die neue Form angepasst wird. Dieses Vorgehen wird in den Entwurfsexperimenten auf zwei unterschiedliche Arten realisiert.

Einmal wird die Elementierung einer *NURBS*-Fläche direkt über die Aufteilung ihres uv-Parameterraums vorgenommen. Eine *NURBS*-Kurve verfügt über einen Parameterraum (‚*t*' oder ‚*u*' genannt), den man sich als eindimensionales Koordinatensystem vorstellen kann, das direkt auf der Kurve liegt. Durch diesen Parameter läßt sich jeder Punkt auf der Kurve eindeutig identifizieren und in ein kartesisches Koordinatentripel (x,y,z) übertragen. Der kleinste t-Parameterwert entspricht dem Anfangspunkt, der größte dem Endpunkt der Kurve. Der Parameterraum einer *NURBS*-Fläche läßt sich folglich als zweidimensionales Koordinatensystem betrachten, die Parameter werden mit ‚*u*' und ‚*v*' bezeichnet. Auch hier liegt jedes uv-Parameterpaar direkt auf der Fläche und läßt sich in die entsprechenden kartesischen Koordinaten übertragen. Die vier Paare aus den kleinsten und größten Parameterwerten entsprechen dabei jeweils den Eckpunkten der Fläche. Die Normalenrichtung der Fläche schließlich bietet noch einen weitere, eindeutige Orientierungshilfe zum Ableiten konstruktiver Systeme aus dem Flächennetz [Abb. 28].

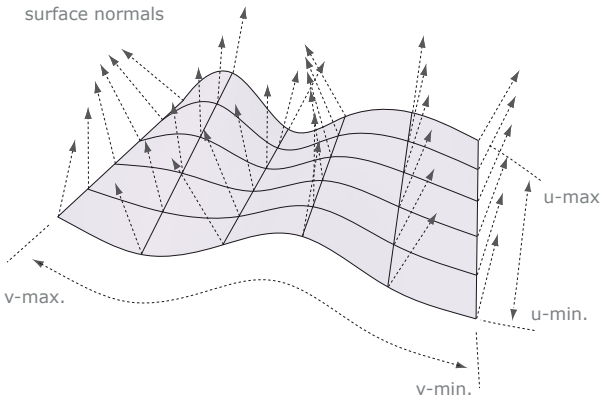


Abb. [28]) uvn-Parameterraum einer NURBS-Fläche.

Die Eigenschaften dieses uvn-Parameterraums sind topologisch, sie bleiben bei der Transformation einer *NURBS*-Fläche erhalten. Wird nun die Form einer Fläche verändert, bleibt die Anzahl der Maschen in beide Flächenrichtungen gleich und damit auch die Anzahl der Bauteile, die mit den Maschen assoziiert sind. Allerdings ändern sich die geometrischen Parameter der einzelnen Maschen zueinanander und zur Fläche [Abb. 29].

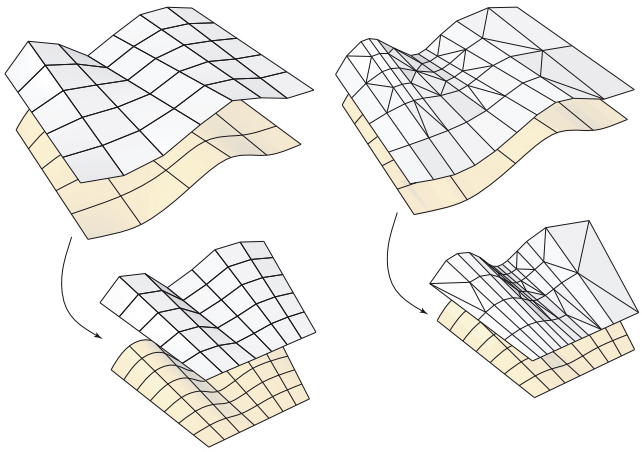


Abb. [29]) Flächenelementierung über uv-Raster (links) und über Mesh-Bildung in Rhino®.

Bei der zweiten Variante werden die Flächen auf dem üblichen Weg in *meshes* unterteilt. Bei dieser Art der Flächennetzerzeugung werden mehrere Iterationen durchgeführt, bis bestimmte vorgegebenen Parameter erfüllt sind. Bei der *Mesh*-Bildung im *CAD*-Programm *Rhino*® beispielsweise wird in einem ersten Schritt ein grobes Netz von Vierecksmaschen *(quad meshes)* erstellt, das drei Parameter erfüllt: Maximale Abweichung im Seitenverhältnis der Vierecke, Maximale Seitenlänge und Mindestanzahl an Vierecken. In einem zweiten Schritt werden die erzeugten *quad meshes* dann (optional) weiter unterteilt und zwar solange, bis die folgenden vier Parameter zusätzlich erfüllt sind: Maximaler Winkel von Flächennormalen zu anliegenden Maschenkanten, maximale Distanz der Kanten zur Fläche, minimale und noch einmal die maximale Seitenlänge. Hierbei ist es möglich, dass eine weitere Verfeinerung durch Vierecke einen Parameter (z.B.: die minimale Seitenlänge) verletzten würde. Dann erfolgt eine Unterteilung durch Dreiecke. In einem nachgeschalteten Arbeitsschritt ist es dann möglich, alle im *mesh* noch vorhandenen Vierecke zu

triangulieren. Wird bei dieser Art der Elementierung die Ursprungsfläche verändert, kann ein neues Flächennetz unter Beibehaltung der Parameter der Netzbestandteile untereinander sowie zur Fläche erzeugt werden. Es ändert sich allerdings in aller Regel die Anzahl und Anordnung der Maschen und somit auch die Anzahl und Anordnung der Bauteile, die daraus abgeleitet werden.

6.4.2 Gemeinsame Architektur der algorithmischen Entwurfsräume

Den algorithmischen Entwurfsräumen, die in den Experimenten entwickelt und untersucht wurden, um virtuelle Formen in konstruktive Strukturen zu überführen, liegt, ausgehend von dem eben beschriebenen Konzept, eine Fläche als Designträger zu begreifen, folgende allgemeine Architektur [Abb. 30] zu Grunde.

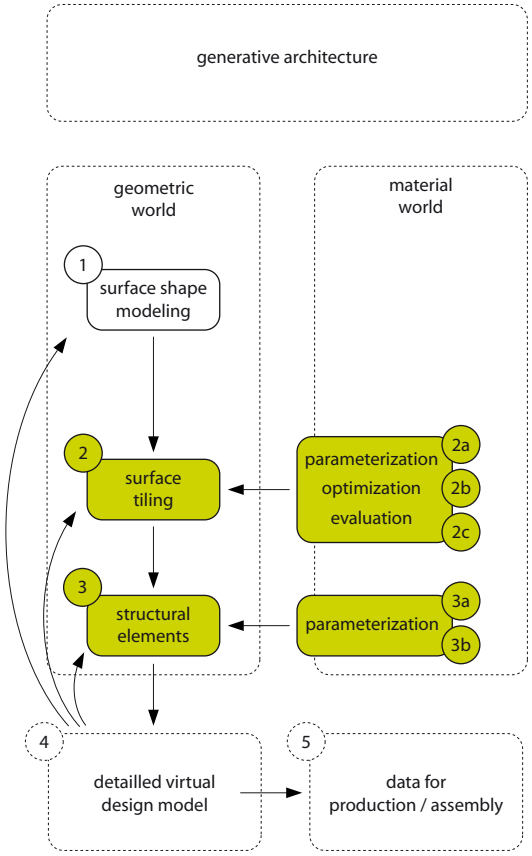


Abb. [30]) Allgemeine generative Architektur der entwickelten algorithmischen Entwurfsräume.

Schritt 1) Flächenmodellierung. Es ist bereits festgelegt worden *(vgl. Kapitel 6.2.1)*, dass die Modellierung einer Ausgangsfläche auf konventionelle Art, von Hand, über ein *CAD*-System erfolgen soll. Durch Maßstab und Form bekommt die Fläche Bedeutung: Als architektonisches Gebilde beispielsweise, als Hocker, Raumteiler oder beliebiges Testobjekt.

Schritt 2) Flächenelementierung. Der nächste Schritt ist die automatische Elementierung einer einzelnen Fläche. Hier sind eine ganze Reihe unterschiedlicher Verfahren denkbar, implementiert wurde eine Unterteilung in Drei- oder Viereckmaschen nach den oben beschriebenen Ansätzen. Dieser Elementierungsschritt kann auf verschieden Arten vorgenommen werden:

2a) Parametrisch: Hier werden lediglich die Parameter eines Netzes definiert, nach denen eine Fläche elementiert werden soll. Je nach gewähltem Aufteilungsverfahren können dies beispielsweise sein: Anzahl der Netzmaschen, Mindest- oder Maximallängen der Maschenkanten, maximale Abweichung des Winkels zwischen Flächenkanten und anliegenden Flächenormalen, usw. Im Prinzip kommen hier alle Parameter in Frage, wie sich auch bei der *Mesh*-Bildung in *CAD*-System verwendet werden.

2b) Optimierung eines Flächennetzes: Optional ist es möglich, ein erzeugtes Netz nachträglich zu verändern und dadurch zu optimieren. Die Optimierungskriterien können einmal aus den mechanischen Anforderungen an die Gesamtstruktur abgeleitet sein: Maximale Knotenverschiebung im Flächennetz, Spannungsverlauf in der Fläche, o.ä. Da die *Mesh*-Bildung immer ein näherungsweiser Prozess ist, der in mehreren Schritten abläuft, kann zum anderen auch eine nachgeschaltete Optimierung nach geometrischen Kriterien sinnvoll sein: Suche nach gleichen Stablängen im Netz, ein bestimmter Wertebereich für die Innenwinkel der Netzmaschen, Planarität von Viereckmaschen, usw.

2c) Evaluierung eines Flächennetzes: Die dritte Möglichkeit schließlich bessteht darin, eine Flächennetz nach de-

finierten Parametern zu erzeugen und nach bestimmten mechanischen Kriterien zu evaluieren. Dabei wird das Netz nicht verändert, die Daten können aber bei der konstruktiven Durchbildung Einfluss nehmen.

Schritt 3 – Konstruktive Durchbildung. Als dritter Schritt erfolgt, ebenfalls automatisiert, die detaillierte Ableitung der Bauteile einer konstruktiven Struktur aus den geometrischen Parametern der Ursprungsfläche sowie eines zugehörigen Flächennetzes. Für jede konstruktive Interpretation wird ein eigenes Programmmodul entwickelt, das auf zwei verschiedene Arten ausgebildet sein kann:

a) Parametrisch. Die wesentlichen geometrischen Parameter der Bauteile eines Konstruktionssystems sind variabel gehalten und untereinander in Beziehung gesetzt. Sie können nahezu beliebig geändert werden, vorausgesetzt, die Parameter von Ausgangsfläche und Flächennetz bieten dafür den entsprechenden Spielraum: Änderungen in den Parametereinstellungen für die Erzeugung der Bauteile dürfen nicht in Ergebnissen resultieren, die zwar geometrisch, aber nicht mehr materiell interpretierbar wären (z.B. Selbstüberschneidungen von Flächen oder Überschneidungen zwischen Bauteilen).

b) Parametrisch unter Einbeziehung der Evaluierungsergebnisse. Bei dieser Art von Programmmodulen sind die wesentlichen Konstruktionsparameter ebenfalls variabel gehalten. Zusätzlich beeinflussen aber die Evaluierungsergebnisse eines Flächennetzes in einem definierten Maß die Dimensionierung der Bauteile des Konstruktionssystems.

Schritt 4 – Visualisierung als Designmodell. In diesem Schritt können die erzeugten Modelle in verschiedenen Detaillierungsgraden visualisiert werden.

Schritt 5 – Ausgabe von Produktions- und Montagedaten. Als letzter Schritt schließlich werden die benötigten Datensätze für Fertigung und Montage ausgegeben. Je nach Entwurf können dies beispielsweise einfache Teilelisten, Konturen für Laser- oder Wasserstrahlschneiden, Flächenkörper für die Fräsbearbeitung, **.stl* - Dateien für das *Rapid-Prototyping*, usw. sein.

6.4.3 Entwerferischer Umgang mit den algorithmischen Entwurfsräumen

Der entwerferische Umgang mit dieser generativen Architektur bewegt sich im wesentlichen auf zwei Ebenen: Die eine Ebene ist die Anwendung bereits entwickelter und implementierter algorithmischer Entwurfsräume – jede Kombination aus jeweils einem Elementierungsverfahren und einem Programmodul zur konstruktiven Interpretation entspricht einem Entwurfsraum. Wesentliche Gestaltungsspielräume sind hier bereits definiert. Die zweite Ebene ist die Entwicklung neuer algorithmischer Entwurfsräume durch weitreichende Modifikationen bereits vorhandener oder dem Hinzufügen neuer generativer Algorithmen. Die nachfolgende, beispielhafte Beschreibung des Einsatzes der generativen Prozessketten orientiert sich an den eben beschriebenen Schritten der generativen Architektur. Obwohl diese Beschreibung der Übersichtlichkeit halber linear angelegt ist, kann der Einstieg in die Prozesskette, wie bei jedem anderen Entwurfsprozess auch, an einer beliebigen Stelle erfolgen:

Schritt 1) – Flächenentwurf. Der Prozess beginnt mit der Modellierung von *NURBS*-Flächen für einen bestimmten Entwurfszweck. Im Gegensatz zum üblichen Vorgehen beim Entwerfen mit *CAD*-Programmen werden hier keine Volumenkörper definiert und schrittweise bildhafte virtuelle Modelle entwickelt. Die modellierten Flächen bleiben ohne Dickendimension, als abstrakte symbolische Modell für eine vorgestellte konstruktive und ästhetische Durchbildung. Die Informationen für diese gedachte gestalterische Interpretation werden durch die nachfolgenden Schritte hinzugefügt und experimentiert.

Schritt 2) – Entwurf und Anwendung von Aufteilungsmethoden. Die Flächenaufteilung ist das Bindeglied zwischen Flächenform und konstruktiver Umsetzung. Die Wahl der Elementierungsmethode richtet sich dabei nach

einer mehr oder minder konkreten Vorstellung, wie die strukturellen Elemente der Bauform sind, aus welchen Materialien und mit welchen Verfahren sie gefertigt sein sollen: Etwa ob flächige Elemente verwendet werden, die nur innerhalb gewisser Toleranzen biegbar sind, Standardknoten, die eine bestimmten Raum benötigen, Mindest- und Maximallängen bei den Stäben, statische Anforderungen, etc. Gelingt es, das Flächennetz im Rahmen der gewünschten Parameter aus einer Fläche zu erzeugen, kann zur konstruktiven Durchbildung übergegangen werden. Ist dies nicht der Fall, dann gibt es folgende Strategien:

a) Änderung der Parameter für die Evaluierung eines Flächennetzes,
b) Änderung der Parameter für die Optimierung eines Flächennetzes,
c) Änderung der Parameter zur Erzeugung eines Flächennetzes.

Die Anwendung von einer oder mehrerer dieser Optionen zieht nur geringfügige Änderungen des entworfenen Bilds einer konstruktiven Umsetzung nach sich, der Entwerfer bewegt sich noch im selben algorithmischen Entwurfsraum. Darüber hinaus gibt es folgende Möglichkeiten:

d) Abwandlung eines bestehenden Elementierungsverfahrens, Entwurf und Implementierung einer neuen Aufteilungsmethode.

Mit diesen Optionen begibt sich der Entwerfer von der Ebene der Anwendung algorithmischer Prozeduren auf die Metaebene des Algorithmendesigns, entsprechend gravierend kann auch der Einfluss auf die ursprünglich vorgestellte Entwurfslösung sein. Dann ist es an dieser Stelle noch möglich,

e) die Ursprungsfläche zu überarbeiten,
f) die Rahmenbedingungen des Entwurfs zu überdenken und alternative Lösungsmöglichkeiten zu entwickeln und selbstverständlich auch
g) einen völlig anderen Entwurfsweg einzuschlagen.

Schritt 3) Entwurf und Anwendung von Algorithmen zur konstruktiven Interpretation eines Flächennetzes. In diesem Teil des Prozesses erfolgt die detaillierte, parametrisch gesteuerte, konstruktive Interpretation eines Netzes. Die verwendeten Programmmodule (Konstruktionsmodule) sind, wie die Programmmodule für die Flächenaufteilung, als eine Art erweiterbare Bibliothek konzipiert. Für jeden gewünschten Konstruktionstyp muss ein eigenes Modul entwickelt und verwendet werden. Bei der Entwicklung solcher Module wird die möglicherweise noch vage Idee einer konstruktiven Umsetzung präzisiert, dabei wird auch entschieden, welche Parameter der Konstruktion variabel gehalten werden sollen und ob und wie die Ergebnisse der Evaluierung eines Flächennetzes in die Erzeugung der Bauteile einfließen sollen.

Schritt 4) Visualisierung des erzeugten Designmodells. Abgesehen von eventuellen Warn-, Fehlermeldung oder Programmabstürzen, kann erst in diesem Schritt visuell überprüft werden, ob das gesamte erzeugte Designmodell den ursprünglichen Vorstellungen entspricht und ob es fehlerfrei erzeugt wurde. Gelingt die konstruktive Durchbildung zufriedenstellend, wird das Konstruktionsmodul so weiterentwickelt, dass es die für Fertigung und Montage benötigten Daten ausgegeben kann. Gelingt sie nicht, oder sind die Ergebnisse aus irgend einem anderen Grund unbefriedigend, dann:

h) werden die Parameter für die Interpretation der Evaluierungsergebnisse verändert,
i) werden die allgemeinen Parameter zur Erzeugung der Bauteile verändert,
j) wird das Konstruktionsmodul überarbeitet oder ein neues Konstruktionsmodul entworfen und implementiert, oder es
k) kommen a, b, c, d, e oder f aus Schritt 3 zum Tragen.

Schritt 5 – Ausgabe von Produktions- und Montagedaten. Die Erzeugung und Ausgabe der Daten für Fertigung und Montage ist der letzte Schritt in einer generativen Prozesskette und gleichzeitig die Schnittstelle eines algorith-

mischen Entwurfsraums zur physischen Realität. Die in hohem Grad abstrakte, symbolische Repräsentation von Designmodellen, wie sie die Entwicklung generativer Prozessketten ist, braucht eine ständige Überprüfung des Entwurfs am physischen Modell oder Prototyp.

Da es fragwürdig erscheint, im Rahmen von Entwurfsexperimenten zur Methodenentwicklung, die nicht in einen realen Kontext eingebunden sind, Entwerfen zu simulieren, ist diese Überprüfung am physischen Modell nur soweit vorgenommen worden, dass die prinzipielle Anwendbarkeit der entwickelten Algorithmen in der Praxis belegt ist. Trotzdem ist dieser letzte Schritt gleichzeitig auch die entscheidende Stelle im Prozess, die bei einer konkreten Aufgabe immer wieder durchlaufen würde und bei der auch alle gerade genannten Strategien von a) bis j) zur Anwendung kommen könnten. Dies beinhaltet selbstverständlich auch die Gefahr oder Chance einer ständigen Überarbeitung und Neudefinition der *Constraints* des Entwurfs und der Entwurfsziele sowie möglicherweise die Erkenntnis, dass sich das vorliegende Entwurfsproblem nicht so konstruieren läßt, dass es generativ bearbeitet werden kann und eine andere Entwurfsmethode vorzuziehen wäre. Der gesamte Entwurfsprozess entlang der beschriebenen generativen Architektur ist in der Abbildung auf *Seite 109* Abb. 31 zusammenfassend dargestellt.

6.5 Entwicklungsumgebungen und Zielsetzungen

6.5.1 Entwicklungsumgebungen

Die entworfenen Algorithmen wurden zunächst in *VisualBasic-Script® (VBScript)* als Arbeitsskripte implementiert, d.h. die einzelnen Programmodule sind voll funktionsfähig, die Einstellungen für den Programmablauf werden jedoch direkt auf Skriptebene vorgenommen. Es sind keine grafischen *Interfaces* und nur die nötigsten Sicherheitsabfragen zur Gewährleistung eines stabilen Programmdurchlaufs vorhanden.

Flächen, die aufgeteilt, optimiert und konstruktiv durchgebildet werden sollen, müssen in *Rhino®* vorliegen, eine *CAD-Software*, die auf *NURBS*-Modellierung spezialisiert ist. Diese Software verfügt neben einer offenen *C++* - Schnittstelle über eine *Scripting*-Umgebung die auf *VB-Script* basiert und die spezielle Methoden enthält, um verschiedene Funktionen von *Rhino®* zu nutzen und Macroskripte in die *Scripting*-Umgebung zu integrieren und auszuführen. Die Berechnung der Statik einer Flächenstruktur erfolgt durch *RSTAB®*, eine *Software* zur Berechnung von Stabtragwerken auf Basis der Finite Elemente (FE) - Methode. Auch diese *Software* verfügt über eine Programmierschnittstelle, über die die Eingabe von Lastfällen, Querschnitten, Materialien, Knotenkoordinaten, Knoten-/ Stabbeziehungen usw. erfolgt und die Anwendung gesteuert wird. Um diese Schnittstelle nutzen zu können, mussten der in Experiment zwei verwendete genetische Algorithmus sowie die Algorithmen zur Flächenelementierung nach *VisualBasic for Applications®* (VBA) portiert werden, was relativ problemlos war, da die Syntax dieser Programmiersprache fast mit der von *VB-Script* identisch ist. Die Topologieoptimierung der Flächen in Experiment 1 wurde über *ANSYS®* durchgeführt, das ebenfalls nach der FE-Methode arbeitet. Die dafür notwendigen Skripte wurden in der *ANSYS®* eigenen Skriptsprache programmiert.

Der Austausch von Flächen zwischen verschiedenen *CAD-Softwares* erfolgt über **.igs (IGES)* – einer der Standards für diese Anwendung. Alle anderen Daten werden über Import und Export von Textdateien ausgetauscht. Die Funktionen, die in den einzelnen Programmmodulen enthalten sind, wurden zum größten Teil selbst entwickelt und implementiert. Einige Methoden zur Vektorberechnung sowie zur Matrixtransformation sind der *design scripting library* von *Stylanios Dritsas (2003)* [13] entnommen. Anregungen zu verschiedenen Sortier-Algorithmen stammen aus *Robert Sedgewicks (2002)* [14] Grundlagenwerk *„Algorithmen"*, die Skripte zur Topoplogieoptimierung in *ANSYS®* wurden von *Gregor Zimmermann (2007)* [15] im Rahmen seiner Promotion zum Thema der *„Membran Beton Gitterschalen Tragwerke"* entwickelt.

6.5.2 Zielsetzungen

Die Zielsetzungen für den experimentellen Teil dieser Arbeit sind aus der Betrachtung verschiedener generativer Arbeiten im weiteren Umfeld des beschriebenen Themenfelds abgeleitet:

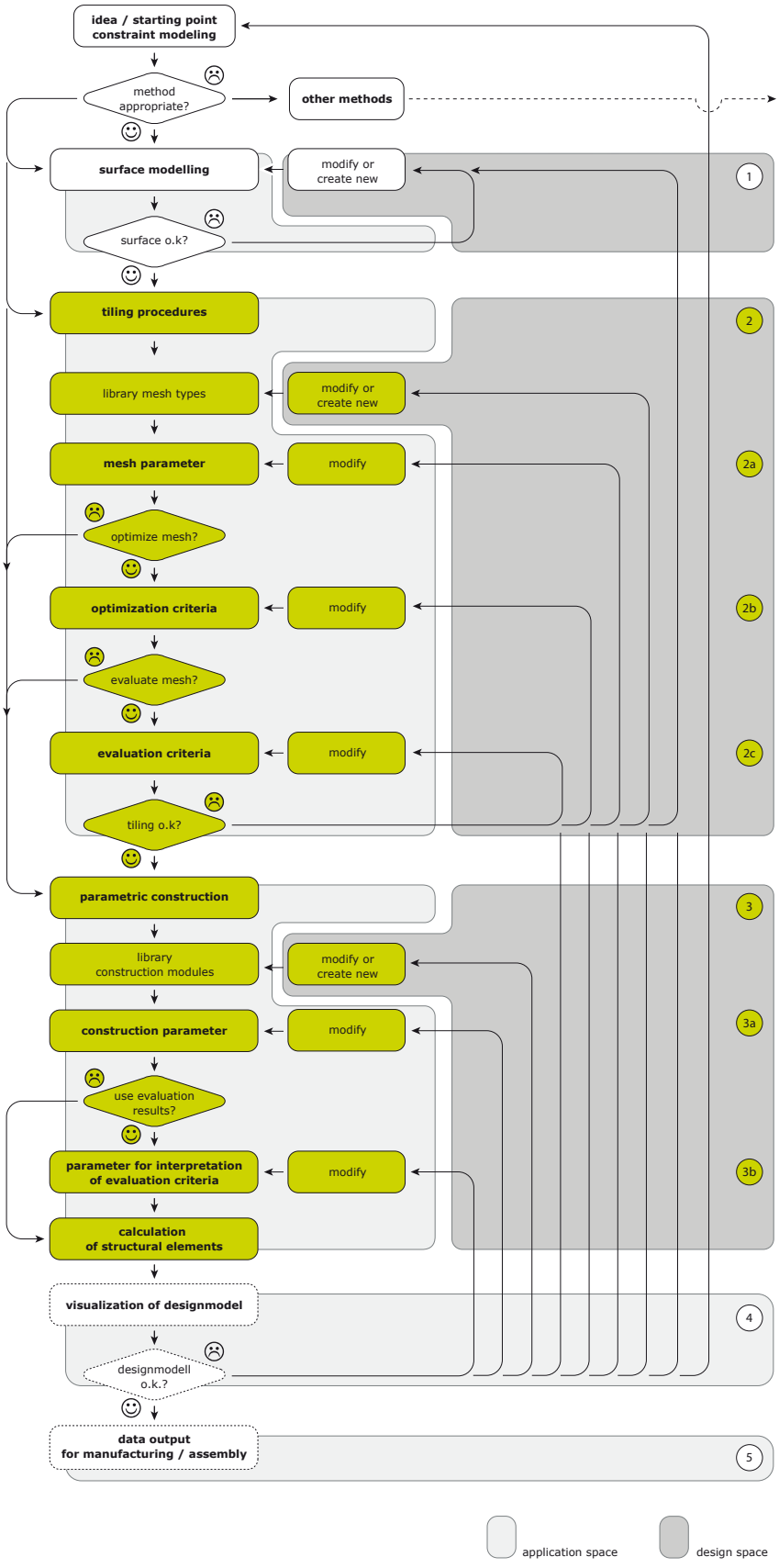
- *Praxistauglichkeit des Ansatzes.* Die Methode sollte soweit entwickelt werden, dass ihre praktische Relevanz deutlich wird und sie auch in einem realen Entwurfskontext verwendet werden könnte.
- *Bearbeitung eines möglichst großen Formenspektrums,* insbesondere von freien Formen, ohne dass diese dabei auf mathematische Flächen zurückgeführt werden müssten.
- *Entwicklung bzw. Verwendung flexibler Konstruktionssysteme,* die gut auf lokale Unterschiede zwischen den einzelnen Elementen reagieren.
- *Durchgängige Datenübertragung* von der Modellierung der Ursprungsfläche bis hin zu Fertigung und Montage.
- *Automatisierung von Routine-Arbeitsabläufen*, die sich in solchen Entwurfsprozessen wiederholen.
- *Standardisierung einer Methode anstatt Standardisierung der erzeugten Ergebnisse:* Eine Vielzahl verschiedener konstruktiver Lösungen soll für eine große Anzahl unterschiedlicher Flächen realisiert werden.

Diese allgemeinen Ansprüche entsprechen einem eher rationalen Zugang zum algorithmischen Entwerfen und benennen intersubjektiv nachvollziehbare Kriterien, deren Erfüllung bislang keine Selbstverständlichkeit ist: Experimentelle generative Entwurfsansätze verbleiben oft auf einem sehr abstrakten Niveau, dadurch ist ihre möglicherweise vorhandene praktische Relevanz nur schwer zu erkennen. Praktische Arbeiten hingegen werden häufig spezifisch, für ein einzelnes Entwurfsprojekt entwickelt und sind dabei auf bestimmte Flächentypen und Konstruktionsverfahren festgelegt.

Neben diesen anwendungsorientierten Zielsetzungen wurde zu Beginn der Arbeit auch die Fragestellung thematisiert, ob und inwieweit generative Methoden über ihre Funktion als Werkzeuge zur Umsetzung bereits weitgehend ausdifferenzierter Entwurfsideen hinaus als Medien des Entwerfens verwendet werden können – wie mit der Idee der algorithmischen Entwurfsräume ausgedrückt. Auch für die Beantwortung dieser Frage ließen sich nachvollziehbare Kriterien formulieren, sie geht aber über ein rein intellektuellen Verstehen hinaus und ist letztendlich nur subjektiv, vom einzelnen Entwerfer zu entscheiden: Er kann sich einen selbstverständlichen Umgang mit generativen Werkzeugen und Methoden erarbeiten, ein Gefühl für die Möglichkeiten des algorithmischen Entwerfens bekommen, es als Erweiterung seines gestalterischen Repertoires sowie seiner Entwerferpersönlichkeit begreifen – oder eben nicht.

Insofern hat auch der experimentelle Teil dieser Arbeit zwei Seiten: Anwendungsbezogene, praxisorientierte Methodenentwicklung und gleichzeitig eine Art entwerferischer Selbstversuch.

Abb. [31] - nächste Seite) Schematische Darstellung möglicher Strategien für den Umgang mit algorithmischen Entwurfsräumen.


idea / starting point
constraint modeling
method appropriate?
other methods
surface modelling
modify or create new
1
surface o.k?
tiling procedures
2
library mesh types
modify or create new
mesh parameter
modify
2a
optimize mesh?
optimization criteria
modify
2b
evaluate mesh?
evaluation criteria
modify
2c
tiling o.k?
parametric construction
3
library construction modules
modify or create new
construction parameter
modify
3a
use evaluation results?
parameter for interpretation of evaluation criteria
modify
3b
calculation of structural elements
visualization of designmodel
4
designmodell o.k.?
data output for manufacturing / assembly
5
application space
design space

7. Entwurfsräume 1 – Topologieoptimierung und parametrisches Design

7.1 Constraints und Beschreibung

Die hier gezeigten algorithmischen Entwurfsräume benutzen parametrische Designmethoden um die Ergebnisse der Topologieoptimierung von Flächen gestalterisch zu interpretieren. Die allgemeinen Rahmenbedingungen für die Entwicklung dieses Entwurfsraums sind, wie bereits in *Kapitel 6.4.2* dargestellt:

1) Modellierung einer Fläche per Hand über ein *CAD*-System,
2) Entwicklung einer räumlichen Struktur aus der Fläche,
3) Bearbeitung einer einzelnen Fläche,
4) Betrachtung einer Fläche als symbolisches Modell.

Darüber hinaus wurde festgelegt:

5) Umsetzung der Struktur über ein 3D-*Rapid-Prototyping*-Verfahren wie 3D-Druck, Lasersintern oder Stereolitografie,
6) der Prototyp ist gleich dem Endprodukt.

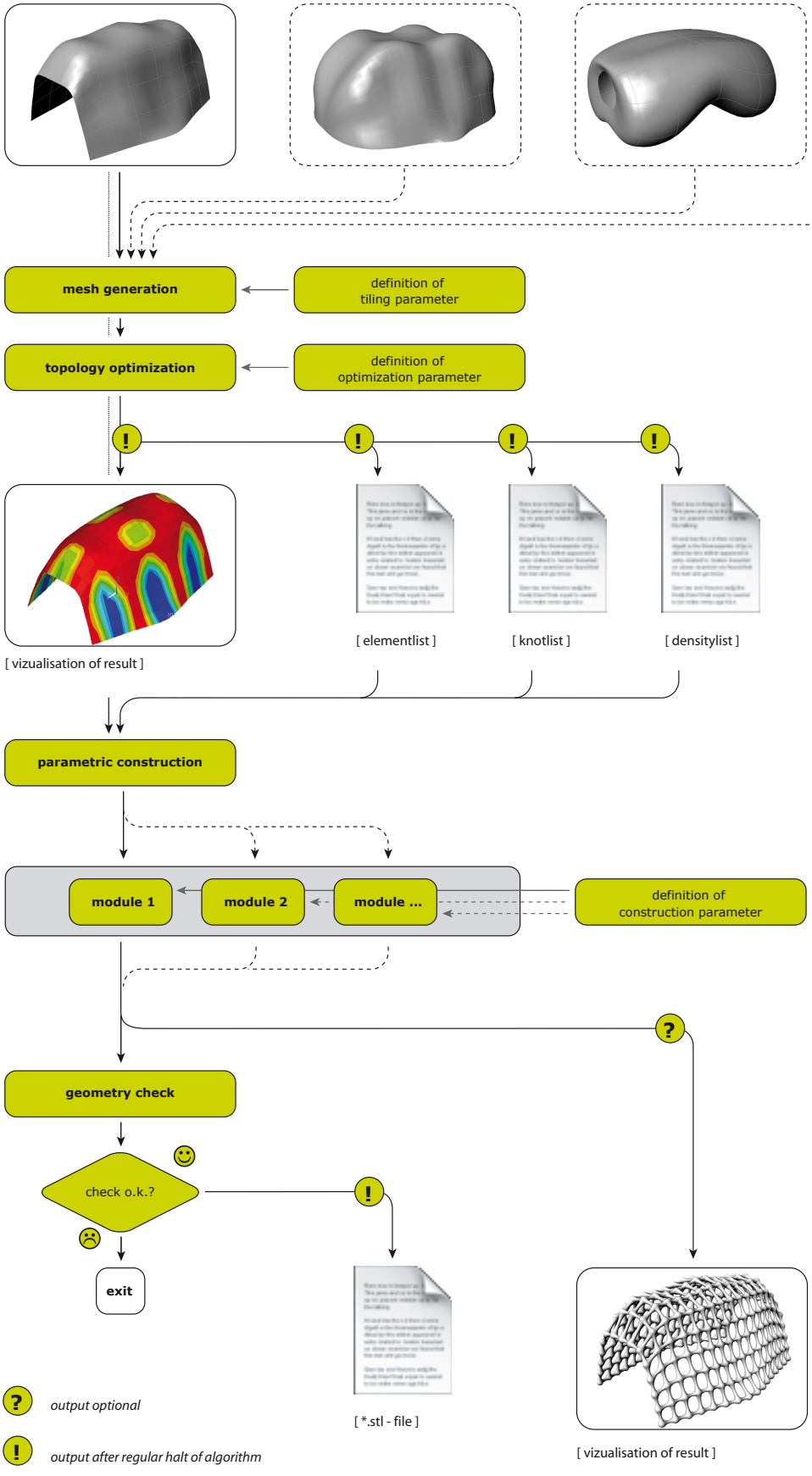
Aus den beiden letzten *Constraints* ergibt sich ein großer Spielraum bezüglich der formalen Ausgestaltung einer konstruktiven Struktur, da mit den 3D-*Prototyping*-Verfahren auch ‚verrücktere' Geometrien umgesetzt werden können und auf sich wiederholende Bauteile verzichtet werden kann – bei gleichen Bauvolumen sind die Herstellungsbedingungen etwa gleich, unabhängig von der Form einer Struktur. Gleichzeitig haben diese Herstellungstechniken auch Limitierungen, wie geringere Bauraumgrößen, hohe Material- und Energiekosten, lange Bearbeitungszeiten, teilweise geringere mechanische Belastbarkeit der erzeugten Bauteile oder die Druckauflösung des gewählten Verfahrens, mit dem sehr feine Strukturen eventuell nur ungenügend abgebildet werden können. Aus diesen Gegebenheiten sind zwei weiter Rahmenbedingungen definiert worden:

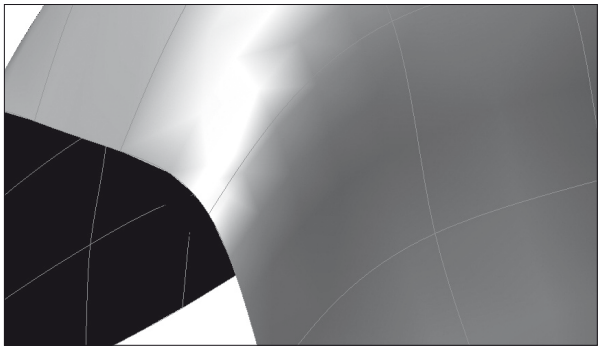
7) Anwendungsgegenstand sind Hocker, die eine dem gewählten Fertigungsverfahren adäquate Größenordnung haben,
8) die konstruktive Struktur dieser Hocker besteht aus einem einzelnen Bauteil.

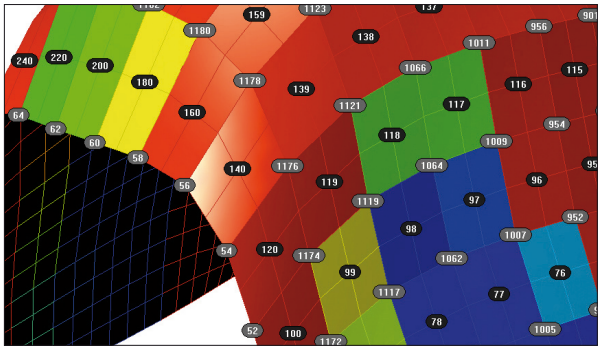
7.2 Prozesskette

Die generative Prozesskette beginnt mit der Modellierung einer Fläche, die dann über das **.igs (IGES)*-Austauschformat aus einer *CAD*-Software exportiert und in *ANSYS®* importiert wird. Danach folgen auf Skriptebene die Parametereinstellung für die Flächennetzerzeugung sowie die Topologieoptimierung, die anschließend durchgeführt werden. Als *Output* wird das Ergebnis der Topologieoptimierung visualisiert und als Bilddateien in mehreren Ansichten gespeichert. An diesen Darstellungen läßt sich schon in etwa erkennen, wie die später angewendeten Konstruktionsmodule die Fläche interpretieren werden. Als weiterer *Output* werden drei Textdateien erstellt. Die erste enthält die Indices der einzelnen Flächenelemente sowie die Indices der jeweils zugehörigen Knoten. Die zweite Textdatei enthält die Knotenindices und die zugehörigen kartesischen Koordinaten. Die dritte Datei schließlich enthält die Ergebnisse der Topologieoptimierung: Jedem einzelnen Flächenelement wird ein Dichtewert zwischen 0 (keine Belastung) und 1 (maximale Belastung) zugeordnet.

Diese drei Textdateien sowie die Ausgangsfläche sind die Grundlage der nun folgenden parametrischen Konstruktion [Abb. 01]. Es wird eines der vorhandenen Konstruktionsmodule ausgewählt und die gewünschten Parameter eingestellt. Darauf hin wird die entsprechende Struktur aus den geometrischen Daten des Flächennetzes, denen der ursprünglichen Fläche sowie optional denen der Ergebnisse der Topologieoptimierung automatisch erzeugt. Nachdem die konstruktive Struktur berechnet wurde, wird die erzeugte Geometrie visualisiert, damit das Ergebnis überprüfbar wird. Schließlich wird dieses Ergebnis noch darauf hin evaluiert, ob sich daraus ein geschlossener Flächenkörper erstellen läßt.


mesh generation
definition of tiling parameter
topology optimization
definition of optimization parameter
[vizualisation of result]
[elementlist]
[knotlist]
[densitylist]
parametric construction
module 1
module 2
module ...
definition of construction parameter
geometry check
check o.k.?
exit
[*.stl - file]
[vizualisation of result]
output optional
output after regular halt of algorithm






```
//* Auszug Element-/Knotenzuordnung
160.000, 1178.000, 56.000, 58.000, 1180.000
180.000, 1180.000, 58.000, 60.000, 1182.000

//* Auszug Knoten-/Koordinatenzuordnung
1178.000, 231.992, 75.166, 213.220
  56.000, 260.000, 71.512, 180.940
  58.000, 260.000, 43.165, 184.899
  60.000, 260.000, 14.408, 184.569
1180.000, 231.992, 47.603, 227.066
1182.000, 231.992, 16.027, 232.365

//* Auszug Element-/Dichtewertzuordnung
160.000, 1.000
180.000, 0.750
```

Abb. [01]) Visualisierung von Mesh-Bildung, den Ergebnissen der Topologieoptimierung im CAD-System sowie der Datengrundlage zur konstruktiven Durchbildung..

Abb. [02] – linke Seite) Schematische Darstellung der generativen Prozesskette.

Dies ist die wesentliche Voraussetzung für eine Übertragung der Daten zu einem *Rapid-Prototyping*-Verfahren. War die Prüfung erfolgreich, wird aus dem Flächenkörper eine **.stl* (Stereolitografie)-Datei erstellt. Andernfalls muss von Hand nachgearbeitet werden um die ‚Löcher' im Flächenkörper zu verschließen. Nach diesen Schritten ist ein Ablauf der generativen Prozesskette [Abb. 02] abgeschlossen. Die wesentlichen Gestaltungsparameter innerhalb der Entwurfsräume, die die Prozesskette beschreibt, bestehen in Änderungen an der Ausgangsfläche, den Einstellungen für *Mesh*-Bildung, Topologieoptimierung und parametrische Konstruktion. Neue Entwurfsräume entstehen innerhalb dieses generativen Systems durch eine Entwicklung alternativer Konstruktionsmodule.

7.3. Flächenelementierung und -evaluierung

Für die Unterteilung einer Fläche werden die internen *Meshing*-Algorithmen von *ANSYS*® benutzt. Es werden zweidimensionale Netzelemente erzeugt. An Einstellungen kann hier definiert werden, ob ein Netz aus drei- oder viereckigen Elementen bestehen soll [Abb. 03], wobei in der Regel (in Abhängigkeit von der Ursprungsfläche) ein Dreiecksnetz auch einige wenige Vierecke enthält und umgekehrt. Die Logik der späteren konstruktiven Interpretation muss also so angelegt sein, dass sie in jedem Fall auch aus solchen gemischten Netzen abgeleitet werden kann. Neben der Art des Flächennetzes kann noch die ungefähre Kantenlänge [Abb. 04] der Elemente festgelegt werden.

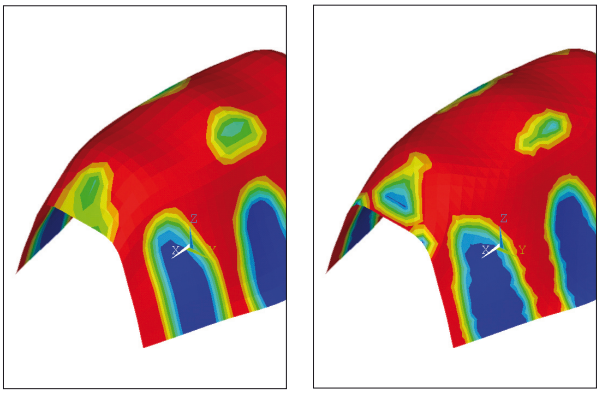


Abb. [03]) Topologieoptimierung mit Vierecks- und Dreiecksmaschen bei der selben Kantenlänge.

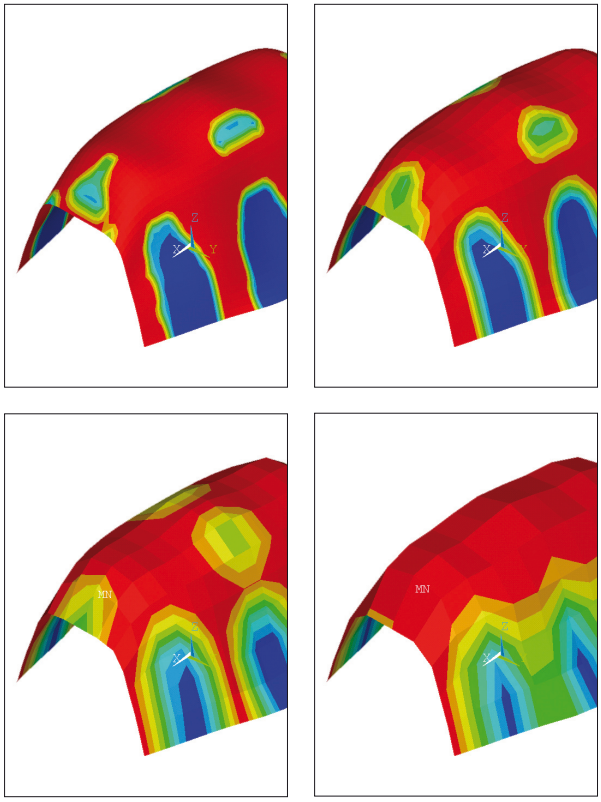


Abb. [04]) Topologieoptimierung mit einer Kantenlänge der Elemente von 10, 20, 40 und 60mm.

Mit diesen beiden Einstellungsmöglichkeiten ist die Einflussnahme des Entwerfers auf die Flächennetzerzeugung ausgeschöpft. Beides sind jedoch wichtige Gestaltungsgrößen, da sie einmal maßgeblich über die Anmutung des generierten Entwurfs entscheiden, zum anderen führt eine zu hohe Zahl an Elementen schnell an die Kapazitätsgrenzen des generativen Systems oder erhöht zumindest die Rechenzeit erheblich – was vor allem dann von Belang ist, wenn diese Methode als Entwurfswerkzeug in frühen Phasen des Prozesses betrachtet wird, also möglichst schnell Ergebnisse sichtbar werden sollen.

Weitere Gestaltungsmöglichkeiten ergeben sich aus den Parametern für die Topologieoptimierung der Fläche: Bei dieser Art der Optimierung wird eine Fläche in mehreren Iterationsschritten so berechnet, dass für gegebene Lastfälle die optimale Materialverteilung einer Fläche ermittelt wird. Diese Verteilung ist abgebildet durch den sogenannten Dichtewert, der den einzelnen Elementen des Flächennetzes zugeordnet wird. Eine Dichtewert von 1 bedeutet statisch hoch aktive Bereiche im entsprechenden Flächenteil, in den Abbildungen rot dargestellt. Der niedrigste Wert ist 0 (blau dargestellt) und bedeutet keine oder vernachlässigbare statische Aktivität. Die übrigen Werte liegen in der Farbskala dazwischen. Die Methode der Topologieoptimierung wird üblicherweise dazu verwendet, die Geometrie einer Fläche so zu verändern, das die Bereiche mit niedrigem Dichtewert entfallen und ein möglichst homogener Spannungsverlauf in der Fläche entsteht. Dazu stehen prinzipiell zwei Methoden zu Verfügung. Die erste besteht darin, die Form einer Fläche entsprechend der Analyseergebnisse in jeder Iteration automatisch verändern zu lassen. Hier kann sowohl die Verformung der Fläche miteinbezogen als auch niedrig belastete Bereiche unter einem definierten Dichtewert entfernt werden. Die zweite Vorgehensweise ist, die Geometrie der Fläche unverändert zu lassen: Hier werden dann lediglich die Bewertungsergebnisse eines Iterationsschrittes in die Ausführung des nächsten miteinbezogen.

Da in diesem Experiment die Ergebnisse der Topologieoptimierung als entwerferisches Mittel dienen, die in die Ausformung der konstruktiven Struktur miteinbezogen werden sollen, wird die zweite Variante verwendet. Was aus Ingenieurssicht Bereiche definieren mag, für die eventuell ein Belastungsnachweis oder eine spezielle konstruktive Lösung zu erbringen beziehungsweise zu entwickeln ist, wird im entwerferischen Sinn zur direkt interpretierten, gestalterischen Einflussgröße – Optimierung als Ausgangspunkt einer Entwurfslösung.

Man muss hier, im Rahmen dieser Experimente allerdings darauf hinweisen, dass durch dieses Vorgehen keine statisch optimierten Strukturen entstehen, sondern lediglich eine Logik von struktureller Belastung durch Gestaltung ausgedrückt wird: Es werden zwar die korrekten Materialkennwerte entsprechend des gewählten Fertigungsverfahrens sowie realistische Lastfälle zur Berechnung verwendet. Der errechnete Dichtewert gibt aber nur eine Verteilung der Belastung über die Fläche wieder, nicht die tatsächlich Kraft, die an einer bestimmten Stelle wirkt.

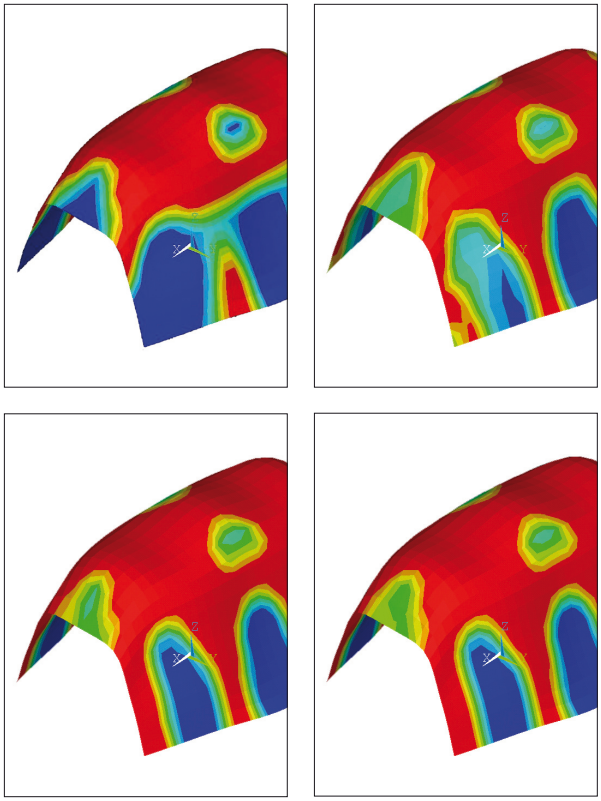


Abb. [05]) Topologieoptimierung mit 2, 5,10 und 25 Iterationen.
Weitere Durchläufe bringen bei diesem Beispiel nur noch marginale Veränderungen.

Die erste Größe, die bei der Topologieoptimierung in der generativen Prozesskette verändert werden kann, ist die Anzahl der Iterationen [Abb. 05]. Da sich eine ideale Flächenstruktur erst nach einer gewissen Anzahl an Durchläufen einstellt, verändern sich mit jedem Berechnungsschritt auch die Dichtewerte der einzelnen Elemente – in den ersten Durchläufen sehr stark, nach einer gewissen Anzahl an Iterationen werden die Änderungen marginal, bis die Dichteverteilung schließlich gleich bleibt.

Der nächste Gestaltungsparameter mit Auswirkungen auf die Optimierungsergebnisse ist der Lasteintrag. Dafür wird zunächst das Elastitzitätsmodul (E-Modul) des gedachten Materials bestimmt – im Falle der hier gezeigten Beispiele für ABS (Acryl-Nitril-Butadien-Styrol), das in einigen 3D-Druckern verwendet wird. Der Lasteintrag ist so angelegt, dass er immer als Druck entlang der z-Achse wirkt, andere Lastfälle sind in der augenblicklichen Version der Skripte nicht vorgesehen. Weiter kann die Gewichtskraft für die Last definiert werden, sowie ein Wertebereich, wieder entlang der z-Achse, in dem die Last wirken soll. Sie wird dann auf alle Elemente, die sich innerhalb dieses Bereichs befinden, gleichmäßig verteilt. Je nach Wertebereich ändert sich dann auch die Dichteverteilung der Elemente, wie in der Abbildung [Abb. 06] gezeigt. Die Auflager sind in allen Fällen statisch und werden an alle Knoten des Flächennetzes angelegt, die im Wertebereich unterhalb von 2mm, wieder entlang der z-Achse, liegen.

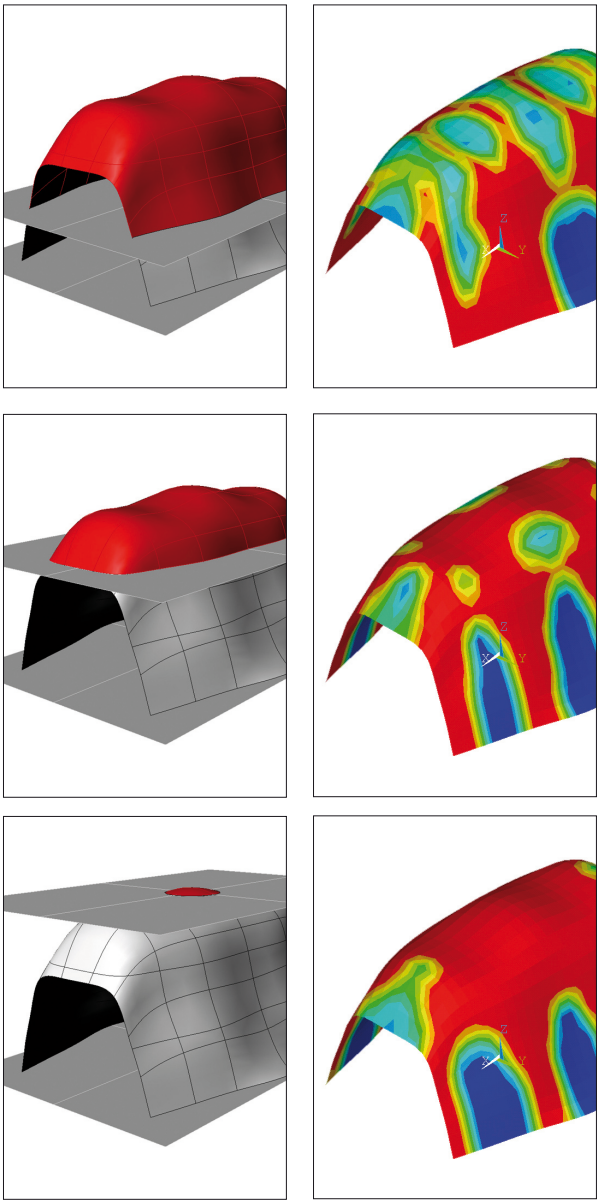


Abb. [06]) Lasteintrag entlang der z-Achse (vertikal), rot dargestellt,
(linke Reihe) und die jeweils korrespondierenden Optimierungsergebnisse.

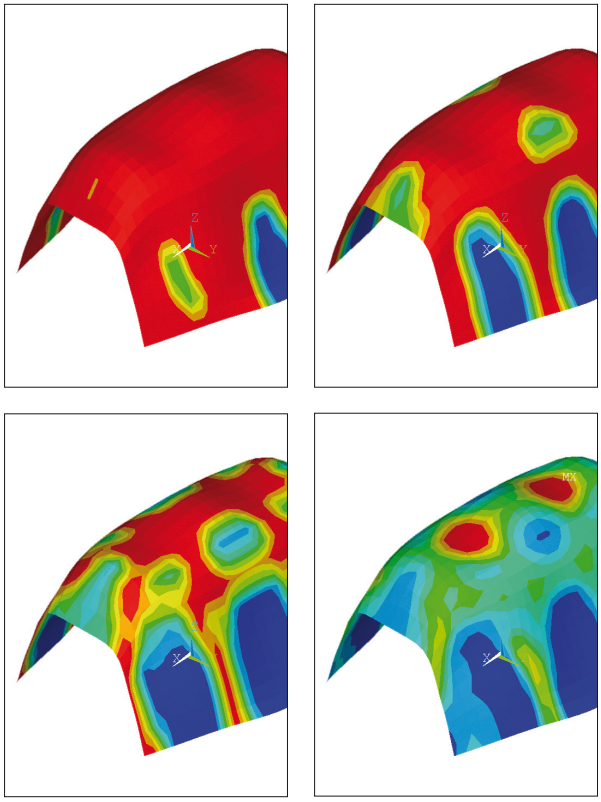


Abb. [07]) Topologieoptimierung unter Einbeziehung eines Flächenanteils von 10, 30, 50 und 70 Prozent.

Der letzte veränderbare Parameter ist der Prozentsatz der Fläche, der in den Optimierungsprozess miteinbezogen wird. Die Abbildung hierzu [Abb. 07] zeigt Beispiele für 10, 30, 50 und 70 Prozent Flächenanteile. Eine letzte Möglichkeit schließlich das Ergebnis der Topologieoptimierung zu beeinflussen, ist die Lage einer Fläche im Raum zu verändern, wodurch sich sowohl Auflager als auch Lasteintrag verändern, wie es in einem der weiter unten folgenden Beispielen zu den Konstruktionsmodulen gezeigt wird.

7.4 Parametrische Konstruktionen

Zur Umsetzung von Elementierung und Optimierungsergebnissen sind beispielhaft zwei Konstruktionsmodule entwickelt worden. Eines erzeugt eine Art vereinfachte Knochenstruktur, das zweite waffelähnliche, luftmatrazenhaft anmutende Gebilde. Mit beiden Modulen können entweder die geometrischen Daten des in *ANSYS®* erzeugten Flächennetzes weiter umgesetzt werden, bei beiden ist es zusätzlich möglich, die ermittelten Dichtewerte in die konstruktive Interpretation mit einzubeziehen.

7.4.1 Konstruktionsmodul 1 – Knochenstruktur – Basisparameter

Bei diesem geometrisch weniger komplexen Modul wird an jedem Knotenpunkt des Netzes eine Kugel plaziert. Der Radius der Kugeln *r1* wird vorab definiert und ist für jede Kugel identisch [Abb. 08 / 1]. Die Verbindung zwischen zwei Kugeln wird über fünf Kreise erzeugt, die senkrecht zu den Maschenkanten ausgerichtet sind. Das Zentrum des mittleren Kreises liegt immer auf der Mitte einer Maschenkante. Für die Position der Mittelpunkte der beiden äußeren Kreise wird ein Wert *a* definiert, für den gilt: *1 > a > 0*. Dieser Wert wird mit dem Kugelradius *r1* multipliziert. Um diesen Wert, von den beiden Endpunkten der jeweiligen Kante verschoben, liegen die Mittelpunkte dieser Kreise. Für die Platzierung der beiden übrigen Kreise wird ein weiterer Parameter *b* bestimmt, wieder gilt: *1 > b > 0*. Der Wert *b*d/2*, wobei *d = Kantenlänge - 2*r1 / 2*, bestimmt die Position der Mittelpunkt der beiden letzten Kreise [Abb 08 / 2]. Die Parameter für die Positionierung der Kreise werden auf alle Verbinderteile gleichermaßen angewendet. Auch die Radien der Kreise werden vorab definiert und gelten für alle Verbinderteile gleich: Ein Radius *r2* wird für den mittleren Kreis festgelegt, ein weiterer Radius *r3* für die beiden nächsten Kreise. Die Radien der beiden äußeren Kreise *r4* entsprechen dem senkrechten Abstand zu den Kugeln, in denen ihre Mittelpunkte liegen [Abb. 08 / 3].

Die Regeln zur Erzeugung der Knochenstrukturen sind für Netze mit Drei- und Viereckmaschen gleich, die Berechnung und Visualisierung der einzelnen Elemente wird nacheinander, Masche für Masche durchgeführt. Entsprechend einfach ist auch der dafür entwickelte Algorithmus. Da Knoten und Kanten in mehreren Elementen vorkommen können, muss bei dieser Systematik lediglich geprüft werden, ob Kugeln und Verbinderteile bereits aus einer in der Schleife vorher bearbeiteten Masche erzeugt wurden.

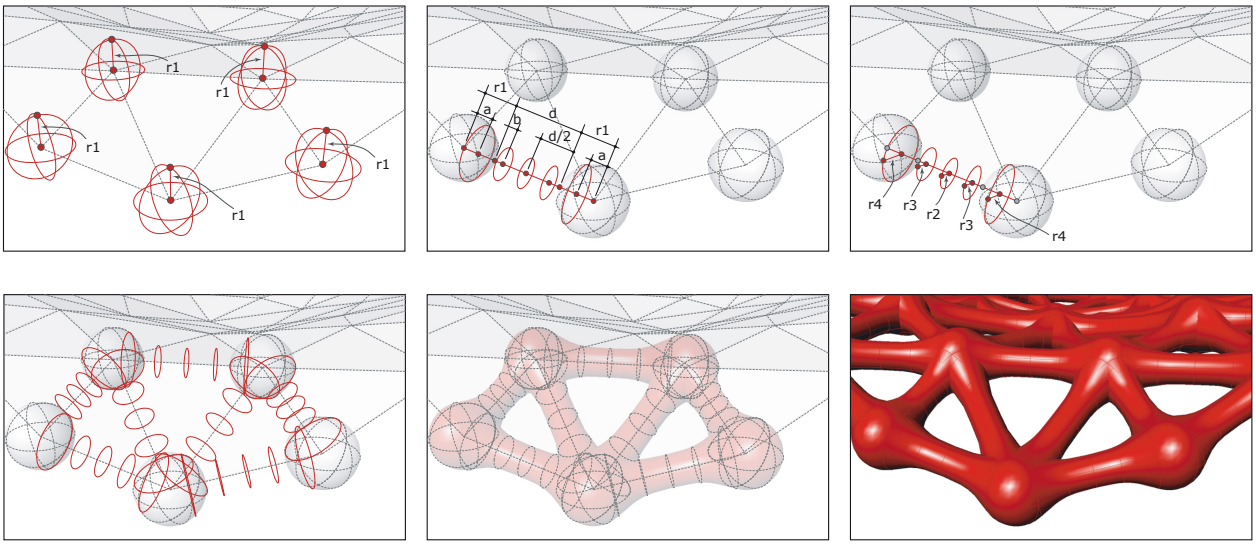


Abb. [08]) Basisparameter für Konstruktionsmodul 1.

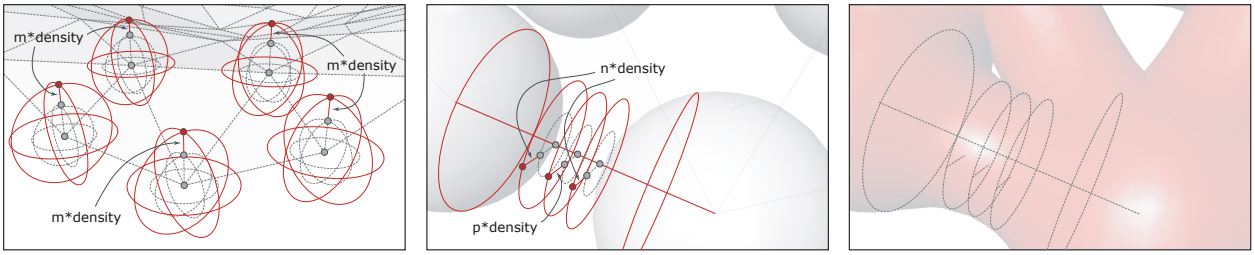


Abb. [09]) Parametereinstellungen für die Interpretation der Dichtewerte bei Konstruktionsmodul 1.

7.4.2 Konstruktionsmodul 1 – Knochenstruktur – Dichtewerte

Für die Interpretation der Dichtewerte bei der konstruktiven Durchbildung werden drei weitere Parameter definiert, die Auswirkungen auf die Radien der Kugeln und Kreise haben. Für die Kugeln wird ein Wert *m* festgelegt, der mit dem jeweiligen Dichtewert an dieser Stelle multipliziert wird. Das Ergebnis wird zum Basisparamter des Radius addiert und ergibt den neuen Kugelradius [Abb. 09 / 1]. Da die Dichtewerte variieren, werden auch die Kugeln in unterschiedlicher Größe erzeugt, ebenso die Kreise der Verbinderteile. Hier wird ein Parameter *p* für die mittleren Kreise und ein Parameter *n* für die folgenden Kreise definiert, mit dem jeweiligen Dichtewert multipliziert und zu den Basisparametern addiert, um die neuen Radien zu erhalten [Abb. 09 / 2].

Die Dichtewerte werden von *ANSYS*® den einzelnen finiten Elemente zugeordnet. Um die Werte für einen einzelnen Knoten zu ermitteln, wird geprüft, in wievielen Elemente dieser Knoten insgesamt vorkommt und der Durchschnittswert gebildet. Das gleiche Verfahren wird für die Dichtewerte der Kanten verwendet. Folglich können auch die erzeugten geometrischen Elemente einer einzelnen Flächenmaschen unterschiedlich dimensioniert sein. Da bei der Interpretation der Dichtewerte also auch die Nachbarschaftsbeziehungen der Flächenelemente eine Rolle spielen, ist dieser Algorithmus entsprechend etwas komplexer aufgebaut: Die Ermittlung der Dichtewerte für jeden Knoten und jede Kante findet in einem vorgeschalteten Durchlauf statt und wird dann durch den Algorithmus für die Interpretation der Basisparameter mit entsprechend veränderten geometrischen Eingabewerten verarbeitet.

7.4.3 Konstruktionsmodul 2 – Waffelstruktur – Basisparameter

Bei dieser zweiten, geometrisch anspruchsvolleren Interpretation der Flächennetze, wird in einem ersten Schritt jeder Knoten im Netz entlang der Normalen zur Ursprungsfläche um einen definierten Wert *d* nach oben sowie unten verschoben [Abb. 10 / 1] – es entstehen für jedes Element drei Ebenen für die weiteren konstruktiven Schritte [Abb. 10 / 2]. Als Nächstes wird für jede der Ebenen der Schwerpunkt ermittelt [Abb. 10 / 3]. Diese entsprechen den Mittelpunkten der Kreise, die in jeder dieser Ebenen erzeugt werden. Der Radius *r* aller Kreise ist gleich und wurde ebenfalls vorab festgelegt [Abb. 10 / 4]. Dann werden Schnittpunkte aus den Kreisen und den Ebenen aus Kreismittelpunkt und Kantenpunkten der Hilfsebenen erzeugt. Über diese Punkte werden Kurven aufgezogen [Abb. 10 / 5], die Kreise der mittleren Ebenen dabei in einzelne Kreisbögen aufgeteilt. Ein Kreissegment sowie zwei Kurven bilden (jeweils nach oben und nach unten) das Grundgerüst für die Erzeugung der Flächen, die die Waffelstruktur ausbilden. Für eine einzelne Fläche fehlt noch eine vierte Kurve, die über zwei Elemente, die an einer Kante anliegen, berechnet wird [Abb. 10 / 6]. Für das Schließen der Elemente an den Kanten der Ursprungsfläche sind verschiedene Optionen implementiert. Im gezeigten Beispiel wird eine *Sweep*-Operation mit einem Kreisbogen [Abb. 10 / 9] verwendet.

Da hier schon bei der geometrischen Interpretation der Flächennetze die Nachbarschaftsbeziehungen der Maschen eine Rolle spielen, ist der hier entwickelte Algorithmus umfangreicher, als der des ersten Konstruktionsmoduls. In einem ersten Durchlauf werden alle Kanten der Maschen darauf geprüft, ob sie am Rand des Flächennetzes liegen oder noch in einer zweiten Masche vorkommen. Ersteres wird im Array entsprechend indiziert, beim zweiten Fall werden gleich die notwendigen Parameter zur Erzeugung der Sattelkurve [Abb. 10 / 6] entlang der Kanten berechnet und gespeichert. Im zweiten Durchlauf werden dann, wieder Masche für Masche, alle weiteren benötigten geometrischen Parameter nach der eben beschriebenen Systematik errechnet und optional als 3D-Modell visualisiert. In einer abschließenden Schleife werden dann noch die benötigten Elemente zum Schließen der außen liegenden Kanten berechnet und – wieder optional – im *CAD*-Modell dargestellt.

7.4.4 Konstruktionsmodul 2 – Waffelstruktur – Dichtewerte

Die Dichtewerte können bei diesem Modul Einfluss auf die Dicke der Struktur und die Größe der Öffnungen nehmen – je höher der Wert, desto größer die Dicke und umso kleiner die Öffnungen. Für das Verschieben der Knoten entlang der Normalen wird ein Maximalwert *a* festgelegt und mit dem Dichtewert des Knotens multipliziert. Das Ergebnis wird zum Basisparameter addiert und ergibt die neue Verschiebelänge [Abb. 11 / 1]. Ausgangswert zur Berechnung der Radien der Öffnungen in der Waffelstruktur ist wieder der definierte Basisparameter. Hier wird der kleinste Abstand zwischen Kreis und Ebenenkante ermittelt. Diese Länge mal dem Dichtewert des Elements minus der Gesamtdistanz wird zum Basisradius dazu gezählt und ergibt somit den neuen Radius der Kreise eines Elements [Abb. 11 / 2].

Wie bei den Knochenstrukturen wird auch bei diesem Modul zur Interpretation von Drei- und Viereckssnetzen der selbe Algorithmus verwendet. Sollen die Dichtewerte in die Ausführung der Konstruktion mit einfließen, wird ebenfalls ein Durchlauf vorangestellt, der an Hand der Nachbarschaftsbeziehungen die durchschnittliche Dichte für die einzelnen Netzbestandteile kalkuliert, die dann wieder von den Algorithmen zur Berechnung der Gesamtstruktur als veränderte geometrische Eingabeparameter verarbeitet werden. Bei beiden Modulen wird zum Abschluss versucht, die einzelnen geometrischen Instanzen zu einem Flächenverbund zu schließen. Bei den Knochenstrukturen wird dies mit Hilfe *Bool'scher* Operationen versucht, bei den Waffelstrukturen durch Vereinigen der einzelnen Flächen. Kann ein geschlossener Flächenverbund generiert werden, wird automatisch eine **.stl*-Datei zur Ausgabe des Entwurfs durch ein 3D-*Rapid-Prototyping*-Verfahren erzeugt. Gelingt dies nicht, muss die Geometrie an den kritischen Stellen von Hand nachgebessert werden.

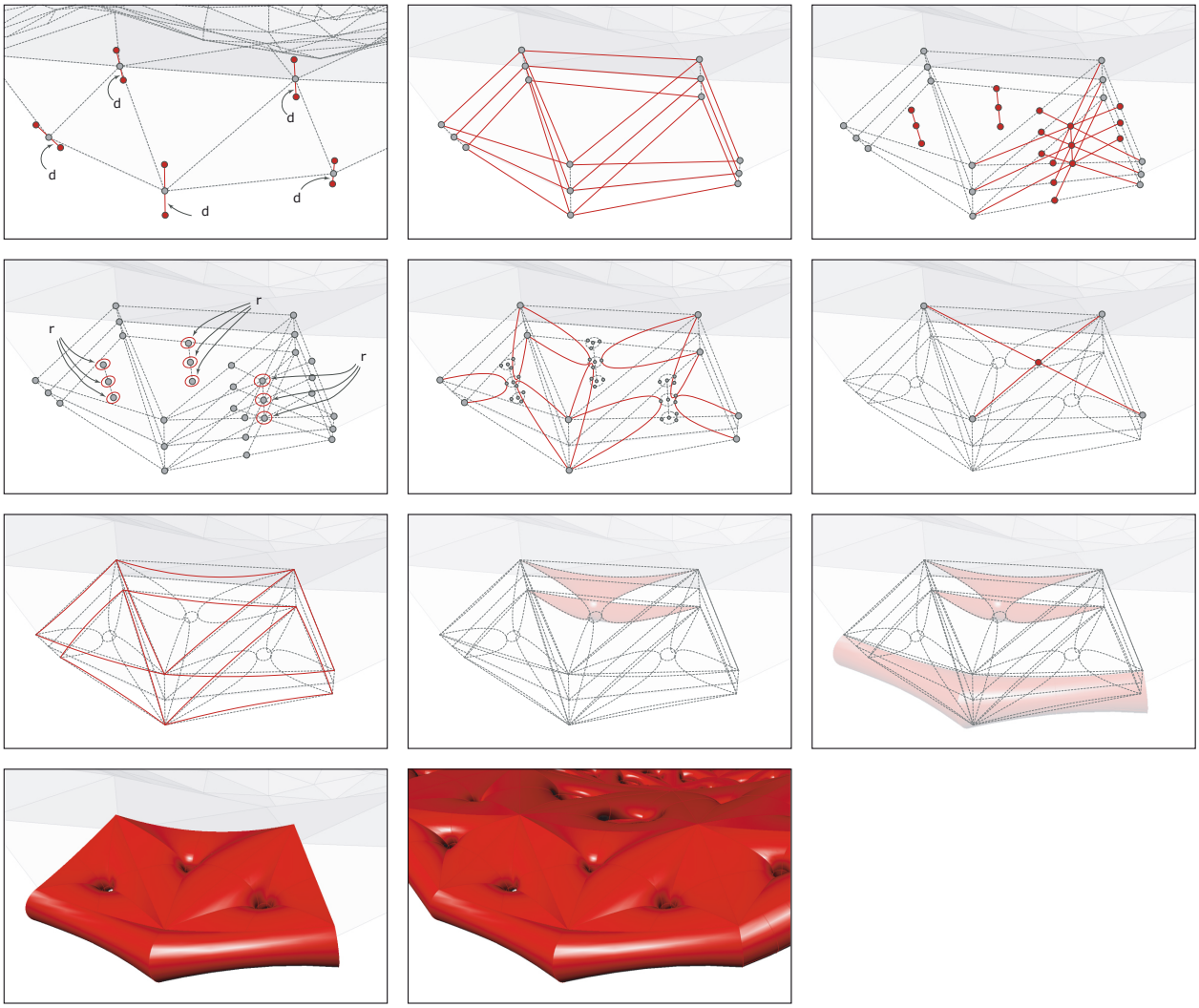


Abb. [10]) Basisparameter für Konstruktionsmodul 2.

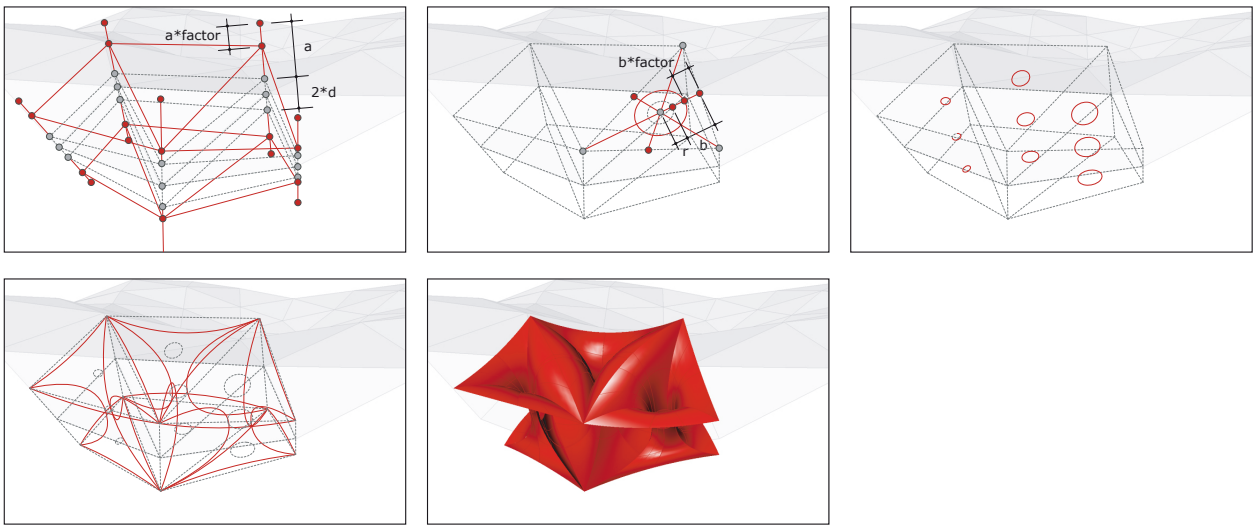


Abb. 11) Interpretation der Dichtewerte bei Konstruktionsmodul 2.

7.5 Beispiele

Auf den folgenden Seiten werden einige Beispiele gegeben, welche Art von Ergebnissen mit den algorithmischen Entwurfsräumen, die aus dem spielerischen Umgang mit der Methode der Topologieoptimierung sowie den beiden Konstruktionsmodulen entstehen, unter den eingangs definierten *Constraints* im Anwendungsfeld ‚Hocker', erzielt werden können.

Bei den gezeigten Beispielen auf der gegenüberliegenden Seite (oben) [Abb. 13] wurden jeweils nur die geometrischen Parameter von Flächennetzen mit Viereckmaschen, die alle aus einer Ursprungsfläche erzeugt wurden, interpretiert. Bei allen Beispielen wurden identische Einstellungen bei den Konstruktionsmodulen verwendet. Umgesetzt wurden Netze mit einer ungefähren Maschenkantenlänge von 20, 30 und 40mm. Die Abbildung [Abb. 12] auf dieser Seite zeigt die Interpretation von Dreiecksnetzen mit einer Kantenlänge von 30mm, ebenfalls von der selben Ursprungsfläche erzeugt und mit den selben Einstellungen bei den Konstruktionsmodulen umgesetzt. Es ist deutlich zu erkennen, wie die Struktur der Flächennetze zusammen mit der Art der konstruktiven Interpretation eine charakteristische Anmutung der Entwürfe hervorrufen.

Diese typische Eigenschaft eines algorithmischen Entwurfsraums bleibt auch erhalten, wenn die Ausgangsgeometrie geändert wird, wie in der unteren Abbildung [Abb. 14] auf der gegenüberliegenden Seite zu sehen. Es entsteht allerdings der visuelle Eindruck einer komplexeren Form, wenn geschlossene oder doppelschalig erscheinende Flächen verwendet werden. Dieser Eindruck wird verstärkt, wenn die umgesetzte Struktur eher luftig ist, also größere Einblicke auf die dahinterliegenden Bereiche zuläßt.

Die Details in der ersten Abbildung [Abb. 15] auf der übernächsten Seite geben einige Beispiele, die über Änderungen der Basisparameter erzeugt wurden.

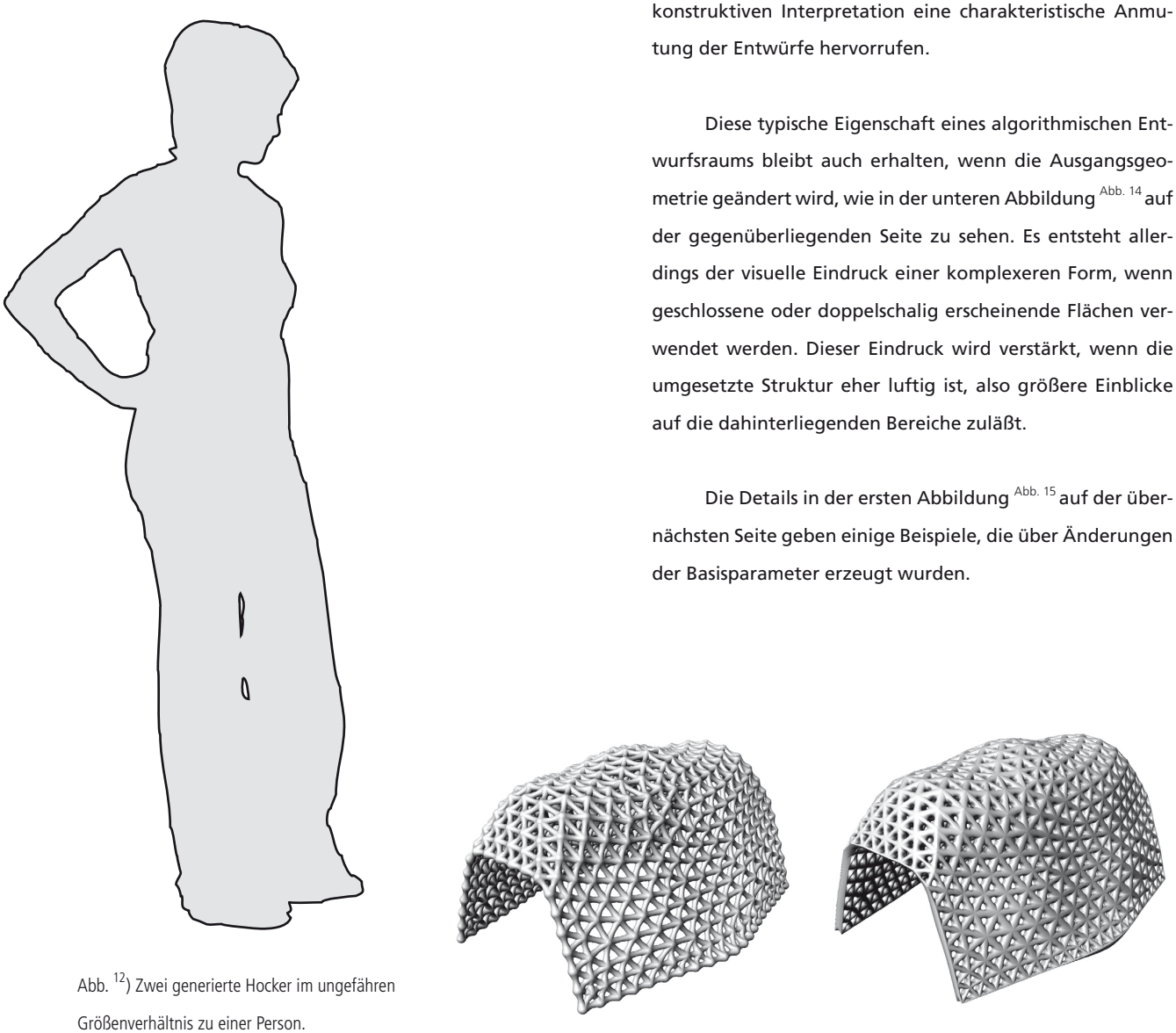


Abb. [12]) Zwei generierte Hocker im ungefähren Größenverhältnis zu einer Person.

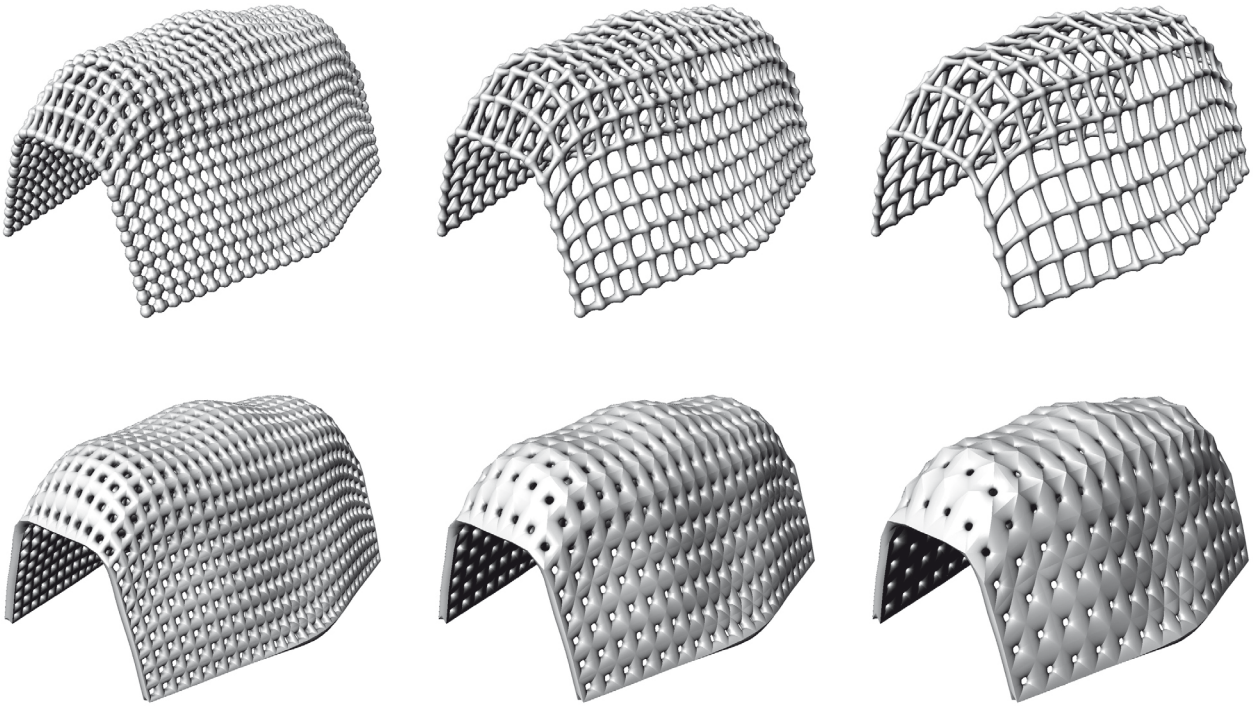


Abb. [13]) Geometrische Interpretation von Viereckmaschen unterschiedlicher Kantenlänge mit beiden Konstruktionsmodulen.

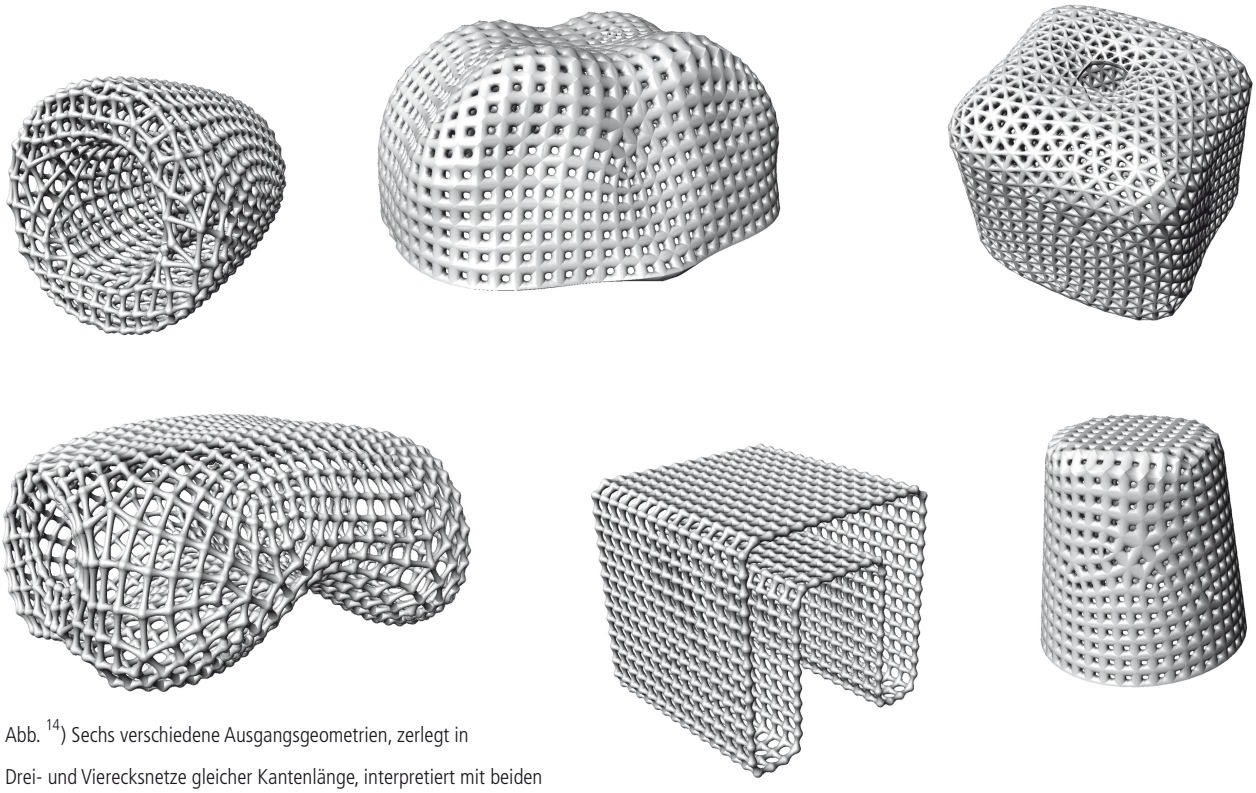


Abb. [14]) Sechs verschiedene Ausgangsgeometrien, zerlegt in Drei- und Viereckmaschen gleicher Kantenlänge, interpretiert mit beiden Konstruktionsmodulen.

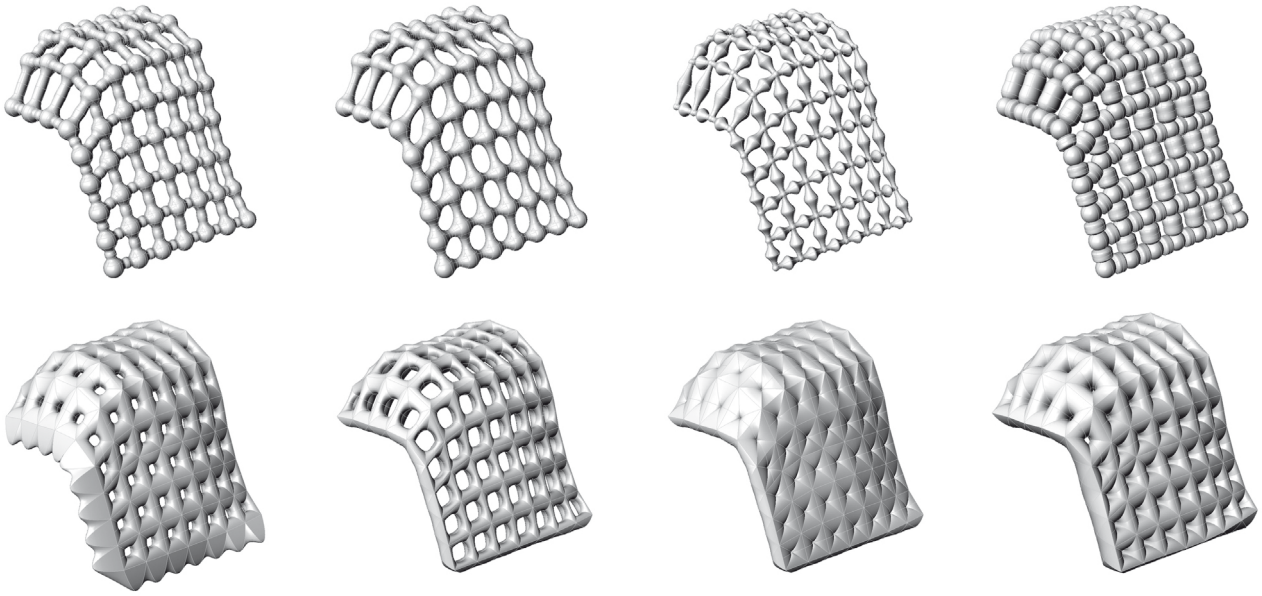


Abb. [15]) Spiel mit den Basisparametern der Konstruktionsmodule,

Beispiele.

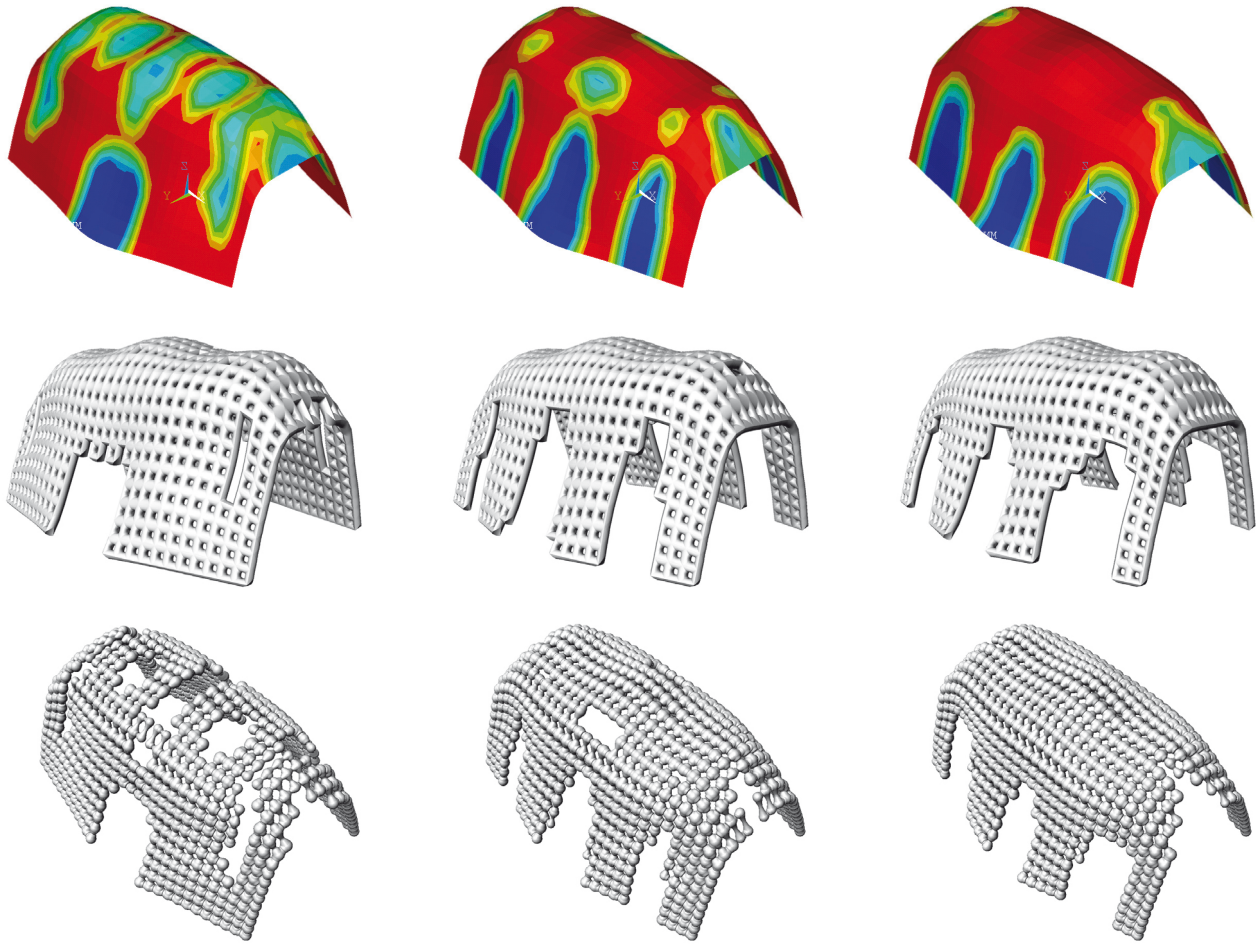


Abb. [16]) Interpretation der Dichtewerte: 1. Reihe: Ergebnisse der Topologieoptimierung; 2. Reihe: Waffelstrukturen, umgesetzt mit Basisparametern und einer Auslassung aller Elemente unter 0,15 Dichtewert; 3. Reihe: Knochenstrukturen, umgesetzt mit Basisparametern unter Einfluß der Dichtewerte und Auslassung aller Elemente unter 0,5 Dichtewert.

Die erste, einfache Art, die Interpretation der Dichtewerte mit in die Ausführung der parametrischen Konstruktion einfließen zu lassen, besteht darin, Flächenelemente des Netzes, die unter einem definierten Dichtewert liegen, bei der Umsetzung zu ignorieren. Im Ergebnis wird dadurch das Resultat der Topologieoptimierung als gestalterisches Element sichtbar, gleichzeitig ist die damit einhergehende Material- und damit auch Zeitersparnis der maßgebliche Faktor für die Produktionskosten eines solchen Hockers im 3D-*Rapid-Prototyping*-Verfahren. Die zweite Möglichkeit ist, das Aussparen von Flächenelementen mit dem Einfluß der Dichtewerte auf die Ausführung der Konstruktion zu kombinieren. Die Abbildung auf der vorigen Seite (unten) [Abb. 16] zeigt Beispiele für diese beiden Möglichkeiten. Die dritte, für Flächen und Netzauflösungen der hier gezeigten Größenordnung auch schönste Art, mit den Dichtewerten umzugehen, ist es, alle Netzelemente zu zeichnen und die Logik der Topologieoptimierung in die konstruktive Interpretation einfließen zu lassen. Die entstehenden Strukturen lassen die optimierte Fläche als kontinuierlicher Verlauf auf der Ursprungsgeometrie sichtbar werden.

Die untere Abbildung [Abb 17] zeigt, als Rückgriff auf die rein geometrische Interpretation, nochmals einen Ausschnitt einer Fläche als ABS-3D-Druck, die als Waffelstruktur ausgeführt wurde. Im Vergleich dazu sind verschiedene Varianten einer Knochenstruktur gezeigt, bei der die Dichtewerte interpretiert wurden, ebenfalls als 3D-Druck und als virtuelles Modell. In der folgenden Abbildung [Abb. 18] sind schließlich noch eine Reihe von Strukturen zu sehen, die aus verschiedenen Ursprungsflächen entwickelt wurden. Die vier im 3D-Verfahren im Maßstab 1:5 ausgedruckten Hocker wurden probehalber mit jeweils 50 N belastet, was allerdings noch nicht der höchsten mögliche Belastung entspricht. Da das Eigengewicht dieser Strukturen vernachlässigt werden kann, kann man in grober Näherung davon ausgehen, dass ein Hocker im Orginalmaßstab um den Faktor 25 mehr Gewicht trägt (Spannung = Kraft / Fläche), was in etwa 125 kg entspricht. Die Idee, Kraftfluß parametrisch als Gestaltungsmittel zu nutzen, bringt also nicht nur ästhetisch äußerst eindrückliche Strukturen hervor, sondern scheint sich, zumindest bei den ausgedruckten Beispielen, auch im statischen Sinn zu bewähren.

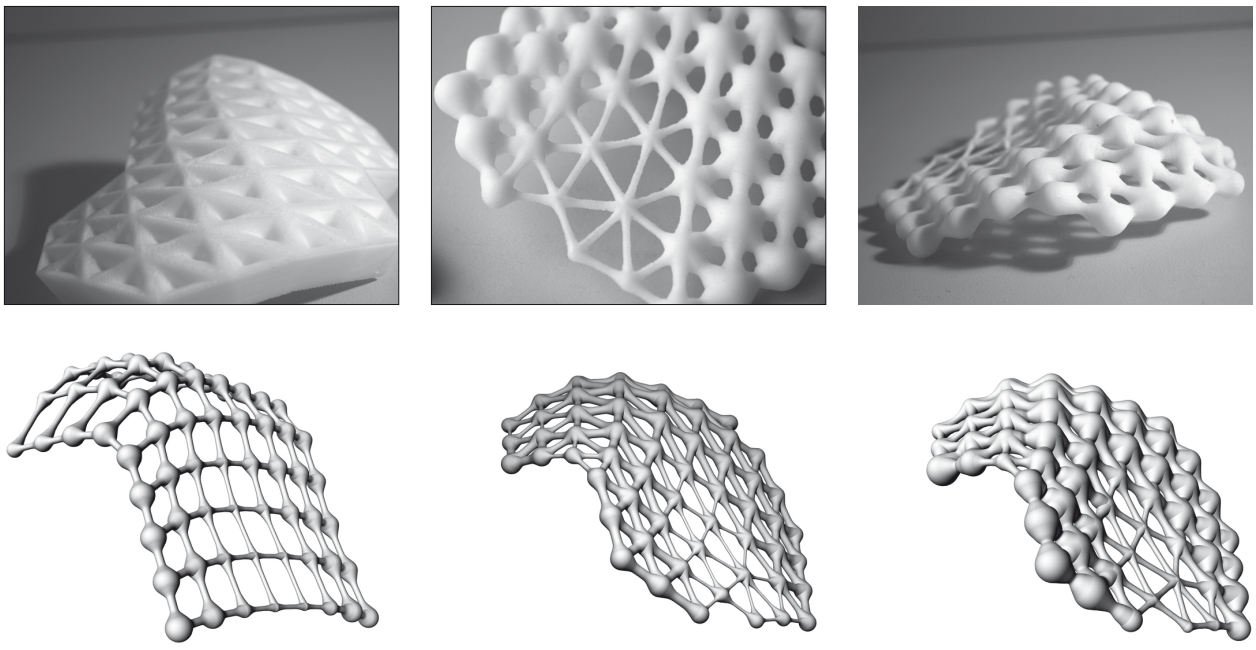
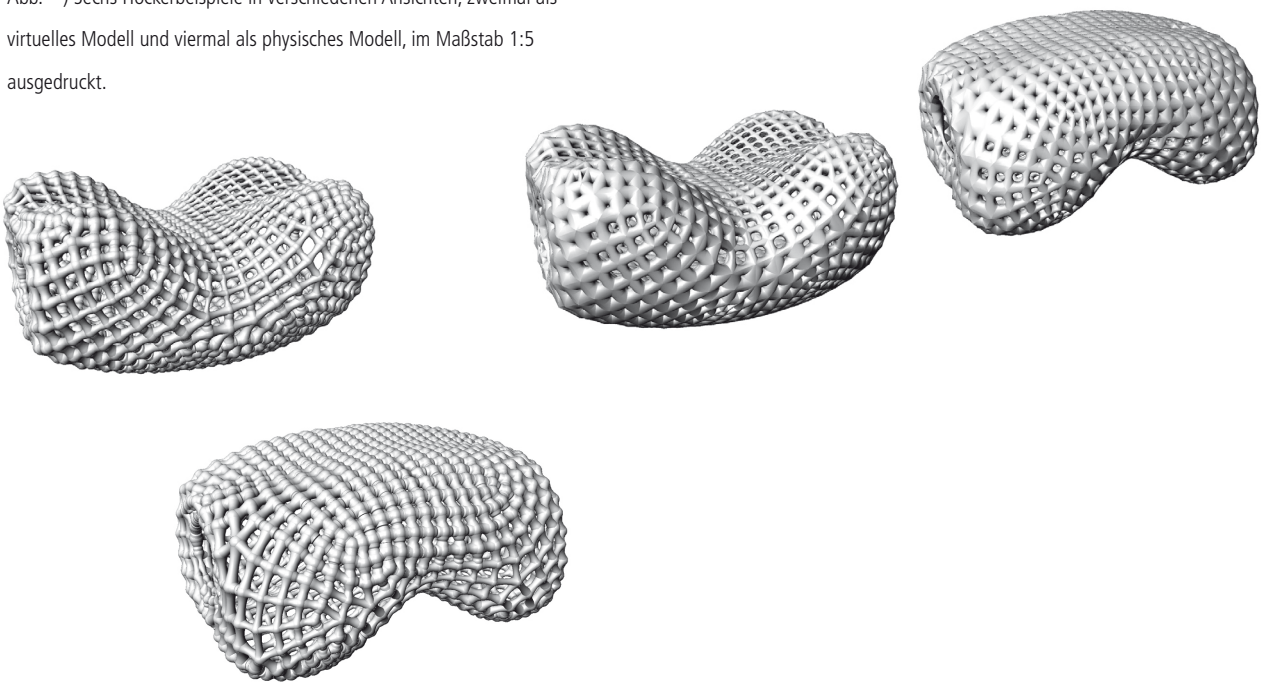
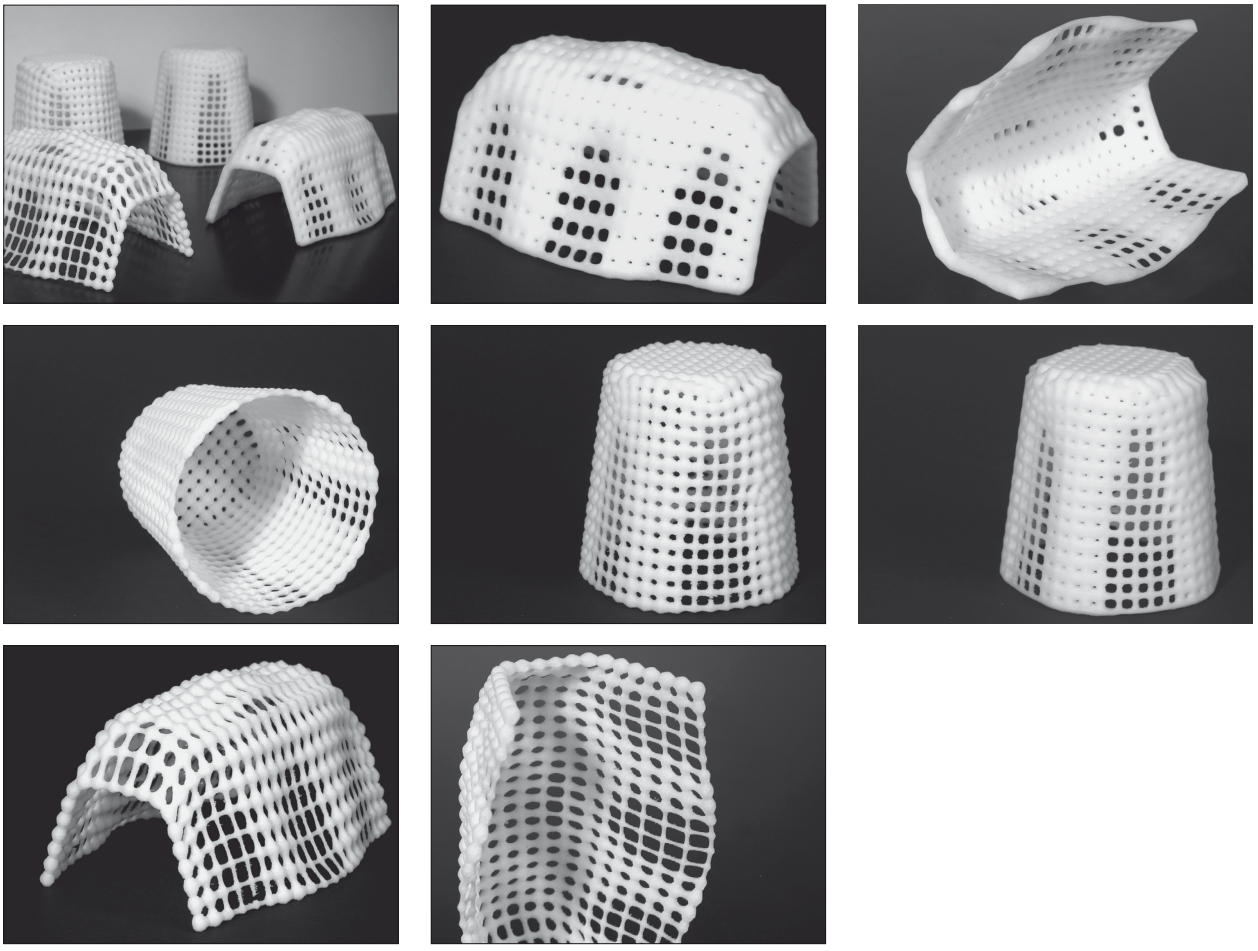


Abb. [17]) Rein geometrisch interpretierte Waffelstruktur (links oben) und Knotenstrukturen, bei denen zusätzlich die Dichtewerte in verschiedenen Varianten interpretiert wurden.

Abb. [18]) Sechs Hockerbeispiele in verschiedenen Ansichten, zweimal als virtuelles Modell und viermal als physisches Modell, im Maßstab 1:5 ausgedruckt.





8. Entwurfsräume 2 – Evolutionäre Flächenstrukturoptimierung und parametrisches Design

8.1 Constraints und Beschreibung

Diese algorithmischen Entwurfsräume kombinieren Methoden des parametrischen Design mit der evolutionären Optimierung von Flächennetzen. Die eingangs definierten Rahmenbedingungen entsprechen denen der zuvor beschriebenen Experimente:

1) Modellierung einer Fläche per Hand in einem *CAD*-System,
2) eine räumlichen Struktur wird ausgehend von der Fläche entwickelt,
3) es wird eine einzelne Fläche bearbeitet,
4) die wieder als symbolisches Modell dient.

In Abweichung von den Entwurfsräumen zuvor wurde darüber hinaus festgelegt:

5) Die Umsetzung der Struktur soll über 2D-*CNC*-Bearbeitungsverfahren (wie Wasserstrahlschneiden oder Lasern) und über einfache *CNC*-3-Achs-Fräsanwendungen möglich sein.

Diese Bedingung ermöglicht eine relativ unproblematische und günstige Fertigung der erzeugten Bauteile in *File-To-Factory* - Prozessen – einfache Datenübertragung, kurze Bearbeitungszeiten, wenig Fehlerquellen. Die Konsequenz daraus ist allerdings, dass alle Teile auf planaren oder zumindest auf ohne Stauchung und Dehnung abwickelbaren geometrischen Elementen basieren müssen. Dies ist bei bestimmten Flächentypen wie Tori oder Sphären problemlos realisierbar, aber gerade bei Freiformflächen mitunter problematisch und erfordert den Entwurf entsprechend konzipierter Konstruktionssysteme, die wiederum einen zum Teil erheblichen Programmieraufwand nach sich ziehen. Um diesen Aufwand in einem überschaubaren Rahmen zu halten, wurden zwei weitere Bedingungen definiert:

6) Anwendungsgegenstand sind mittelgroße Tragstrukturen, die ohne zusätzliche funktionale Anforderungen wie Bedeckung, Raumprogramm, usw. umgesetzt werden.
7) Die konstruktive Umsetzung dieser Strukturen entwickelt sich im Wesentlichen aus einem Detail, das parametrisch variiert und den verschiedenen Flächen angepasst wird.

8.2 Prozesskette

Auch diese Prozesskette beginnt mit der Modellierung einer einzelnen *NURBS*-Fläche, die in der *CAD*-Software *Rhinoceros*® vorliegen muss, damit die generativen Programme ausgeführt werden können.

Der erste Schritt im Ablauf ist die Erzeugung eines Flächennetzes aus Viereckmaschen, das aus den uv-Parametern der Ursprungsfläche abgeleitet wird. Hier kann zwischen zwei Wegen gewählt werden: Einmal können zufällige Netze generiert werden, die dann durch einen genetischen Algorithmus nach bestimmten geometrischen und mechanischen Parametern optimiert werden. Als Ergebnis erzeugt die evolutionäre Optimierung vier Textdateien. Die erste enthält den binär codierten Genotyp des besten Flächennetzes, die zweite die Parametereinstellungen, mit denen die Genotypen erzeugt wurden, die dritte die Einstellungen, mit denen die evolutionäre Flächenstrukturoptimierung durchgeführt wurde und die vierte Angaben darüber, welcher Fitnesstest mit welchen Parametern angewendet wurden und wie die besten Testergebnisse sind. Als Arbeitshilfen bzw. zur Visualisierung des Evolutionsfortschritts können optional die jeweils besten Flächennetze einer Generation in definierten Abständen sowie das beste resultierende Netz gezeichnet werden. Weiterhin ist es möglich, vom besten Netz den detaillierten Status [Abb. 01] für jeden durchgeführten Fitnesstest visualisieren zu lassen und die zugehörigen Graphen auszugeben, die den Evolutionsfortschritt [Abb. 02] erkennen lassen.

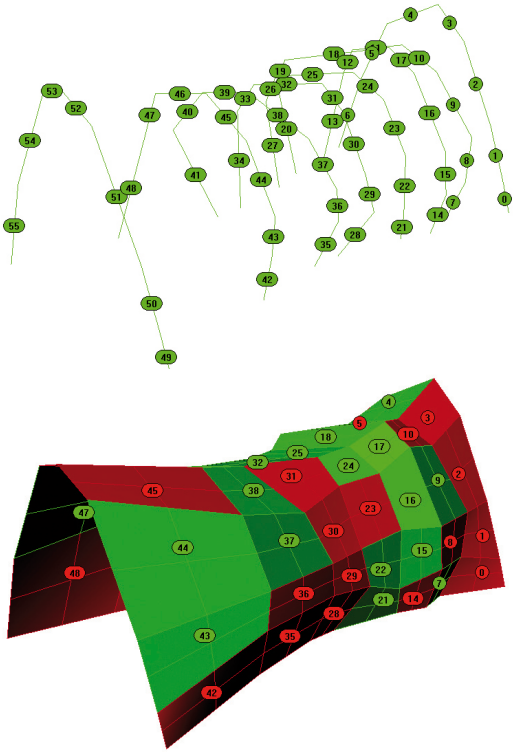


Abb. [01]) Beispiele für die Visualisierung des Status für Fitnesstests - grün bedeutet, das entsprechende Element ist innerhalb, rot es ist außerhalb der gewünschten Toleranz.

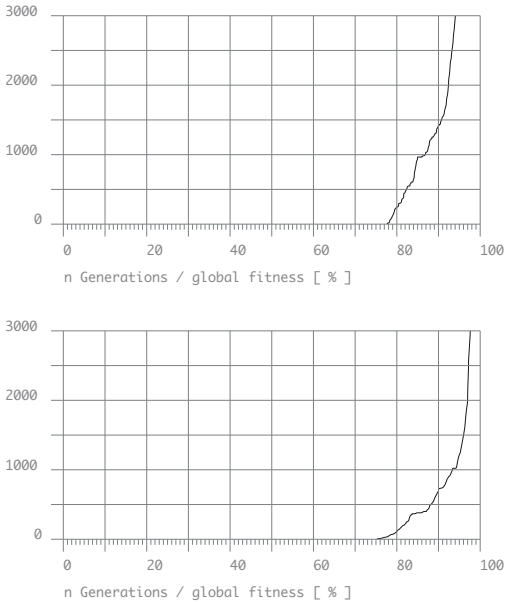


Abb. [02]) Beispiele für verschiedene Verläufe des Evolutionsfortschritts.

Der zweite Weg besteht darin, auf eine Optimierung zu verzichten. Dann wird ein regelmäßiges Viereckssnetz erzeugt, das durch eine gleichmäßige Aufteilung der uv-Parameter entsteht. Dieses Netz wird zwar nicht optimiert, aber trotzdem durch die Fitnesstest geprüft. Die Datenausgabe sowie die Möglichkeiten der Visualisierung sind die selben, wie bei der evolutionären Optimierung. Der komplette Ablauf von Flächenelementierung bzw. -optimierung ist auf *Seite 127* Abb. 03 als *Input* / *Output*-Schema dargestellt.

Um die parametrische Konstruktion auszuführen, werden die Genotypdaten, die Genotypinformationen und die zugehörige Ursprungsfläche benötigt. Es wird eines der vorhandenen Module gewählt und dafür die gewünschten Konstruktionsparameter gesetzt. Der Ablauf beginnt damit, den ausgewählten Genotyp wieder als Flächennetz zu interpretieren. Dazu werden die binären Werte wieder in die ursprünglichen uv-Parameter übersetzt und daraus wieder die kartesischen Koordinaten der Netzknoten abgeleitet. Ausgehend von den Koordinaten der Knoten werden schließlich die einzelnen Bauteile der Tragwerke berechnet und die erzeugte Struktur im gewünschten Detaillierungsgrad visualisiert. Der Programmablauf endet mit der Ausgabe der Parametereinstellungen für die konstruktive Umsetzung. Als Option können jeweils noch die benötigten Fertigungs- und Montagedaten (Teilelisten, Dateien mit Schnittpfaden, etc.) erzeugt und gespeichert werden. Auch dieser Teil der Prozesskette ist als *Input* / *Output*-Schema dargestellt Abb. 04.

Nach diesen Schritten ist ein Ablauf der generativen Prozesskette abgeschlossen. Die wichtigsten Gestaltungsgrößen, die diese Entwurfsräume enthalten, sind Änderungen an der Ausgangsfläche, die Parametereinstellungen für Flächenelementierung und -optimierung sowie die Einstellungen für die automatische Ausführung der parametrisch angelegten Konstruktion. Auch hier ergeben sich neue Entwurfsräume nur dann, wenn neue Formen der konstruktiven Interpretation entwickelt und programmiert oder andere Arten von Flächenaufteilungen implementiert werden.

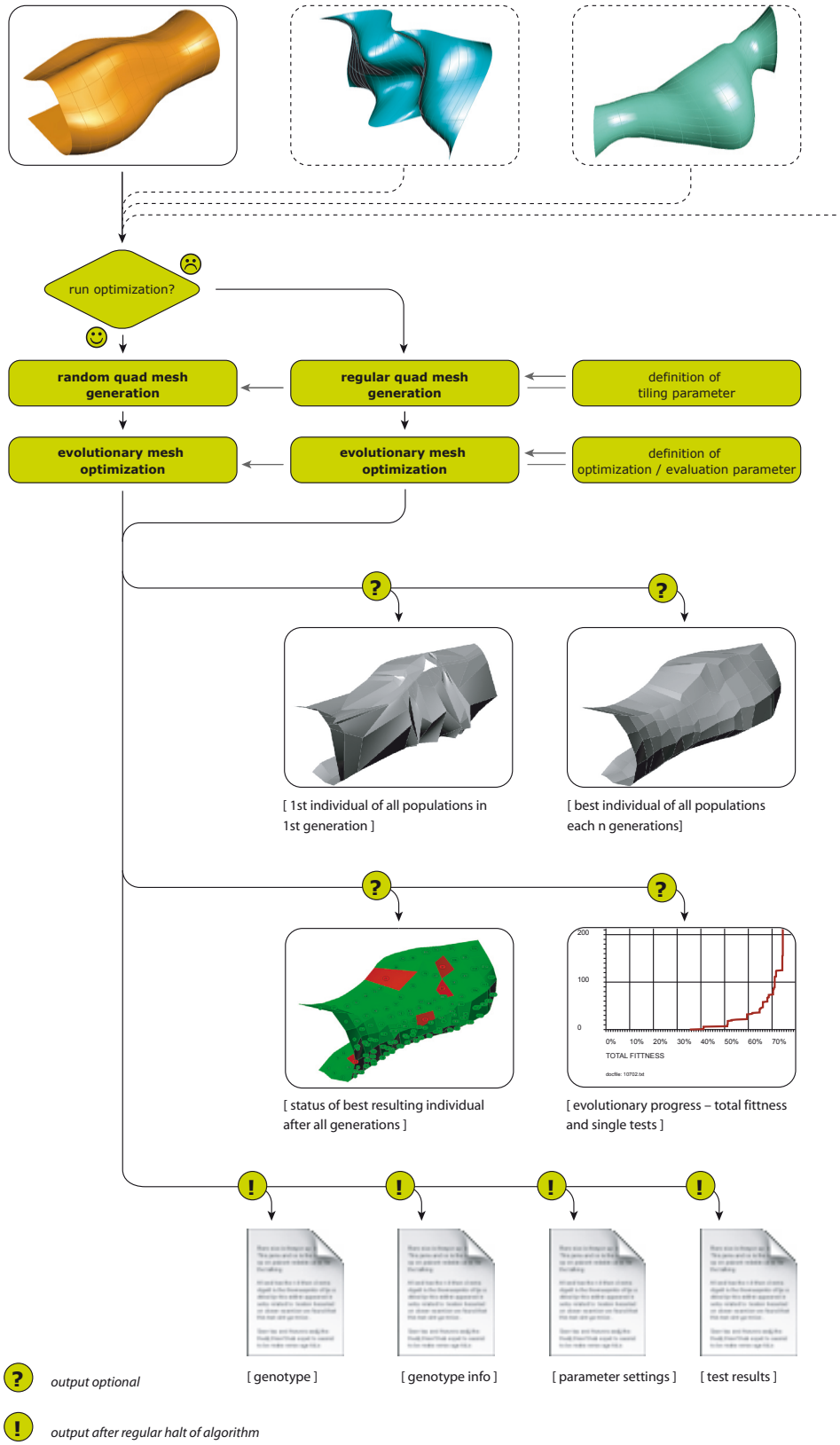


Abb. [03]) Schematische Darstellung der generativen Prozesskette – Flächenelementierung und evolutionäre Flächennetzoptimierung.

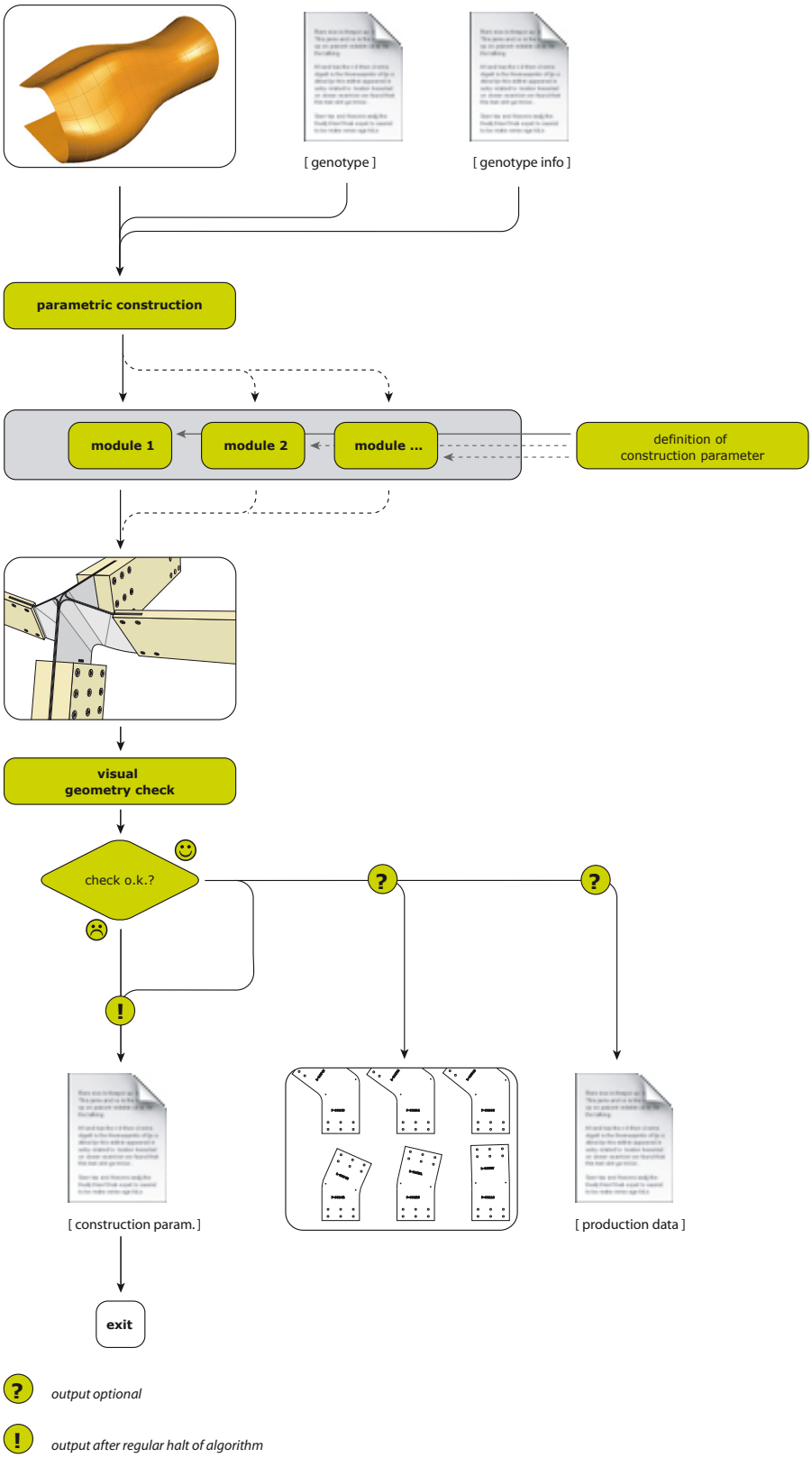


Abb. [04]) Schematische Darstellung der generativen Prozesskette – konstruktive Interpretation.

8.3. Flächenelementierung

Aufbau und Funktionsprinzip eines genetischen Algorithmus sind weitgehend unabhängig vom Gegenstand der Bearbeitung – Basis ist immer ein Variablensatz, bei dem einzelne Parameter variiert werden. Im Prinzip können also auch mit dem hier entwickelten Algorithmus Flächennetze optimiert werden, die durch unterschiedliche Elementierungsmethoden erzeugt wurden. Umgesetzt wurde eine Unterteilung in die konstruktiv günstigen Viereckmaschen, bei der nur drei verschiedene Knotenwertigkeiten im Flächennetz entstehen: Zwei- und dreiwertige Knoten an den Ecken bzw. Rändern sowie vierwertige Knoten in der Fläche. Netze aus Viereckmaschen (*quad meshes*) haben zudem den Vorteil, dass sie sich in einfachen *Array*-Strukturen abbilden und bei Bedarf natürlich auch (innerhalb der gleichen Strukturen, mit zusätzlichen Zählersprüngen in einer Schleife) als Dreiecksmaschen behandeln lassen.

8.3.1 Prinzip der Flächenaufteilung

Aus entwerferischer Perspektive mag es wünschenswert sein, dass eine Ursprungsform durch eine Elementierung möglichst genau abgebildet wird und in irgendeiner Form Eigenschaften der Ursprungsfläche, wie zum Beispiel Krümmungswechsel, wiederspiegelt. Dazu ist es zunächst nötig, dass sich die Knoten des Flächennetzes immer genau auf der Ursprungsfläche befinden. Zudem verringert sich die Abweichung des Netzes von einer Fläche, je größer die Anzahl der Viereckmaschen wird. Entsprechend ist die Unterteilung am uv-Parameterraum einer Fläche orientiert: Der maximale Parameterwert in u- Richtung sowie der maximale Wert in v-Richtung werden gemäß der definierten Anzahl von Maschen für die jeweilige Flächenrichtung aufgeteilt. Diese Aufteilung kann gleichmäßig sein, wodurch ein regelmäßiges Flächennetz entsteht, das dann für Evaluierungszwecke oder für Fälle verwendet wird, bei denen keine Optimierung sondern lediglich eine Elementierung vorgesehen ist. Durch eine zufällige Unterteilung des uv-Parameterraums werden die Flächennetze für die Startgeneration des genetischen Algorithmus erzeugt [Abb. 05].

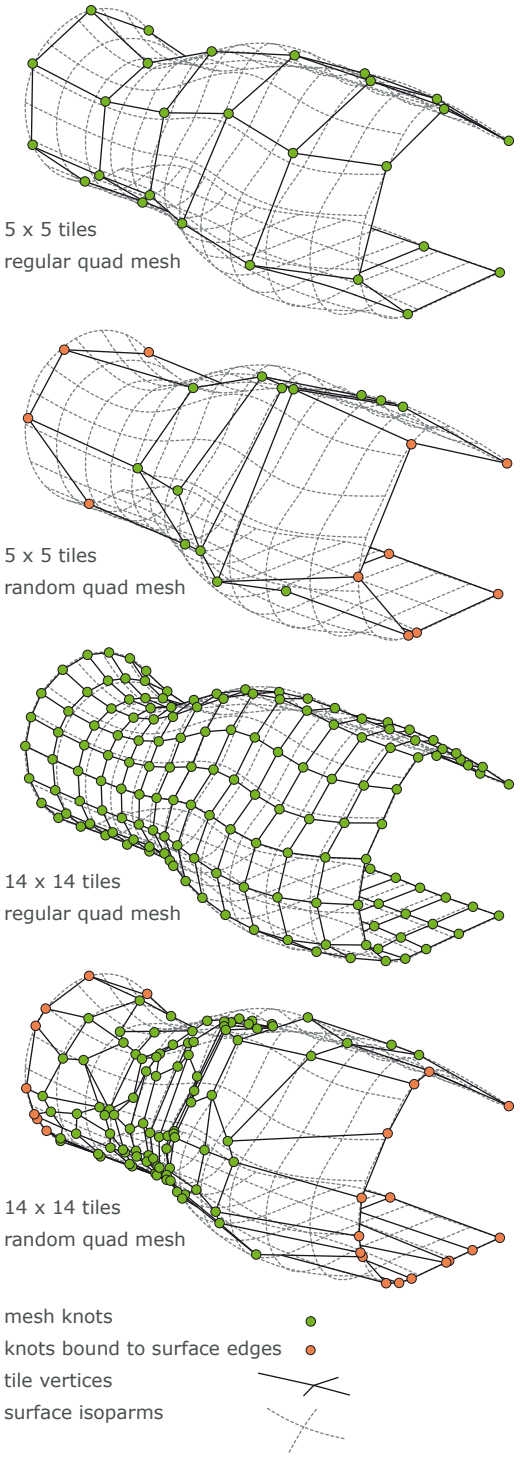


Abb. [05]) Aufteilungsbeispiele: Regelmäßig und zufällig erzeugte Flächennetze mit unterschiedlichen Maschenzahlen.

Für die evolutionäre Optimierung jedoch kann es – je nach Testkriterien – hinderlich sein, wenn sich die Knoten immer genau auf der Ursprungsfläche befinden. Damit die Flächennetze variabler auf unterschiedliche Optimierungs-

anforderungen reagieren können, ist optional noch eine kontrollierte Abweichung der Knoten auf ihrer jeweiligen Normalen zur Fläche möglich. Veränderungen in den uvn-Parameterwerten der Knoten durch die genetischen Operatoren sorgen dann dafür, dass die Netze im Fortlauf des evolutionären Prozesses unterschiedliche Formen annehmen und so, entsprechend den definierten Testkriterien, allmählich zu einer gewissen Ausprägung hin konvergieren. Dieses Aufteilungsprinzip ist in zwei unterschiedlichen Varianten umgesetzt worden:

Aufteilung - Variante 1 – Direkte Ableitung der Knoten aus dem uv-Parameterraum

Bei dieser Aufteilungsart werden für die Flächennetze der Stargeneration die uv-Parameter der einzelnen Knoten direkt aus dem uv-Parameterraum der Fläche abgeleitet. Dazu wird zunächst für jede Reihe von Knoten in u-Richtung ein zufälliger Wert aus dem u-Parameterraum der Fläche genommen und die entstehende Liste dann nach aufsteigenden Werten sortiert. Dies geschieht so lange, bis für jede Reihe die u-Koordinaten der Knoten bestimmt sind. Anschließend werden die v-Koordinaten eines jeden Knotens auf dem gleichen Weg ermittelt. Im Ergebnis entsteht ein sehr flexibles Flächennetz, bei dem die einzelnen Maschen über sehr große Bereiche der Ursprungsfläche spannen, aber auch viele Maschen in einem relativ kleinen Bereich kummulieren können, wie in nebenstehender Abbildung Abb. 06 / oben angedeutet. Die Kehrseite dieser Flexibilität besteht darin, dass trotz der Vorsortierung der uv-Werte die zufällig erzeugten Netze der Startgeneration des genetischen Algorithmus sehr inhomogen sind und viel von der evolutionären Kraft schon für das *Constraint-Handling* verwendet wird: Die Netze müssen sich so entwickeln, dass keine sich überschneidenden oder stark gegeneinander verkippten Maschen mehr vorhanden sind, die durch die Konstruktionsmodule kaum oder gar nicht interpretiert werden können. Ein Unterdrücken dieser Eigenschaften durch entsprechende Fitnesstests hat sich als nur bedingt praktikabel erwiesen, deshalb wurde eine zweite Aufteilungssystematik entwickelt, die dieses Problem etwas entschärft.

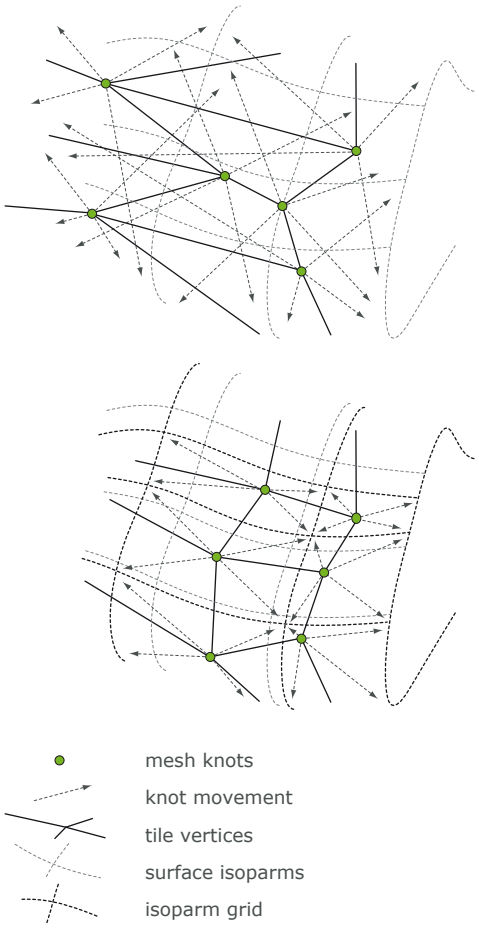


Abb. 06) Ableitung der uv-Knotenkoordinaten aus einer Fläche: Variante 1 (oben) und Variante 2.

Aufteilung - Variante 2 – Einbindung der Knoten in ein uv-Flächenraster

Bei dieser Aufteilungsart werden die Netze der Stargeneration mit Hilfe eines zusätzlichen Rasters erstellt, das auf der Fläche aufgespannt wird. Die Position der einzelnen Rasterlinien in u- sowie v-Richtung entsprechen wieder zufälligen Werten aus dem u- beziehungsweise v-Parameterraum. Jeder einzelne Knoten liegt jetzt innerhalb einer eigenen Rasterzelle, die Position der Knoten wird, wieder durch einen Zufallswert, relativ zu den Grenzen der zugehörigen Zelle bestimmt. Dadurch wird gewährleistet, dass die Knoten immer in der selben Reihenfolge im Netz liegen, was von vorneweg schon Flächennetze erzeugt, die relativ homogen sind und so – zumindest im Prinzip – auch immer baulich umgesetzt werden können Abb. 06 / unten.

Randknoten der Flächennetze Abb. 07

Damit die Außenkanten der Fläche möglichst genau kontrollierbar bleiben (z.B. zur Anbindung an Fundamente, zur Einbettung einer Großform in einen bereits bestehenden architektonischen Kontext, etc.), können die Randknoten der Flächennetze optional an die Aussenkanten der Ursprungsfläche ‚gebunden' werden. Dazu werden, je nach Lage der Randknoten, die entsprechenden u- oder v-Parameter auf den minimalen oder maximalen Wert gesetzt, folglich können die genetischen Operatoren nur Bewegungen der Randknoten entlang der Flächenkante hervorrufen, an die diese Knoten gebunden sind. Diese Option ist bei Aufteilungsvariante 1 und Variante 2 identisch.

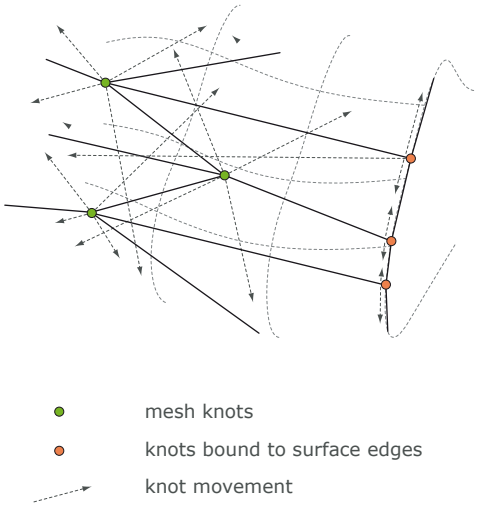


Abb. 07) Fixieren der Randknoten an den Kanten der Ursprungsfläche, dargestellt an Variante 1.

Abweichung von der Ursprungsfläche Abb. 08

Für eine mögliche Abweichung der Netzknoten von der Ursprungsfläche wird ein Wert definiert, der die maximale Distanz eines Knotens senkrecht zur Ursprungsfläche bezeichnet. Nachdem die uv-Parameter der Knoten festgelegt sind, werden daraus die entsprechenden kartesischen Koordinaten abgeleitet. Für die Startgeneration wird jedem Knotenpunkt eine zufällige Abweichung zugewiesen, die zwischen dem negativen und dem positiven Maximalwert liegt, anschließend wird noch jeder Knotenpunkt um die zugewiesene Distanz senkrecht zur Fläche verschoben und so seine endgültige Position bestimmt. Je nach Wert kann eine Verschiebung zur Innen- oder zur Aussenseite der Fläche erfolgen. Liegt der Zufallswert genau zwischen dem positiven und negativen Maximalwert, wird der entsprechende Knoten nicht verschoben. Da die Abweichung ebenfalls Bestandteil des Genotyps eines Flächennetzes ist, kann sie durch die genetischen Operatoren im Laufe eines Evolutionsprozesses innerhalb der definierten Grenze verändert werden. Soll für alle Knoten eine Abweichung von der ursprünglichen Fläche ausgeschlossen sein, wird der Wert für die maximal erlaubte Abweichung auf 0 gesetzt. Dieses Verfahren ist bei beiden Aufteilungsvarianten implementiert. Die Abweichung von der Fläche wird nicht bei Randknoten angewandt, die an die Flächenkanten gebunden sind. Werden regelmäßige Flächennetze erzeugt, liegen alle Knoten immer genau auf der Fläche, die Einstellung für die Abweichung wird ignoriert. Auch die Option, Randknoten zu fixieren, ist bei regelmäßigen Netzen obsolet: Hier liegen die äußeren Knoten ohnehin auf den Begrenzungen der Ursprungsfläche.

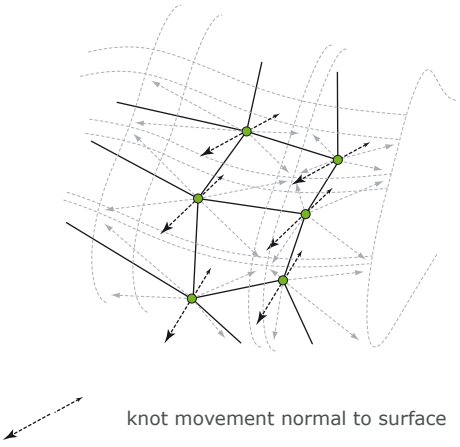


Abb. 08) Abweichung der Knoten senkrecht zur Ursprungsfläche, dargestellt an Variante 2.

8.3.2 Erweiterungsmöglichkeiten

Für das beschriebene Elementierungsverfahren sind einige Ergänzungen denkbar. Einmal könnte die Anzahl der Maschen in u- und v-Richtung, die jetzt bei beiden Aufteilungsvarianten während eines Durchlaufs des genetischen

Algorithmus gleich bleibt, auch während des Evolutionsvorgangs verändert werden, also ebenfalls zum Bestandteil des Genotyps eines Flächennetzes werden. So würde sich während des Optimierungsprozesses von selbst die geeignete Anzahl an Zellen einstellen. Zum Zweiten könnten alle einzelnen Viereckmaschen in weitere, immer feinere Subnetze unterteilt werden [Abb. 09]. Beide Optionen zusammen genommen könnten die Fähigkeit der Netze, sich auf unterschiedliche Flächen und Optimierungskriterien einzustellen, stark verbessern. Dies gilt besonders für Flächen, die häufige und extreme Krümmungswechsel aufweisen.

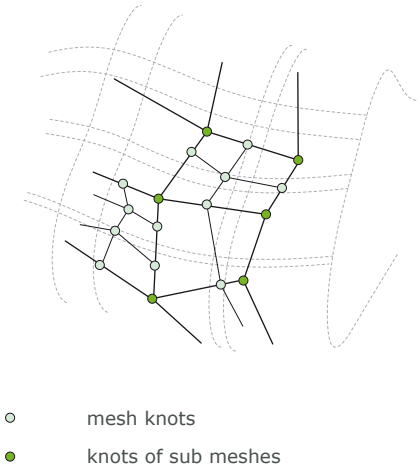


Abb. [09]) Mögliche weitere Aufteilungsvariante: Unterteilung von Flächenmaschen in weitere Subnetze.

Diese aus entwerferischer Sicht und aus dem Blickwinkel der Optimierung interessanten Erweiterungen der Anpassungsfähigkeit der Flächennetze steht jedoch gegenüber, dass sich dadurch der Suchraum (Anzahl der Lösungen) stark vergrößern würde und damit auch erheblich höhere Anforderungen an die Rechenleistung gestellt wären. Zudem müssten weitere Fitnesstests implementiert werden, um solche Flächennetze kontrollieren zu können. Beide Optionen würden auch Genotypen von variabler und unterschiedlicher Länge bedeuten, was zwar kein grundsätzliches Problem ist, aber vor allem bei den Konstruktionsmodulen einen erheblich höheren Programmieraufwand nötig gemacht hätte. Insofern wurde entschieden, diese Erweiterungen im Rahmen dieser Arbeit nicht zu experimentieren.

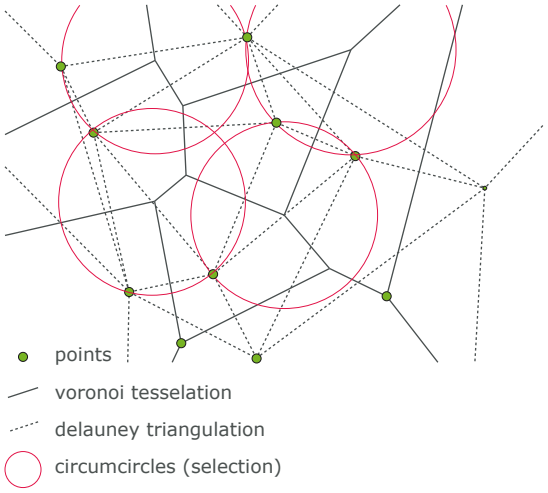


Abb. [10]) Ebenfalls als Ausgangspunkt für eine evolutionäre Optimierung geeignet: Voronoi-Polygone und Delauney-Dreiecksnetz.

Es ist auch möglich, Flächennetze, die über andere Methoden [Abb. 10] erzeugt wurden, zu optimieren, etwa ein Dreiecksnetz, aus einem *Delauney*-Algorithmus [01] oder eine Elementierung über das *Voronoi*-Diagramm [02]. Dazu sind

---

01) Durch eine Delauney-Triangulation (nach Boris N.Delauney, 1934) wird aus einer vorgegebenen Punktemenge ein Dreiecksnetz erstellt, wobei alle Dreiecke die sogenannte Umkreisbedingung erfüllen, d.h. der Umkreis eines Dreiecks darf keine weiteren Punke der Menge enthalten. Dieses Verfahren ist auch im 3D-Raum anwendbar, aus der Umkreisbedingung wird die Umkugelbedingung.

02) Das Prinzip der Konstruktion von Voronoi-Polygonen (nach Georgi F. Woronoi) ist, auf den Verbindungslinien von jeweils zwei Punkten einer Punktemenge die Mittelsenkrechten zu konstruieren und alle so erzeugten MIttelsenkrechten miteinander zu verbinden. Es entsteht eine Polygon-Fläche mit Polygonen unterschiedlicher Kantenzahl. Die Eckpunkte der Polygone entsprechen den Umkreismittelpunkten der Dreiecke aus der Delauney-Triangulation. Auch dieses Prinzip ist für den 3D-Raum abwandelbar.

---

verschiedene Anpassungen im Programmcode nötig: Einmal muss die Anzahl Gene (= Parameter) bei der Erzeugung der zufälligen Startgeneration angepasst werden. Zudem muss für jede neue Aufteilungsart eine Funktion, die den Genotyp in die Parameter des Flächennetzes umwandelt, ergänzt werden. Weiterer Anpassungs- bzw. Ergänzungsbedarf entstünde gegebenfalls bei den Fitnesstests, sowie den Programmmodulen, die dazu genutzt werden, die Flächennetze konstruktiv zu interpretieren: Hier müsste eine Array-Systematik verwendet werden, die eine Zuordnung von Flächenelementen, Stäben und Knoten erlaubt.

8.3.3 Limitierende Faktoren durch / für die Flächenelementierung

Als Ziel wurde bereits formuliert, dass mit Hilfe dieser Prozesskette eine möglichst breite Palette an unterschiedlichen Formen bearbeitbar sein soll. Dennoch müssen Flächen, die hier optimiert und konstruktiv umgesetzt werden sollen, einige Bedingungen erfüllen, damit eine Bearbeitung gute Ergebnisse erzielen bzw. eine Fläche überhaupt bearbeitet werden kann. Die daraus resultierenden Einschränkungen liegen in der Logik der Flächenrepräsentation im *CAD*-System und in der gewählten Konzeption der Flächenaufteilung begründet.

Einschränkungen durch die Flächenrepräsentation

Polysurfaces. *NURBS*-Flächen sind äußerst variabel und können vielfältige Formen darstellen, trotzdem hat dieses Repräsentationssystem auch seine Beschränkungen. Dort, wo die mathematische Beschreibung einer *NURBS*-Fläche nicht mehr ausreicht, um eine Form zu definieren, wird mit Flächenverbünden (*polysurfaces*), also einer Zusammenstellung einzelner Flächen gearbeitet. Da die Erzeugung eines Flächennetzes auf die Beschreibung einer einzelnen Fläche zurückgreift, können bei Flächenverbünden entweder nur einzelne Flächen aus dem Verbund bearbeitet werden [Abb. 11] oder es muss versucht werden, einen Flächenverband näherungsweise durch eine einzelne Fläche nachzubilden.

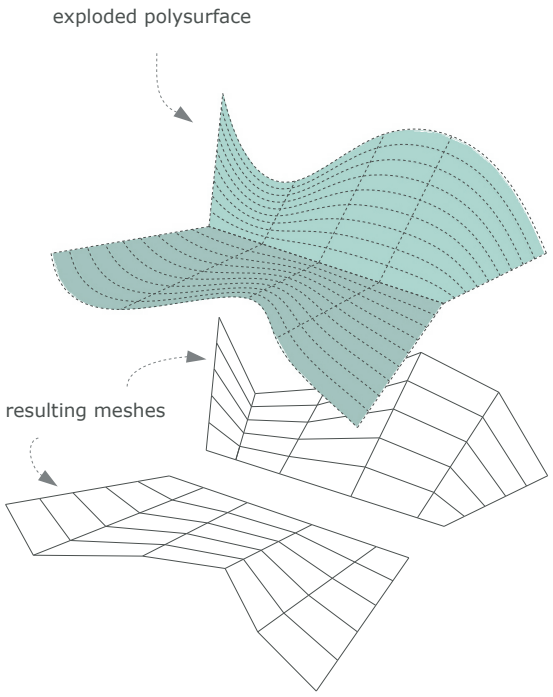


Abb. [11]) Regelmäßige Flächennetze aus den zwei Flächen eines aufgelösten Flächenverbunds.

Trimmed Surfaces. Jede *NURBS*-Fläche ist begrenzt durch vier Kanten, wenn diese manchmal am digitalen Flächenmodell wegen ihrer unter Umständen starken Verformung auch kaum erkennbar sind. Diese Kanten entsprechen dem minimalen und maximalen v-Wert der Fläche sowie dem kleinsten und größten u-Wert. Um bestimmte Formen verwirklichen zu können, werden Flächen getrimmt (geschnitten). Bei dieser Trimmung kann ein Teil abgeschnitten oder ausgeschnitten (‚Löcher') werden, es entsteht ein getrimmter und ein nicht-getrimmter Flächenteil. Trotzdem bezieht sich die *NURBS*-Beschreibung nach wie vor auf die Gesamtfläche, der weggeschnittene Teil wird nur unsichtbar, ähnlich wie bei einer Beschnittmaske bei der Bildbearbeitung. Entsprechend arbeitet die Flächenaufteilung auch immer auf der Gesamtfläche (zwischen den minimalen und maximalen uv-Werten) und bezieht so auch den weggeschnittenen, nicht sichtbaren Flächenteil mit ein. Um dies zu vermeiden, kann dieser Teil einer Fläche beispielsweise neu modelliert werden: In der unten folgenden Darstellung [Abb. 12] ist gezeigt, wie dieses Problem für das ‚Abschneiden' gelöst werden kann: Oben ist der beschnittene Teil einer

NURBS-Fläche zu sehen (erkennbar an der Lage der Isolinien) und die dazugehörige Flächenaufteilung, die über die Gesamtfläche läuft. Unten wurden aus dem getrimmten Flächenteil einzelne Kurven extrahiert und über den *Loft*-Befehl zu einer neuen Fläche geschlossen. Jetzt arbeitet die Aufteilung genau auf der Fläche. Löcher in einer Fläche sind so nicht zu lösen, hier würde gegebenenfalls eine Fitnessfunktion helfen, die dafür sorgt, dass auf den Ausschnitten in der Fläche keine Knoten platziert werden. Das Problem des Aus- oder Wegschneidens taucht im Übrigen nicht nur auf, wenn explizit der *Trimm*-Befehl eines *CAD*-Programms benutzt wird, es gibt auch Befehle, die bereits getrimmte Flächen erzeugen.

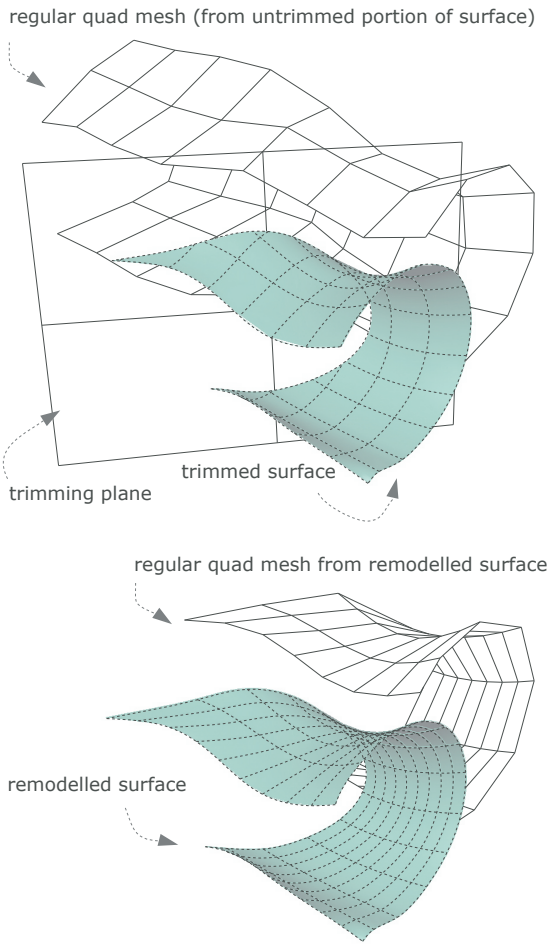


Abb. [12]) Ergebnis der Aufteilung in regelmäßige Viereckснetze mit gleicher Maschenzahl bei einer getrimmten sowie einer neu modellierten NURBS-Fläche.

Unterschiedliche uv-Parameter, identische kartesische Koordinaten. Eine weitere Eigenheit der *NURBS*-Flächen ist, dass es auf den Flächen Stellen geben kann, an denen unterschiedliche uv-Parameter den selben Punkt im kartesischen Raum beschreiben. Dies ist bei bestimmten Flächentypen der Fall, etwa im Scheitelpunkt eines Ellipsoids, Paraboloids oder Kugelabschnitts [Abb. 13], wo alle Paare aus minimalem u-Wert und allen möglichen v-Werten immer auf der selben Stelle liegen. Dieses Phänomen tritt aber auch dann auf, wenn beispielsweise ein Punkt als Querschnitt einer *Loft*-Operation verwendet wird. Zudem gibt es bei manchen Flächen einen Bereich, in dem sich zwei Flächenkanten treffen, was ebenfalls bei den genannten Flächentypen sowie grundsätzlich bei allen geschlossenen, rotationssymmetrischen Flächen auftritt. Auch auf diesen Kanten können zwei oder mehr Knoten im gleichen Punkt zu liegen kommen. Solche Fälle können zwar mit Einschränkungen optimiert werden, sind aber durch die Programmmodule zur konstruktiven Umsetzung nicht bearbeitbar. Auch hier hilft nur näherungsweises Nachbilden der entsprechenden Ursprungsfläche.

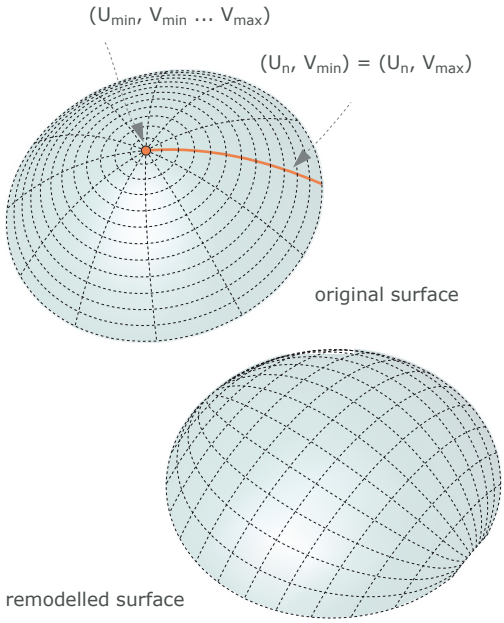


Abb. [13]) Kugelabschnitt, bei dem unterschiedliche uv-Werte im Pol und entlang einer Isoline zusammenfallen sowie näherungsweise nachgebildeter Kugelabschnitt.

Einschränkungen durch die Flächenaufteilung

Deckungsgleiche Flächenkanten Abb. 14. Während Flächen, bei denen mehrere uv-Werte in einem Punkt zusammenfallen, zum Abbruch des Programmablaufs führen, sind geschlossene Flächenbänder, bei denen zwei Flächenkanten deckungsgleich sind, für den genetischen Algorithmus kein Problem – sie können zunächst ohne Schwierigkeiten elementiert und optimiert werden. Allerdings müsste hier eine Verbindung der Knoten über die aufeinanderliegenden Flächenkanten hinweg hergestellt werden, also etwa die letzte Knotenreihe entlang der Kante gelöscht werden und die Knoten der davorliegenden Reihe mit der ersten Reihe entlang der Kante zu Maschen verbunden werden. Um diese Verbindung zu realisieren, müsste eine weitere Aufteilungsvariante implementiert werden. Da die Algorithmen in der jetzigen Version diese Konnektivität nicht herstellen können, scheiden auch solche Fläche für eine Bearbeitung aus.

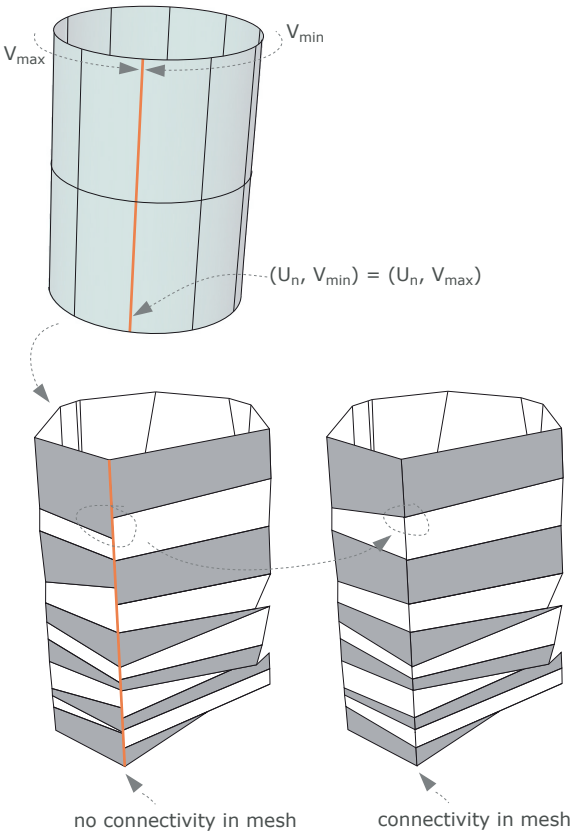


Abb. 14) Problem der Konnektivität im Flächennetz bei geschlossenen Flächenbändern.

Unterschiedliche Isolinienlängen. Weiterhin sollten Flächen, die bearbeitet werden, nicht zu große Längenunterschiede entlang der Isolinien aufweisen. Dies ist keine grundsätzliche Einschränkung, kann aber bei bestimmten Kombinationen von Fitnesstests dazu führen, dass keine zufriedenstellende Lösung gefunden werden kann.

Einfluss der erzeugenden Elemente auf das Flächennetz Abb. 15. *NURBS*-Flächen haben einen gewissen ‚Sinn' für Historie – die Verteilung der uv-Parameter auf einer Fläche ist abhängig von den Parametern und den Positionen der erzeugenden Kurven: Flächen mit exakt identischer Form können deshalb auch schon bei einer regelmäßigen Aufteilung unterschiedliche Netze hervorbringen. Dies kann zum Problem werden, etwa wenn unerwartet Überschneidungen einzelner Flächenmaschen auftauchen, kann aber auch schon bei der Modellierung der Ursprungsfläche strategisch genutzt werden, um ein bestimmtes Aufteilungsbild zu erzeugen.

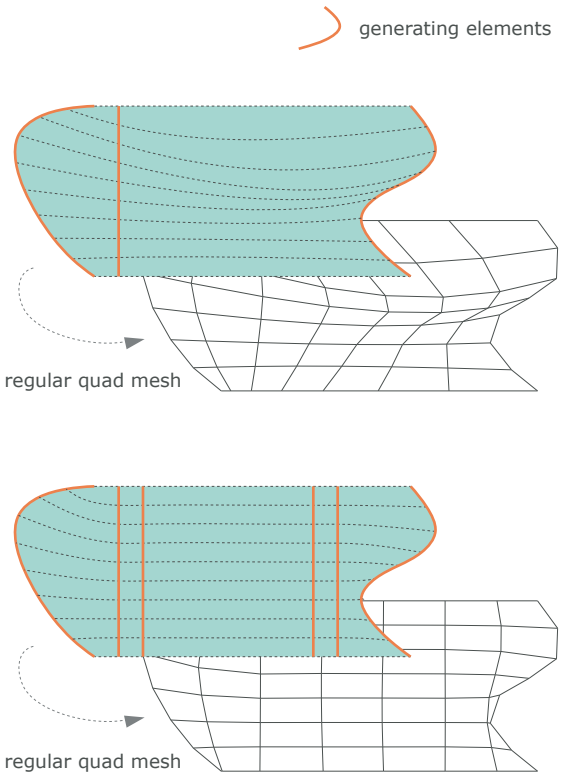


Abb. 15) Unterschiedliche Netze entstanden aus Flächen von der gleichen Form.

Damit sind die wichtigsten Einschränkungen genannt, die bei der Modellierung einer Ursprungsfläche zu beachten sind. Weitere Limitierungen bezüglich der Form einer Fläche gibt es kaum: *„Mit Freiformflächen kann man natürlich auch mathematisch definierte Flächen optisch gut annähern." (Gläser, 2005)* [03] Entsprechend können nicht nur freie Formen, sondern auch mathematische Formen, die nur einen speziellen Fall in den Parametern der *NURBS*-Beschreibung darstellen, in der Prozesskette bearbeitet werden.

8.4 Evolutionäre Flächenstrukturoptimierung

8.4.1 Genotyp, Phänotyp, Modell

In einem genetischen Algorithmus besteht ein Individuum aus zwei Komponenten. Die eine Komponente ist der sogenannte Phänotyp: Er umfasst alle variablen Parameter, die ein Modell beschreiben, die also durch die genetischen Operatoren verändert werden sollen. Dies kann beispielsweise die Höhe eines Stuhls, das Fassungsvermögen eines Glases oder die Kraft sein, die ein Hebelarm ausübt. Bei den genetischen Algorithmen arbeiten die Operatoren auf der Ebene des Genotyps, der zweiten Komponente eines Individuums. Er enthält eine binär codierte Form der veränderbaren Modell-Parameter, also eine Zeichenkette aus Nullen und Einsen. Die Bewertung (Fitnesstest) wird auf Ebene des Phänotyps durchgeführt. Bei der Anwendung eines genetischen Algorithmus findet demnach ein ständiger Transformationsschritt von Genotyp zu Phänotyp statt. Das was gegebenfalls als (digitales) Modell visualisiert und der Wahrnehmung eines Design-Modells entspricht, ist ein weiterer Transformationsschritt: Hier werden die variablen Parameter eines Modells mit den konstanten Größen zusammengebracht und dargestellt.

Der Genotyp eines Flächennetzes besteht aus einer binären Zeichenkette, die in mehreren Umwandlungsschritten in den Phänotyp überführt wird. Zunächst werden die binären Werte in Dezimalzahlen umgewandelt und auf die uv-Parameter einer bestimmten Fläche sowie den Bewegungsspielraum der Knoten entlang ihrer jeweiligen Normalen zur Fläche bezogen. Daraus werden die Positionen der Knoten in kartesischen Koordinaten abgeleitet. Der Phänotyp ist ein Array mit allen Knotenpositionen.

Diese Trennung in Genotyp und Phänotyp ist im Prinzip nicht notwendig und kommt auch nicht bei allen Typen von evolutionären Algorithmen vor. Hier hat sie sich aber bei einigen Vortests als zwingend erforderlich erwiesen, weil dadurch die Speicherbelastung der Rechner deutlich vermindert und häufige Abstürze des Betriebssystems vermieden werden konnten. Zudem hat die binäre Codierung den Vorteil, dass die genetischen Operatoren sehr einfach anzuwenden sind und kein aufwändiges *Constraint Handling* betrieben werden muss. Der Aufbau eines Genotyps ist folgend an der zweiten Aufteilungsvariante dargestellt.

Struktur des Genotyps

Alle im Genotyp gespeicherten Werte sind Teiler für die uv-Parameterwerte einer Fläche und die definierte Abweichung entlang der Normalen. In den Abbildungen ist die Verwendung eines 8-Bit Rasters gezeigt. Es können, innerhalb gewisser Grenzen, die durch die Zahl der Maschen vorgegeben wird, aber auch andere Raster verwendet werden. Die Änderung des Bit-Rasters entspricht einer Veränderung der Auflösung des Suchraums: Je größer das Raster, desto feiner die minimal möglichen Änderungen bei den Knotenpositionen.

An erster Stelle der Zeichenkette steht die Unterteilung der Ursprungsfläche entlang der Isolinien der u-Richtung [Abb. 16 / oben]. Die Position jeder Unterteilung wird durch einen binären Wert repräsentiert. Sind die Randknoten des Netzes an die Fläche gebunden, ist der erste Wert immer 0, der letzte Wert immer der höchste, der im gewählten Bit-Raster ausgedrückt werden kann. Alle anderen Werte liegen dazwischen. Das Gleiche gilt für die Unterteilung der Fläche entlang der Isolinien in v-Richtung, deren Werte sich in der Zeichenkette anschließen [Abb. 16 / unten].

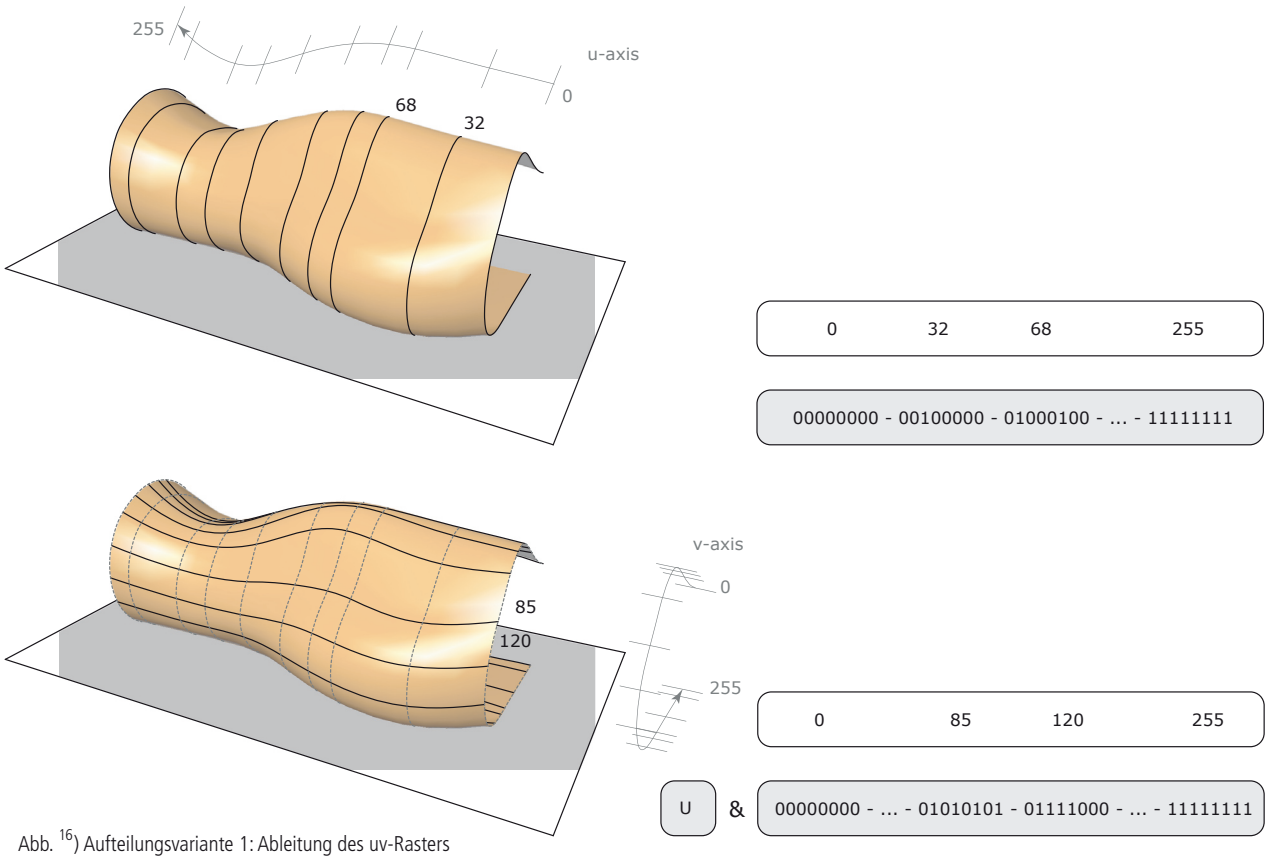


Abb. [16]) Aufteilungsvariante 1: Ableitung des uv-Rasters aus dem Genotyp.

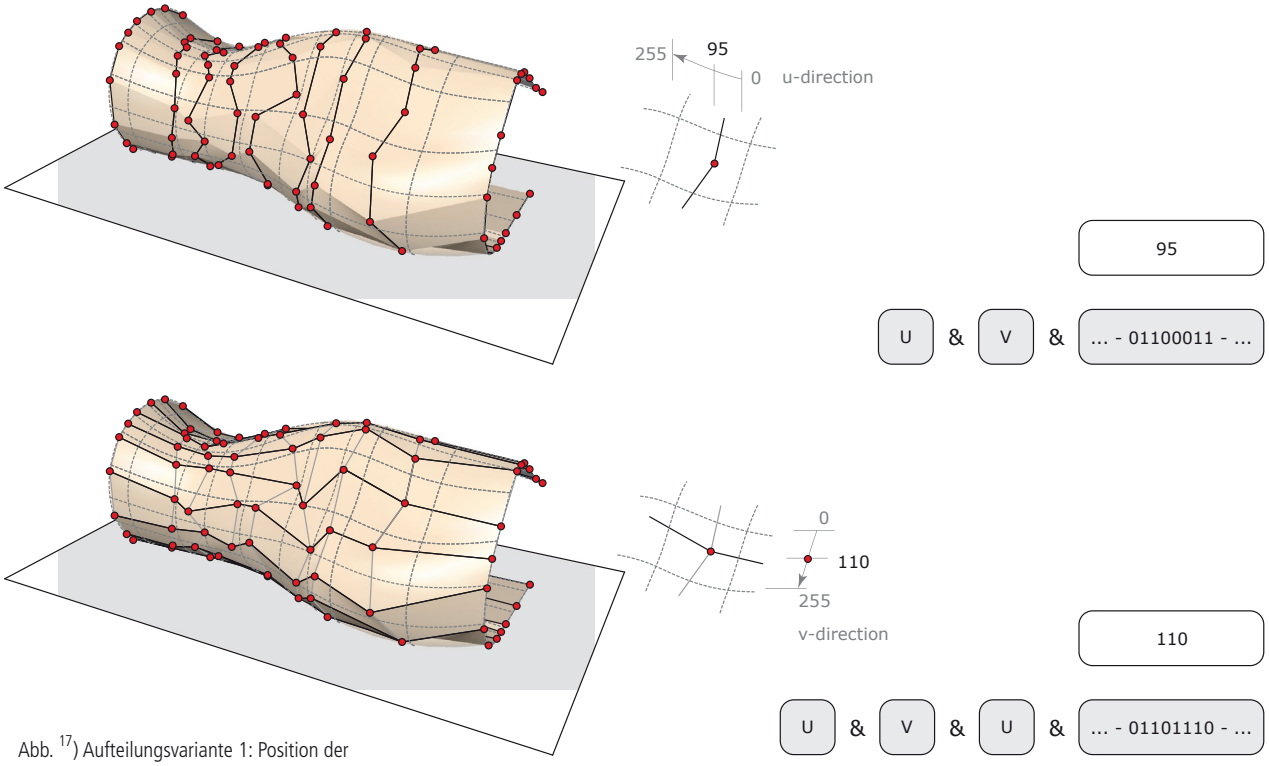


Abb. [17]) Aufteilungsvariante 1: Position der Knoten innerhalb der Rasterzellen.

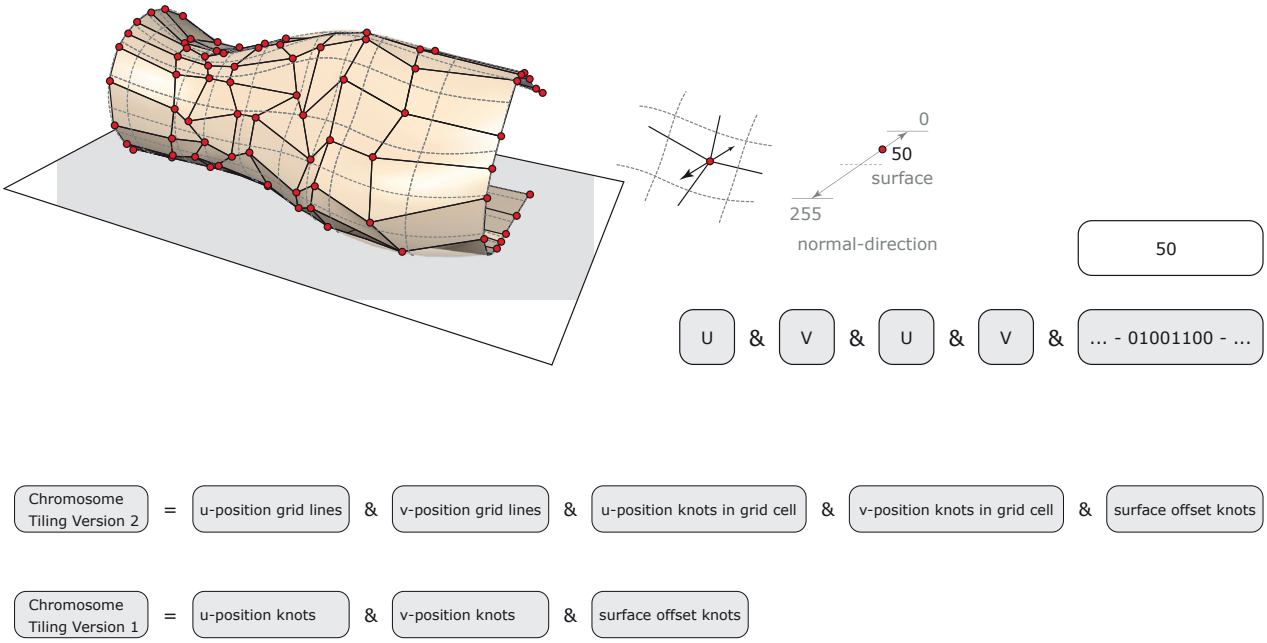


Abb. [18]) Abweichung der Knoten von der Fläche sowie die Struktur des Genotyps beider Aufteilungsvarianten.

Die Werte für die einzelnen Knotenpunkte repräsentieren ihre relative Position zu den Rändern ihrer Zelle. 0 bedeutet also, sie liegen auf einem, der höchste Wert im binären Raster, sie liegen auf der anderen Begrenzung der Zelle. Die dazwischen liegenden Werte werden wieder entsprechend umgerechnet. In der Zeichenkette sind zuerst alle Positionen in u-Richtung und dann alle in v-Richtung enthalten [Abb. 17]. Schließlich folgt noch die Abweichung der Knotenpunkte von der Fläche. Die binären Werte repräsentieren hier einen Teiler für die maximale Abweichung [Abb. 18]. Dieser Teil des Chromosoms wird auch dann von den genetischen Operatoren verändert, wenn der Wert für die Abweichung gleich 0 ist, die Änderungen haben lediglich keine Auswirkungen, die Gene werden zu *Junk*.

Der gesamte Genotyp für die 2. Aufteilungsvariante setzt sich zusammen aus: Position der Rasterlinien in u-Richtung, Position der Rasterlinien in v-Richtung, relative Positionen der Knoten in u-Richtung, relative Positionen der Knoten in v-Richtung, Abweichungen der Knoten in Normalen-Richtung. Entsprechend besteht der kürzere Genotyp für die 1. Elementierungsvariante aus den Postionen der Knoten in u-Richtung, Positionen der Knoten in v-Richtung und den Normalen-Abweichungen [Abb. 18].

8.4.2 Fitnesstests

Die hier dargestellte Version des genetischen Algorithmus enthält 12 verschiedene Fitnesstests, die in unterschiedlichen Kombination zur Bewertung der Güte von Flächennetzen benutzt werden können. Sie dienen dem *Constraint Handling* der Flächennetze, der Kontrolle geometrischer Parameter und der Strukturentwicklung unter statischen Gesichtspunkten. Die einzelnen Tests sind größtenteils voneinander unabhängig. Sie können, je nach Anwendungsfall, aktiviert oder deaktiviert sein. Die beiden Ausnahmen hiervon sind Test 12, der nur in Kombination mit einem Test zur statischen Evaluierung verwendet werden kann, sowie Test 1, der für das *Constraint Handling* zwingend erforderlich ist und in jedem Fall durchgeführt wird.

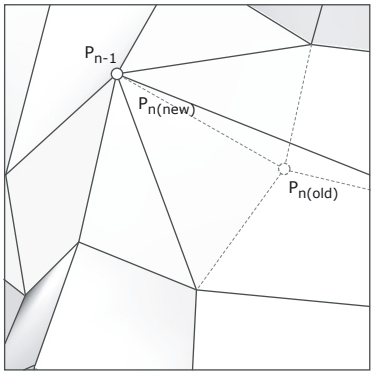
Constraint Handling

Test 1 – Doppelte Knoten im Netz [Abb. 19]

Die gewählte Systematik, wie ein Netz aus einer Fläche abgeleitet und im Genotyp repräsentiert wird, kann dazu führen, dass durch *Crossover* oder *Mutation* Netze erzeugt werden, bei denen zwei oder mehr Knoten auf dem selben Punkt im kartesischen Raum liegen: Es entsteht eine Mischung aus drei- und viereckigen Flächenmaschen, die weder durch die Fitnesstests des genetischen Algorithmus bewertet noch durch die später beschriebenen Konstruktionsmodule interpretiert werden können. Um solche nicht gewünschten Eigenschaften zu unterdrücken, gibt es bei genetischen Algorithmen im Allgemeinen drei Strategien:

a) Durch entsprechendes Design des Generators wird verhindert, dass Modelle, die den Randbedingungen nicht entsprechen, überhaupt erst entstehen können.
b) Solche Modelle werden unmittelbar nach ihrer Entstehung wieder gelöscht.
c) Es wird ein Fitnesstest konzipiert, der solchen Modellen einen sehr niedrigen globalen Fitnesswert zuweist. Die Folge davon ist, dass sich die unerwünschten Eigenschaften in einer Population nicht weiter ausbreiten können.

Welche Strategie die Richtige ist, muss fallabhängig entschieden werden. Für das konkrete Problem weist dieser erste Test Flächennetzen, bei denen zwei oder mehr Knoten auf dem selben Punkt liegen, den global niedrigsten möglichen Fitnesswert (0%) zu. Weitere Tests werden in diesem Fall nicht mehr durchgeführt, die Flächennetze aber nicht gelöscht: Ihr Genotyp kann trotzdem Informationen enthalten, die durch *Crossover* oder *Mutation* zu einer Verbesserung in einem neuen Individuum beitragen. Besteht eine Population ausschließlich aus Flächennetzen, bei denen zwei oder mehr Knoten auf einer Stelle liegen, wird die Ausführung des genetischen Algorithmus beendet – die Ursprungsfläche besitzt Eigenschaften, die eine Bearbeitung in der Prozesskette unmöglich machen.



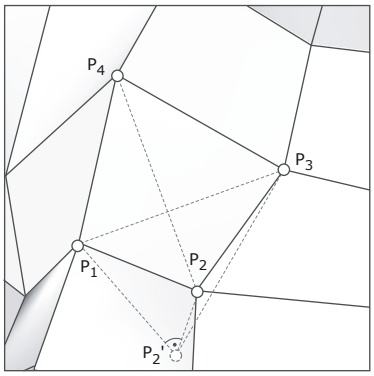
surface mesh valid if:
for each (n) knot (P) in surface mesh:
$P_{(n)}$ (x,y,z) < > $P_{(n-1)}$ (x,y,z);
$P_{(n)}$ (x,y,z) < > $P_{(n-2)}$ (x,y,z); ...
$P_{(n)}$ (x,y,z) < > $P_{(0)}$ (x,y,z)

Abb. [19]) Test 1: Doppelte Knoten im Flächennetz.

Evaluierung geometrischer Parameter

Test 2 – Durchbiegung der einzelnen Flächenmaschen [Abb. 20]

Hier wird jede einzelne Masche des Netzes darauf geprüft, ob sie durch eine planare Scheibe aus doppeltem Isolierverbundglas belegt werden könnte. Es wird ein näherungsweises Berechnungsverfahren genutzt, das auch in der Praxis der Tragwerkskonstruktion Anwendung findet.



for all faces in surface mesh:
$d_1 = \overline{P_2P_4}$; $d_2 = \overline{P_1P_3}$
if d1 > d2 then
    test passed if: d_1 / 400 >= $\overline{P_2P_2'}$
else
    test passed if: d_2 / 400 >= $\overline{P_3P_3'}$

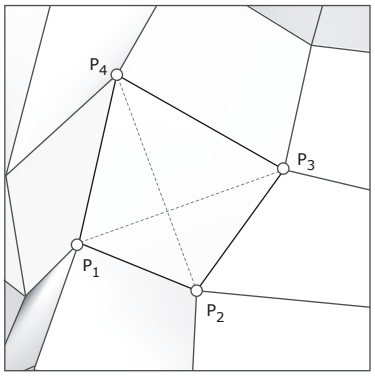
Abb. [20]) Test 2: Durchbiegung Flächenmaschen.

Zunächst wird die längeren Diagonale einer Vierecksmasche ermittelt und deren Länge durch 400 geteilt. Dann wird der maximale Abstand der längeren Diagonale zu der Ebene, die die übrigen drei Eckpunkte des Vierecks beschreiben, errechnet. Ist die Distanz kleiner als der zuvor berechnete Quotient, ist die Durchbiegung im erlaubten Bereich. Wegen unterschiedlicher Materialkennwerte kann diese vereinfachte Berechnung nur für Isolierverbundglas verwendet werden.

Die folgenden sieben Tests (3 bis 9) dienen im Wesentlichen dazu, die Ausformung der einzelnen Maschen im Netz zu beeinflussen. Geprüft werden ähnliche geometrische Parameter, wie sie bei der *Mesh*-Bildung von *CAD*-Systemen verwendet werden. Dies hat einerseits eine ästhetische Komponente, z.B. die Annäherung von Maschenkanten an vorher definierte Proportionsverhältnisse oder die Entwicklung eines Flächennetzes hin zu eher homogen wirkenden Maschen. Andererseits sind diese Tests auch für eine spätere konstruktive Umsetzung von Bedeutung: Treten etwa sehr spitze Winkel in einer Viereckmasche auf, kann die Fertigung eines zugehörigen Knotens problematisch sein, werden gewisse Mindestlängen unterschritten, kann ein Stab möglicherweise nicht genügend Raum zwischen zwei Knoten finden, etc.

Test 3 – Verhältnisse der Diagonalen Abb. 21

Hier wird das Längenverhältnis der beiden Diagonalen einer Viereckmasche geprüft: Die maximal erlaubte Längendifferenz zweier Diagonalen wird als Prozentsatz ihrer Gesamtlänge ausgedrückt.

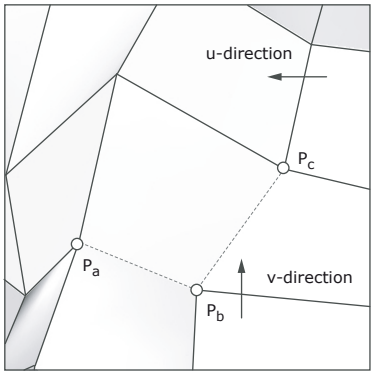


for all faces in surface mesh:
$d_1 = \overline{P_1P_3}$; $d_2 = \overline{P_2P_4}$
test passed if:
Abs (d_1 - d_2) <= (d_1 + d_2) / 100 * p
p > 0; userdefined

Abb. 21) Test 3: Diagonalenverhältnis.

Test 4 und 5 – Minimallängen der Kanten Abb. 22

Mit diesen beiden Tests wird ermittelt, ob die Kanten der Maschen bestimmte Längen unterschreiten. Einmal werden alle Viereckskanten geprüft, die entlang der u-Richtung einer Fläche orientiert sind, der andere Test führt die Prüfung für alle Kanten der v-Richtung aus. Für beide Richtungen können unterschiedliche Längen definiert werden.



for all vertices in u- direction:
$d = \overline{P_aP_b}$
test passed if: d >= d_{min}
d_{min} > 0; userdefined

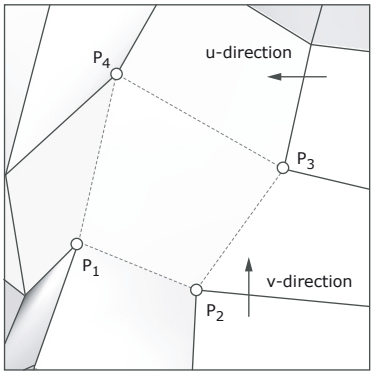
for all vertices in v- direction:
$d = \overline{P_bP_c}$
test passed if: d >= d_{min}
d_{min} > 0; userdefined

Abb. 22) Test 4 und 5: Minimallängen.

Test 6 und 7 – Seitenverhältnisse der Kanten Abb. 23

Test 6 und 7 prüfen die Längenverhältnisse der jeweils gegenüberliegenden Kanten einer Viereckmasche. Test 6 errechnet die Verhältnisse für die Kanten, die entlang der u-Richtung einer Fläche ausgerichtet sind. Test 7 berechnet die Kanten entlang der v-Richtung. Entsprechend können auch hier für die beiden unterschiedlichen Richtungen verschie-

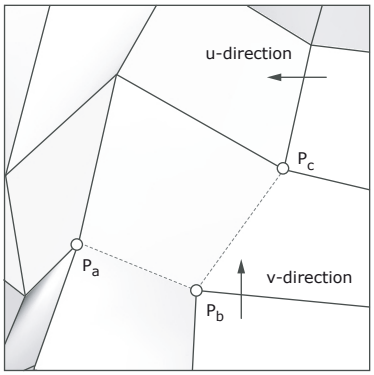
dene Grenzwerte definiert werden. Diese Werte bezeichnen, analog zu Test 3, die maximale Abweichung in Prozent der Gesamtlänge des jeweils geprüften Kantenpaares.



for all faces in surface mesh (u-direction):
$d_1 = \overline{P_1P_2}$; $d_2 = \overline{P_4P_3}$
test passed if:
Abs (d_1 - d_2) <= (d_1 + d_2) / 100 * p
p > 0; userdefined

for all faces in surface mesh (v-direction):
$d_1 = \overline{P_1P_4}$; $d_2 = \overline{P_2P_3}$
test passed if:
Abs (d_1 - d_2) <= (d_1 + d_2) / 100 * p
p > 0; userdefined

Abb. [23]) Test 6 und 7: Kantenverhältnisse.



for all vertices (n) in u- direction:
$d = d + \overline{P_aP_b}$
test passed if: Abs (d / n - $\overline{P_aP_b}$) <= d_{min}
d_{min} >= 0; userdefined

for all vertices (n) in v- direction:
$d = d + \overline{P_aP_c}$
test passed if: Abs (d / n - $\overline{P_aP_c}$) <= d_{min}
d_{min} >= 0; userdefined

Abb. [24]) Test 8 und 9: Gleiche Kantenlängen.

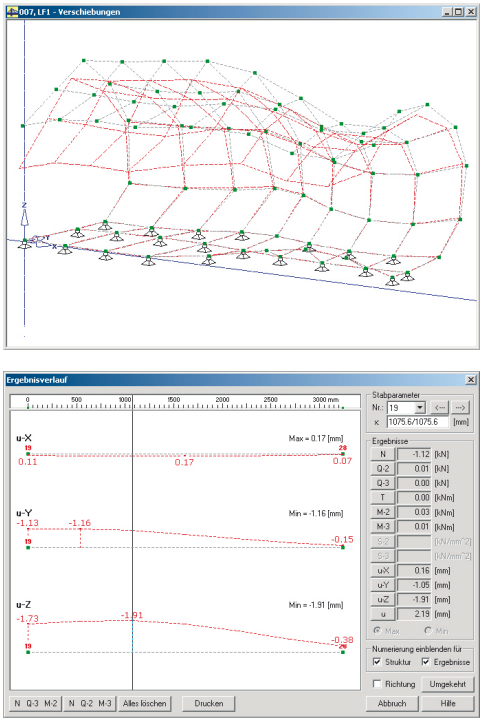
Test 8 und 9 – Gleiche Kantenlängen Abb. 24

Hier wird evaluiert, inwieweit die Längen der einzelnen Stäbe voneinander abweichen, diese Prüfung wird wieder getrennt für die beiden Flächenrichtungen durchgeführt. Die hierfür definierten zwei Werte entsprechen der maximal erwünschte Abweichung eines Stabs von der Durschnittslänge aller Stäbe der jeweiligen Richtung.

Strukturoptimierung unter statischen Gesichtspunkten

Test 10 und 11 – Knoten- und Stabverformung Abb. 25

Über diese beiden Tests kann näherungsweise eine statische Optimierung der Struktur der Flächennetze erreicht werden. Zur Berechnung werden die Netze nach *RSTAB®* exportiert, eine *Software* zur Evaluierung von Stabtragwerken nach der Finite Elemente Methode. Als Grundlage für die Berechnung kann jedes Material und jeder Querschnittstyp verwendet werden, der in den Bibliotheken von *RSTAB®* enthalten ist. Der Anwendung dieser beiden Tests liegt folgende Überlegung zu Grunde: Ein wesentlicher Aspekt beim Tragwerksentwurf ist die Dimensionierung der Bauteile, die so gewählt werden muss, dass das Tragwerk den Baunormen entspricht, aber aus ökonomischen und eventuell auch ästhetischen Erwägungen heraus auch nicht zu üppig ausfallen sollte. Insofern macht es keinen Sinn, für eine Strukturoptimierung die Größe der Querschnitte bereits so zu definieren, dass sie den statischen Anforderungen (vermutlich) in jedem Fall genügen werden. Entsprechend wird für diese beiden Tests ein Querschnitt gewählt, der zwar noch realistisch, aber gleichzeitig deutlich zu gering dimensioniert erscheint. Für diese Strategie ist eine Berechnung unter Eigenlast ausreichend: Die tatsächliche Bemessung der Querschnitte unter konkreten Bedingungen muss dann nachträglich, am evolutionär optimierten Flächennetz erfolgen. Aus den Ergebnissen, die die Berechnung in *RSTAB®* liefert, werden die maximale Knotenverschiebung (Test 10) und die maximale Stabverschiebung (Test 11) extrahiert.



knots

beams

knot / beam deviation

knot deviation for one mesh:
test passed if: deviation <= $deviation_{min}$
$deviation_{min}$ >= 0; userdefined

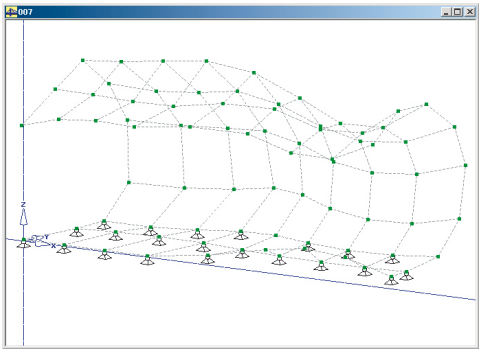
beam deviation for one mesh:
test passed if: deviation <= $deviation_{min}$
$deviation_{min}$ >= 0; userdefined

Abb. [25]) Test 10 und 11: Darstellung der Stab- und Knotenverschiebungen (oben) und des detaillierten Ergebnisses für einen Stab in RSTAB®.

Die größte Knotenverschiebung gibt Auskunft über die maximale lokale Verformung eines Stabtragwerks. Die Aussagekraft dieses Wertes ist abhängig von Größe und Art des Tragwerks. Bei Balkentragwerken sollte die maximale Knotenverschiebung z.B. nicht größer sein als die Spannweite geteilt durch 300, bei Auskragungen nicht größer als die Länge der Auskragung durch 150. Die maximale Stabverschiebung kann gleich der maximalen Knotenverschiebung sein. Ist die Stabverschiebung größer, kann sie als größte auftretende Durchbiegung bzw. Knickung bei den Stäben interpretiert werden. Die Aussagekraft dieses Wertes ist von der Länge des entsprechenden Stabes abhängig. Für beide Tests kann ein Zielwert für die jeweils erwünschte maximale Verschiebung definiert werden. Diese Werte sollten entweder 0 sein, zumindest aber im Verhältnis zur Größe der Struktur sehr klein gehalten werden, damit über diese Kriterien auch während des gesamten evolutionären Prozesses selektiver Druck auf die Flächennetze ausgeübt wird.

Test 12 – Auflager Abb. 26

Für die Berechnung der Flächennetze in *RSTAB*® ist es nötig, Auflagerpunkte zu bestimmen. Dazu wird bei jedem Flächennetz der Knoten mit dem kleinesten z-Wert gesucht und ein eingangs definierter Wert aufaddiert. Alle Knoten, die im so festgelegten Bereich liegen, werden als feste Auflager betrachtet. Im Fortlauf eines Optimierungsprozesses, bei dem einer oder beide Tests zur statischen Berechnung aktiviert sind, zeigen alle Knoten der Netze eine deutliche Tendenz auf der Fläche in diesen Bereich zu wandern und zu Auflagern zu werden – je mehr Auflager um so weniger Verschiebung. Evaluationskriterium dieses letzten Test ist eine festgelegte Anzahl von Auflagerpunkten. Durch seine Anwendung (unter Umständen zusammen mit einem oder mehreren Tests für geometrische Parameter) kann die Verteilung von Auflagern ungefähr kontrolliert werden.



supports

supports for one mesh:
test passed if: n supports <= n $supports_{min}$
n $supports_{min}$ >= 1; userdefined

Abb. [26]) Test 12: Anzahl der Auflagerpunkte.

Berechnung der Fitness eines Flächennetzes

Die globale Fitness eines Individuums ist das Maß für seinen Fortpflanzungserfolg. Sie entscheidet nach dem vom *Dawkins (1976)* [04] in *„The Selfish Gene"* beschriebenen Prinzip darüber, welche Gene von Generation zu Generation weitergegeben werden und damit auch mit über die Ausbildung der spezifischen Eigenschaften eines Individuums. Dies gilt ebenso bei der technischen Nachahmung des Evolutionsprinzips durch einen genetischen Algorithmus. Hier wird in der Regel mit konstantem selektivem Druck gearbeitet (die Evolution kann keinen Umweg über ‚schlechtere' Ereignisse machen). Dies führt dazu, dass sich bestimmte Kombinationen von Genen (Parametern) eines Modells durchsetzen (falls nicht nur mit gleichbleibenden Parameterlisten gearbeitet wird) und dass die Werte der Parameter (Allele) allmählich immer geringere Schwankungen aufweisen: Es entsteht der gewollte Effekt, dass die Lösungen allmählich zu einer Lösung hin konvergieren.

Bei einem multimodalen Optimierungsproblem, was bedeutet, dass nach mehreren Testkriterien evaluiert wird und *„dass sich die Optima nicht nach einer Regel im Variablenraum anordnen" (Rechenberg, 1994)* [05], wird eine evolutionäre Optimierung normalerweise immer auf einem lokalen Optimum ‚hängenbleiben' – eine Art Kompromiss, bei der die gefundenen Lösungen so gut wie möglich den unterschiedlichen Kriterien entsprechen. Der Anwendungsfall ‚Flächennetz' macht es dem genetischen Algorithmus besonders schwer, innerhalb einer angemessenen Zeit eine gute Lösung zu entwickeln: Die Knoten eines solchen Netzes können nicht diskret behandelt werden. Die Verschiebung eines innenliegenden Knotens kann zum Beispiel bei einem oder mehreren Tests lokal eine Verbesserung bei einer Viereckmasche bedeuten, gleichzeitig aber auch zu deutlichen Verschlechterungen bei den drei anliegenden Maschen führen.

Cross-Injection, der Austausch einzelner Individuen zwischen verschiedenen Populationen, ist eine Möglichkeit, die Leistungsfähigkeit eines genetischen Algorithmus hinsichtlich multimodaler Optimierungsprobleme zu steigern. Neben diesem evolutionstechnischen Kniff ist das Design sowie die Zusammenstellung geeigneter Fitnessfunktionen eine weitere Stellschraube, um die Leistungsfähigkeit eines genetischen Algorithmus zu erhöhen. Eine wichtige Rolle hierbei spielt auch, auf welche Weise die Ergebnisse einzelner Tests zusammengefasst werden, wie also die globale Fitness eines Individuums berechnet wird.

Fitnesswerte für einzelne Tests

Die verschiedenen Tests beziehen sich überwiegend auf lokale Eigenschaften von Flächennetzen. Test 3 z.B., der das Längenverhältnis der Diagonalen jeder einzelnen Masche prüft, besteht für sich schon aus einer ganzen Reihe von Fitnesstests: Pro Masche eines Netzes entsteht ein Testergebnis. Um ein gesamtes Testergebnis für die Diagonalenabweichung eines Flächennetzes zu erhalten, werden alle Abweichungen vom Zielwert, die für jede Masche gefunden wurden, zu einem Wert aufsummiert: Je höher die Summe, desto schlechter das Gesamtergebnis. Analog wird für die übrigen Tests verfahren, Ausnahmen hiervon sind Test 1, der gegebenfalls sofort eine globale Fitness von 0 zuweist, sowie die Tests 10, 11 und 12, bei denen das Ergebnis bereits durch einen einzelnen Wert repräsentiert ist.

$$resultTest_n = Test_n\,(mesh\ element_1) + Test_n\,(mesh\ element_2) \ldots + Test_n\,(mesh\ element_m)$$

Das schlechtestmögliche Ergebnis für einen Test ist für den hier besprochenen Anwendungsfall nicht absolut bestimmbar, es muss daher definiert werden: Die schlechteste Bewertung von 0 (0% Fitness) entspricht für jeden Test immer dem Doppeltem der Abweichungen beim jeweils schlechtesten Ergebnis, das in der zufälligen Startgeneration gefunden wurde. Die maximale Fitness 1 (100%) ist erreicht, wenn für einen Test das Ergebnis im gewünschten Wertebereich liegt, die Summe der Abweichungen also gleich 0 ist. Alle dazwischen liegenden Summenwerte werden in den entsprechenden Fitnesswert umgerechnet:

$$fitnessValueTest_n = 1 - (1 / badestResultTest_n * resultTest_n)$$

Globale Fitness

Die Berechnung der globalen Fitness (Zusammenfassen aller Einzeltests in einem Fitnesswert) für ein Flächennetz kann optional auf zwei Arten erfolgen. Bei der ersten Variante werden die einzelnen Fitnesswerte addiert und durch die Anzahl der durchgeführten Tests geteilt. Dabei kann der Wert jedes einzelnen Tests mit einem Faktor gewichtet werden. Entsprechend findet eine Verbesserung für höher gewichtete Tests einen stärkeren Niederschlag im globalen Fitnesswert. Der Optimierungsprozess lässt sich damit in einem gewissen Rahmen so beeinflussen, dass als wichtiger erachtete Eigenschaften oder Eigenschaften, die eine Tendenz dazu zeigen, sich in einer gewissen Kombination von Tests schlechter zu entwickeln, durch eine geeignete Gewichtung gefördert werden.

totalFitness =
($fitnessValueTest_0$ * $factor_0$ + $fitnessValueTest_1$ * $factor_1$ ... +
$fitnessValueTest_n$ * $factor_n$) / ($factor_0$ + $factor_1$ ... + $factor_n$)

Bei der zweiten Variante werden die einzelnen Fitnesswerte miteinander multipliziert *(vgl. Willems, 2006)* [06] anstatt addiert. Zum Vergleich: Wird bei der ersten Option für einen Test ein Fitnesswert von 0,9 und für einen zweiten Test ein Fitnesswert von 0,1 ermittelt, ergibt dies, bei gleicher Gewichtung, eine globale Fitness von 50%. Bei der zweiten Option ergibt sich dagegen eine globale Fitness von 0,9 * 0,1 = 0,09, also 9%. Entsprechend führt diese zweite Variante der Fitnessberechnung automatisch zu einer Bevorzugung von Verbesserungen, die gleichzeitig mehrere unterschiedliche Testkriterien erfüllen.

totalFitness =
$fitnessValueTest_0$ * $fitnessValueTest_1$ ... * $fitnessValueTest_n$

8.4.3 Beschreibung des genetischen Algorithmus

Funktionsweise

Jenseits der üblichen Termini der evolutionären Optimierung kann man sich die Funktionsweise des genetischen Algorithmus so vorstellen, dass hier zunächst einfach zufällig eine Reihe unterschiedlicher Flächennetze erzeugt und anschließend nach den gewünschten Kriterien bewertet wird. Dann werden – wieder zufällig – einzelne Parameter der Netze verändert, was immer dazu führt, dass sich die Position eines oder mehrer Knoten ändert. Dann wird wieder bewertet, erneut verändert usw. Der besondere Clou dabei ist, dass nach dem schon allgemein beschriebenen Prinzip *(vgl. Kapitel 5.1)* die besseren bewerteten Netze eine größere Chance haben die jeweils nächste Bewertungsrunde zu erreichen um dort wieder verändert zu werden. Dies führt zu einer Anhäufung kleiner Verbesserungen, wodurch die Qualität der Flächennetze im Fortlauf der Optimierung immer mehr steigt – zu Beginn recht zügig, je länger der evolutionäre Prozess dauert, desto kleiner werden die Verbesserungen pro Durchgang. Die besondere Schwierigkeit dabei ist, die Kriterien und die Optimierungsstrategie so zu

07) Genetische Operatoren sind Funktionen zum Ändern der binären Zeichenketten von Genotypen: Mutation (eine 1 in der Zeichenkette wird zur 0 oder umgekehrt), Crossover (Auseinanderschneiden und kreuzweises wieder Zusammenfügen zweier Zeichenketten), Multiple Crossover (mehrfaches Auseinanderscheiden und Zusammenfügen) und Inversion (Umkehren der Zeichenkette). (vgl. Bentley,1999)

08) Roulette Wheel Selection: Bei dieser Art der Selektion werden die Individuen entsprechend ihrer Fitnesswerte entlang einer Linie verteilt. Über eine zufällige Zahl wird eine bestimmte Position auf dieser Linie markiert. Das Individuum, zu dem das entsprechende Segmente gehört, wird ausgewählt, wie beim Roulette. Ein Nachteil ist, dass oft auch Individuen mit niedriger Fitness ausgewählt werden. Hat ein Individuum z.b. 100% Fitness und das folgende 99%, so nimmt das erste 1% des Rouletterads in Beschlag. Hat das schlechteste Individuum 40% Fitness, beansprucht es auch 40% des Rades und wird so mit weit höherer Wahrscheinlichkeit ausgewählt. Dieser Nachteil wird hier kompensiert, indem die Zufallszahl direkt aus der Summe aller Fitnesswerte einer Population genommen wird, so dass mit großer Häufigkeit nur gut bewertete Flächennetze ausgewählt werden.

wählen, dass es überhaupt möglich ist, innerhalb eines vernünftigen Zeitrahmens zu einer befriedigenden Lösung zu kommen.

Struktur und Anwendung

Der hier entwickelte evolutionäre Algorithmus lässt sich als genetischer Algorithmus charakterisieren, der als genetische Operatoren die Standards *Mutation* und einfaches *Crossover* [07] verwendet, bei dem die Individuen nach einer abgewandelte Form der *Roulette Wheel* - Methode [08] in Verbindung mit *Elitism* [09] selektiert werden und der einige Elemente von *Island-Injection* [10] enthält. Das Ablaufschema des Algorithmus ist auf *Seite 147* [Abb. 27] dargestellt, hier eine kurze ergänzende Beschreibung:

Voreinstellungen. Bevor der Optimierungsprozess beginnen kann, müssen die Rahmenparameter für den evolutionären Prozess gesetzt werden:

[n] Individuen pro Population	(n >= 2)
[n] Populationen pro Generation [1)]	(n >= 1)
[n] Generationen	(n >= 1)
[n] Binäres Raster [2)]	(n >= 4)
[n] Injection-Versuche [3)]	(0 <= n <= 100; % n Ind.)
[n] Beste Individuen	(0 <= n <= 100; % n Ind.)
[n] Zufällige Individuen	(0 <= n <= 100; % n Ind.)
[p] Crossover	(0 <= n <= 100; % n Ind.)
[p] Mutation	(0 <= p <= 100)
[n] Mutationsrate [3)]	(n >= 1)

n = Anzahl, p = Wahrscheinlichkeit

1) n = 1 bedeutet, Island Injection ist nicht möglich

2) Minimale Rastergröße entsprechend der Anzahl der Zellen.

3) Doppelte Ereignisse werden ebenfalls gezählt.

Dann werden die Parameter für die Flächenaufteilung (sowie die Visualisierung der Flächennetze) gesetzt:

[n] Maschen in u-Richtung	(n >= 2)
[n] Maschen in v-Richtung	(n >= 2)
[b] Aufteilungsart Knoten [4)]	(ja / nein)
[b] Aufteilungsart Knoten+Raster [4)]	(ja / nein)
[b] Regelmäßiges Raster [5)]	(ja / nein)
[b] Anpassung Testkriterien [6)]	(ja / nein)

4) Ausschließlich eine der beiden Optionen ist möglich.

5) Hier wird keine Optimierung durchgeführt, nur das Netz evaluiert.

6) Passt die Testkriterien an ein Flächennetz an, nur für regelmäßige Netze.

Initialisierung. Nachdem die Einstellungen festgelegt sind, wird der genetische Algorithmus aus *Rhino*® heraus gestartet. Es kann eine beliebige Anzahl Flächen zur Optimierung ausgewählt werden, die dann in einer Schleife nacheinander, mit den selben Einstellungen den Evolutionsprozess durchlaufen. Bei der Initialisierung werden alle Populationen der Startgeneration mit zufällig erzeugten Individuen gefüllt.

---

09) Elitism

bedeutet einfach, dass einige der besten Individuen direkt in die Folgegeneration übernommen werden, ohne dass genetische Operatoren angewandt werden. Dadurch wird vermieden, dass gute Lösungen durch die Operatoren zu schlechteren gemacht werden.

10) Island Injection:

Diese Methode ist eine Evolutionstechnik, bei der mehrere Populationen (Islands) parallel entwickelt werden. Nach bestimmten Regeln kann ein Austausch einzelner Individuen zwischen den parallel laufenden Populationen vorgenommen werden (Island Injection). Es besteht daher die Möglichkeit, dass sich die ‚guten Eigenschaften' des neuen Individuums in der Population mit den Eigenschaften der bereits vorhandenen so mischen, dass insgesamt ein höherer Hügel in der Fitnesslandschaft ereicht werden kann. (vgl. Lin et al., 1994)

Mapping. In diesem Schritt werden die Genotypen in die entsprechenden Parameter umgewandelt und so die Positionen der Knoten aller Flächennetze bestimmt und nach Wunsch visualisiert.

Evaluation. Hier werden alle Flächennetze durch die aktivierten Tests untersucht, bewertet und ihre globale Fitness berechnet, d.h. sie bekommen eine ‚Note' zwischen 0% Fitness (= schlechteste Bewertung) und 100% Fitness (= beste Bewertung) zugewiesen.

Haltebedingungen. Im Anschluss daran werden die Haltebedingungen geprüft. Die Ausführung des genetischen Algorithmus wird beendet, wenn

a) eine Population nur ungültige Individuen enthält,
b) die definierte Anzahl an Durchläufen erreicht ist ,
c) eine bestimmte Konvergenzrate erreicht ist, oder
d) ein Individuum einen Fitnesswert von 100% hat.

Ist keine dieser vier Bedingungen erfüllt, dann erfolgt:

Island Injection. Sofern mehr als eine Population vorhanden ist, kann ein Austausch einzelner Flächennetze zwischen den einzelnen Populationen stattfinden. Dazu werden zufällig zwei verschiedene Inseln und daraus je ein zufälliges Individuum ausgewählt und getauscht. Die geschieht solange, bis die definierte Anzahl an Austauschen vorgenommen ist. Die Auswahl der Individuen erfolgt nach dem abgewandelten *Roulette-Wheel* Verfahren, das weiter unten beschrieben ist.

Selektion und Erzeugung neuer Populationen. Die neuen Populationen für die nächste Generation werden nach folgenden Regeln gefüllt: Zunächst werden gut bewertete Individuen übernommen: Erst das Beste der Vorläuferpopulation, dann das Zweitbeste, das Drittbeste usw., solange bis die definierte Anzahl erreicht ist. Dann kann noch eine bestimmte Anzahl neuer, zufällig erzeugter Individuen hinzukommen. Danach kommen die genetischen Operatoren *Mutation* und *Crossover* zum Einsatz.

Crossover. Es wird nach einem abgewandelten *Roulette Wheel* - Prinzip selektiert. Dazu werden alle Fitnesswerte der vorherigen Population aufaddiert und daraus ein Zufallswert genommen. Das Flächennetz, das am nächsten an diesem Wert liegt, wird ausgewählt. Praktisch bedeutet dies, dass die Wahrscheinlichkeit, dass ein schlechtes Netz ausgewählt wird, sehr gering ist. Es herrscht also ein großer Selektionsdruck in der Population. Dann wird, wieder mit einer gewissen Wahrscheinlichkeit, entschieden, ob *Crossover* angewendet werden soll. Falls ja, wird über das *Roulette Wheel* ein zweites Flächennetz ausgewählt. Die Chromosomen der beiden werden jetzt an einem zufälligen Punkt auseinandergeschnitten und kreuzweise wieder zusammen gesetzt. Je nach dem, an welcher Stelle dieser zufällige Schnitt erfolgt, ändert sich das durch Kreuzung entstandene, neue Flächennetz mehr oder minder stark. Wird beispielsweise genau nach dem letzten Gen der Rasterlinien geschnitten, bleibt das Raster zwar gleich, aber (wahrscheinlich) ändern sich sehr viele Positionen der Knotenpunkte und ihre Abstände zur Fläche.

Mutation. Nach der Entscheidung über *Crossover* folgt die Entscheidung über *Mutation*, nach dem selben Prinzip wie eben. Soll das ausgewählte Individuum mutiert werden, werden einzelne Bits geflippt, d.h. aus einer 1 eine 0 gemacht oder umgekehrt. Dadurch ändert sich der durch das Gen codierte Parameter und entsprechend die Position eines Knotens. *Mutation* bedeutet also im Gegensatz zu einfachem *Crossover* immer eine eher kleine Änderung eines Flächennetzes. Wie oft ein Bit invertiert wird, ist durch den Wert bestimmt, der für die Mutationsrate gesetzt ist. Je höher dieser Wert, desto wahrscheinlicher ist es, dass größere Änderungen am Flächennetz vorgenommen werden.

Abschließend wird das ausgewählte Individuum in die neue Population gegeben, unabhängig davon, ob *Crossover* und / oder *Mutation* stattgefunden haben. Dann wird wieder der Schritt drei und vier angewandt (umwandeln – evaluieren) und danach wieder die Haltebedingungen geprüft. Dieser Prozess läuft solange, bis mindestens eine der Haltebedingungen erfüllt ist.

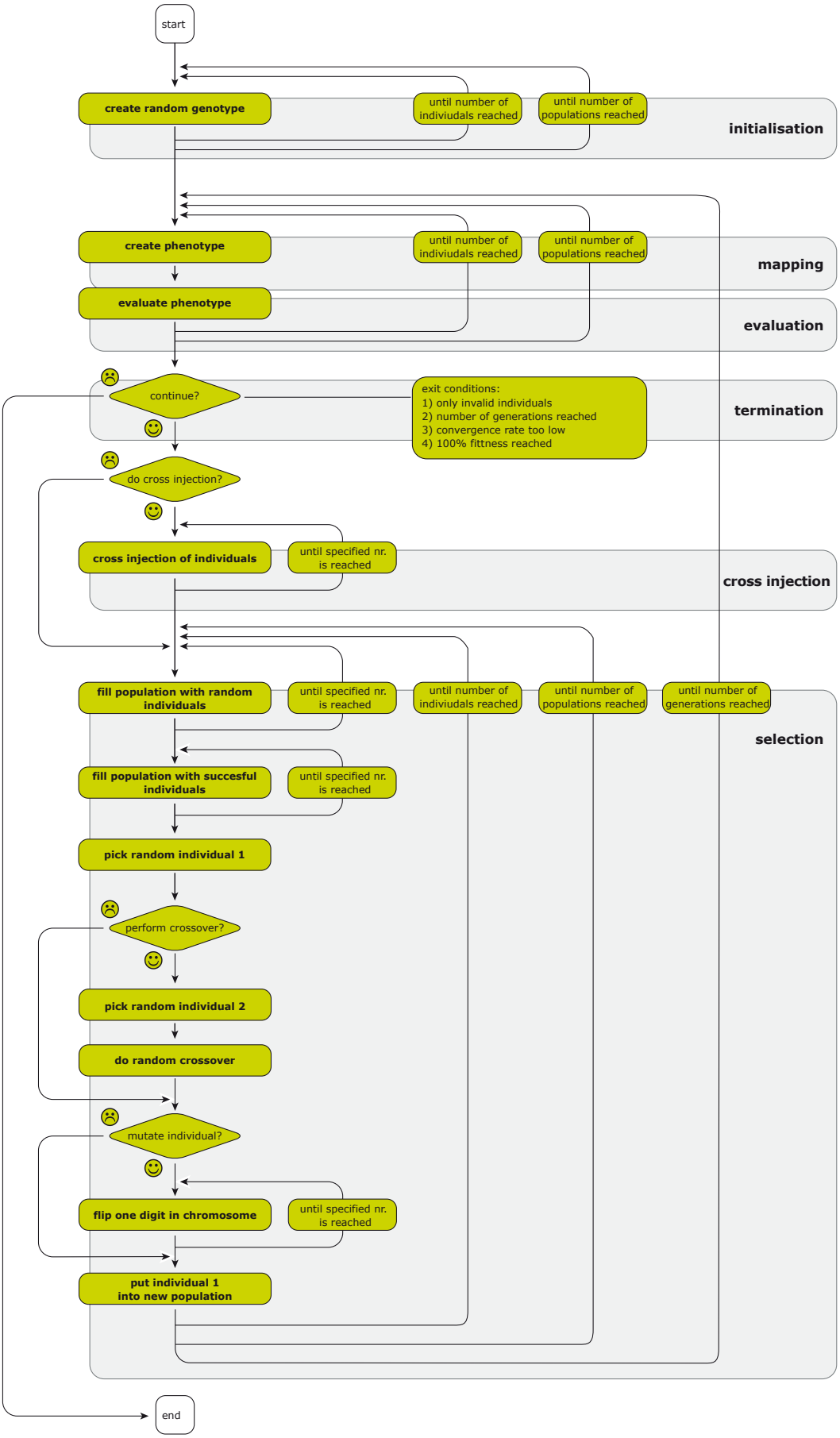


Abb. [27]) Ablaufschema genetischer Algorithmus.

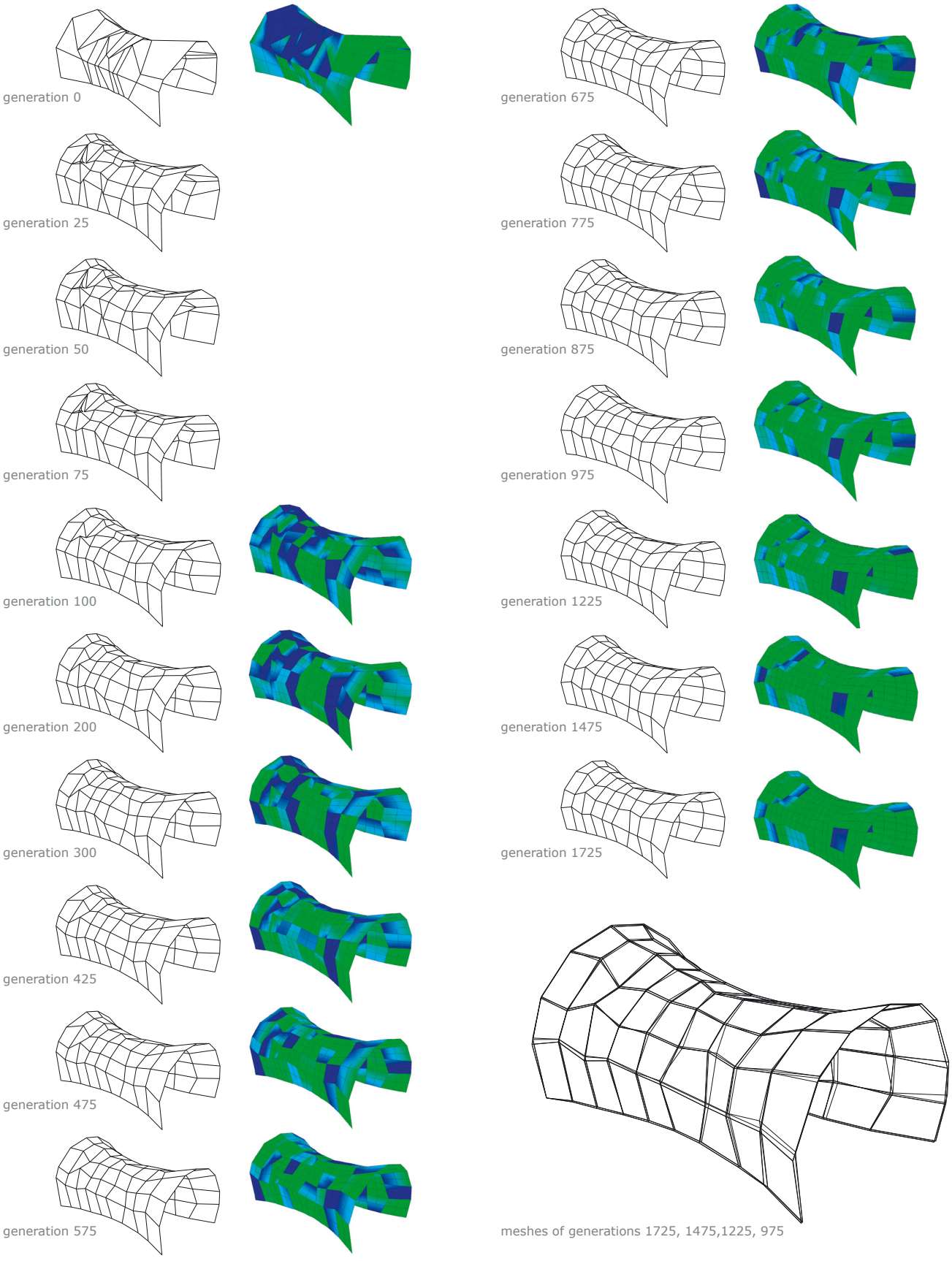


Abb. [28]) Visualisierte Zwischenschritte eines Optimierungsvorgangs –

Veränderung des Flächennetzes und zugehörige Gauss'sche Kurvaturanalyse.

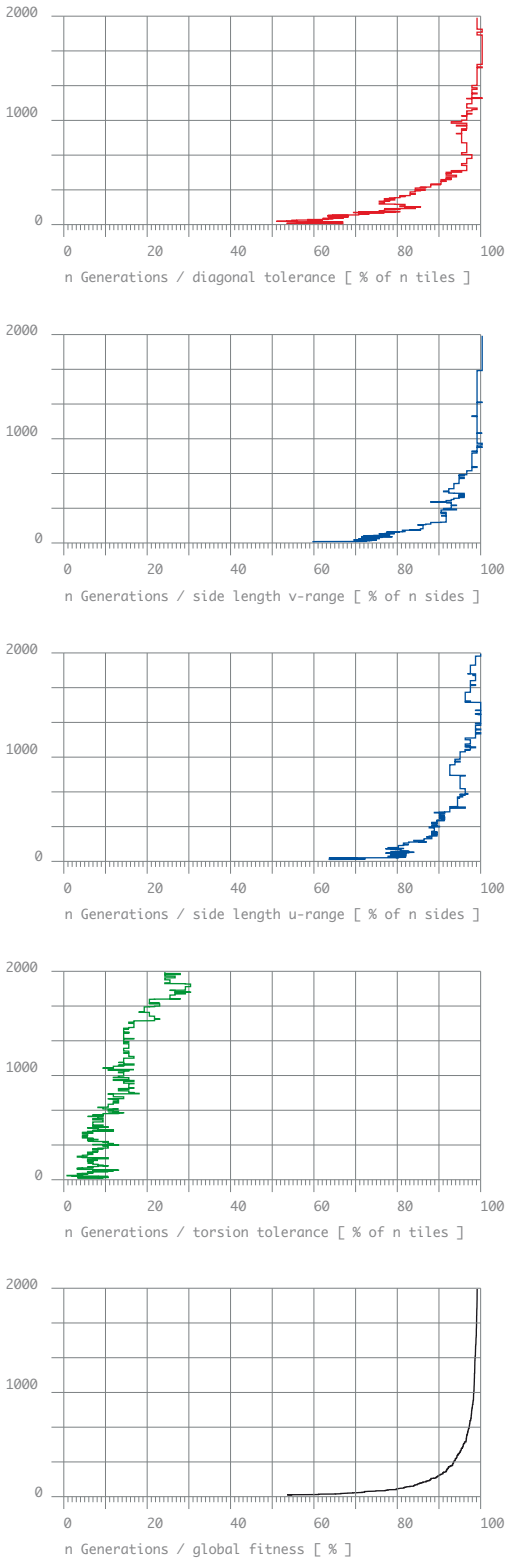


Abb. [29]) Evolutionsfortschritt für den in Abbildung 28 dargestellten Optimierungsvorgang. Farbig dargestellt ist der Verlauf der einzelnen Fitnesstest, schwarz zeigt die Entwicklung der globalen Fitness.

Die Abbildung auf der gegenüberliegenden Seite [Abb. 28] macht an einem Beispiel anschaulich, wie die evolutionäre Optimierung eines Flächennetzes vor sich geht. Optimierungskriterien für dieses Beispiel waren: Mindestlängen für die Maschenkanten in beide Flächenrichtungen, eine bestimmte maximale Diagonalenabweichung und die Durchbiegung der einzelnen Flächenkanten.

Die Linienzeichnungen zeigen die Veränderungen des Flächennetzes über die Zeit. Hier ist gut erkennbar, wie der technische Evolutionsvorgang sehr schnell aus einer zufälligen, chaotisch wirkenden Aufteilung heraus Ordnung schafft. Schon nach recht wenigen Generationen wirkt das Flächennetz – entsprechend der gewählten Werte für die Fitnesstests – relativ homogen. Die farbigen Darstellungen zeigen das Ergebnis der *Gauss'schen* Kurvaturanalyse für die einzelnen Flächenelemente – je grüner, desto weniger durchgebogen. Auch für diesen Test stellt sich also die gewünschte Entwicklung ein.

Anhand dieser Darstellungen kann man auch den typischen Verlauf des Evolutionsfortschritts nachvollziehen, wie er in den zugehörigen Graphen der nebenstehenden Abbildung [Abb. 29] zu sehen ist: Während zu Beginn die Änderungen noch sehr deutlich sichtbar sind, kann man gegen Ende Unterschiede mit dem blosem Auge nur noch erkennen, wenn die Flächennetze direkt übereinander gelegt sind. Je nach Aufgabenstellung muss eine solche evolutionäre Optimierung (theoretisch) nie enden, nur werden die Verbesserungen pro Zeiteinheit immer geringer. Für den praktischen Nutzen ist es maßgeblich, dass die Zielkriterien so gewählt werden können, dass sie innerhalb des Evolutionsfensters zu erreichen sind, also dort, wo ein gutes Verhältnis von Zeit zu Verbesserung herrscht.

8.5 Parametrische Konstruktionen

Zur weiteren Interpretation der erzeugten und optimierten Flächennetze sind beispielhaft zwei Programmmodule implementiert worden. Das erste Modul verwendet Stahl- und Holz als Materialien und kann als Rippen- und Gittertragwerk ausgeformt werden. Das zweite Modul verwendet ausschließlich Holz und ist nur als Gittertragwerk umgesetzt.

8.5.1 Konstruktionsmodul 1 – Rippen- und Gittertragwerk

Bei diesem Modul werden die Flächenetze als eine Tragstruktur aus Stäben und Knotenverbindungen abgebildet. Die Stäbe sind als Balken aus Holz definiert, die entweder ohne den Einsatz von *CNC*-Maschinen oder unter Verwendung einfacher 3-Achs-Fräsverfahren herstellbar sein sollen. Entsprechend sind alle Holzbalken der Konstruktion rechtwinklig und haben an ihren Enden gerade Schlitze sowie Bohrungen zur Aufnahme und Befestigung der Verbindungselemente. Die Maße dieser Schlitze und Bohrungen sowie die Länge und Breite der Balken sind entlang der u-Richtung einer Fläche identisch, die einzelnen Balken unterscheiden sich lediglich in der Länge. Gleiches gilt für die Balken, die entlang der v-Richtung liegen. Die Köpfe der Balken (Aufnahme Verschraubung, Abschrägung) einer Richtung sowie deren Breite können, müssen sich jedoch nicht von denen der anderen Richtung unterschieden Abb. 30 / 31.

Die Verbindungen zwischen den Balken werden aus flächigen Stahlelementen gebildet. Werden die zu interpretierenden Netze aus Freiformflächen abgeleitet oder sind sie evolutionär optimiert, ist es nahezu sicher, dass es keine zwei (oder mehr) Verbindungselemente geben kann, die exakt die gleiche Form haben. Deshalb ist es zwingend nötig, für eine wirtschaftlich tragfähige Herstellung dieser Teile *CNC*-Verfahren einzusetzen. Die Kontur der Verbinder, eine Kennung, die zur Montage benötigt wird, sowie die Bohrungen für die Verschraubung, sollen per *CNC*-Laserschneidverfahren hergestellt werden. Eine weitere Bedingung, die für eine einfache Herstellung der Stahlverbinder festgelegt wurde, ist, dass sie alle nur eine einseitige Krümmung aufweisen dürfen. Dies ermöglicht einerseits eine einfache Abwicklung der Elemente zur Erstellung der Pfade der Schnittkontur und andererseits eine Herstellung der Biegung mit gängigen *CNC*-Biegemaschinen. Jedes der Verbinderteile hat wieder mit sehr großer Wahrscheinlichkeit einen anderen Biegewinkel, aber immer den gleichen Biegeradius. Die Biegekante kann bereits beim Lasern der Schnittkontur durch zwei kleine Kreise markiert werden.

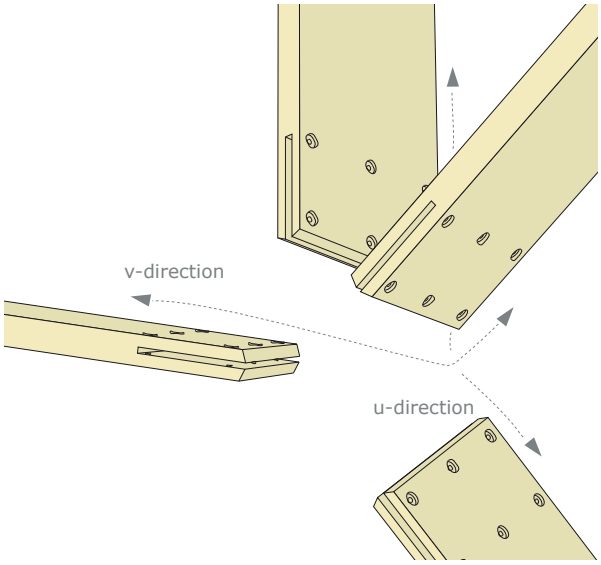


Abb. [30]) Gittertragwerk: Balken der Tragstruktur, in u- und v-Richtung identisch ausgeformt.

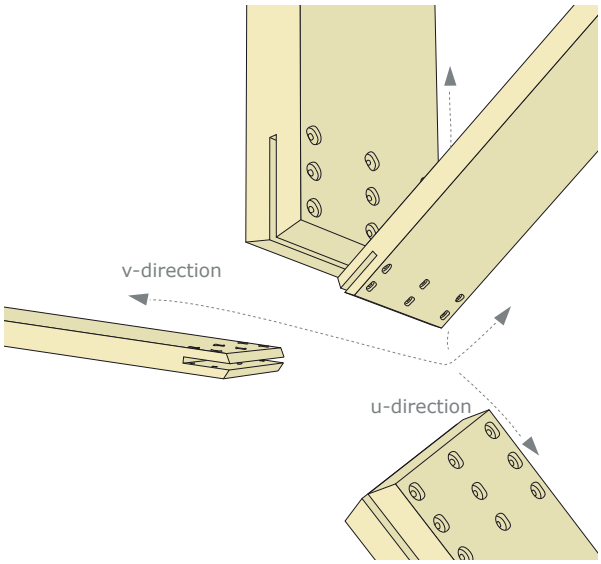


Abb. [31]) Rippentragwerk, Balken der Tragstruktur, in u- und v-Richtung jeweils verschieden.

Je nachdem ob ein Gitter- oder Rippentragwerk umgesetzt werden soll, stehen im Programmmodul zwei Konstruktionsvarianten zur Auswahl. Jeder Stab der Tragstruktur wird mit allen benachbarten Stäben verbunden. Bei Gittertragwerken werden dazu an den Rändern der Fläche zwei, ansonsten vier Verbindungselemente benutzt [Abb. 32], die Tragstruktur kann die Kräfte gleichmäßig in alle Richtungen ableiten. Wenn allerdings die Hauptlast nur entlang einer Flächenrichtung abgeleitet werden kann (Rippentragwerk), ist es vorteilhaft, die Knoten in diese Richtung zu verstärken. Dazu wird ein weiteres Verbinderelement benutzt. Gleichzeitig ist es günstig, die Dimensionierung der Balken, die quer zur Richtung des Lastabtrags liegen, geringer zu halten, damit ein kleineres Eigengewicht entsteht [Abb. 33].

Zur Verschraubung von Balken und Verbindern werden Innensechskantschrauben (DIN 912), entsprechende Beilagscheiben (DIN 440 oder DIN 125 sowie Innengewindehülsen (keine DIN-Normierung) verwendet [Abb. 34]. Durch Ersetzen der entsprechenden Routinen im Konstruktionsmodul könnte auch jede andere geeignete Verschraubungsart realisiert werden.

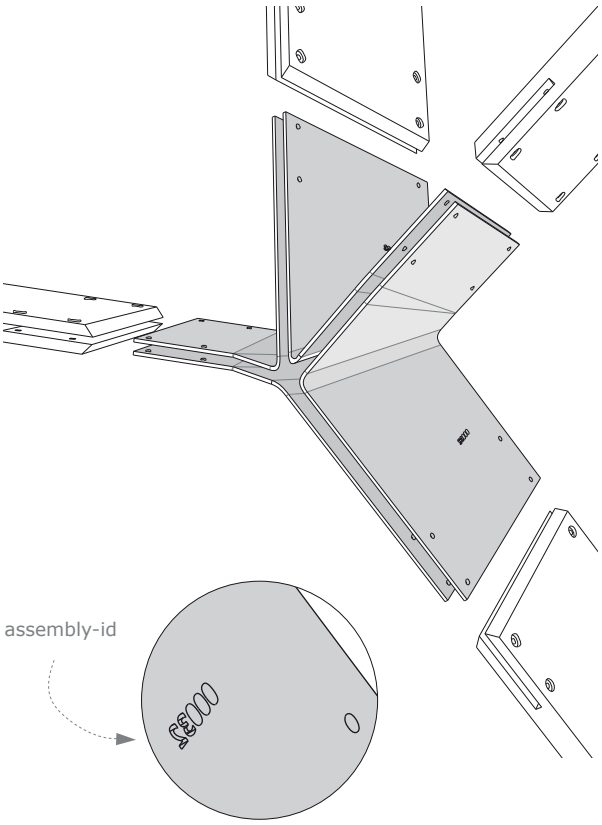


Abb. [32]) Gittertragwerk: Stahlverbinder mit Montagekennung.

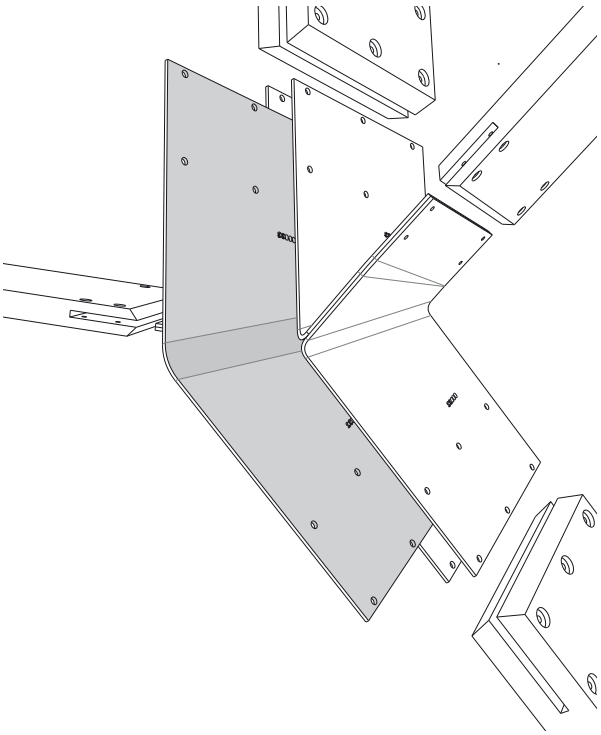


Abb. 33) Rippentragwerk: Stahlverbinder mit zusätzlichem Element.

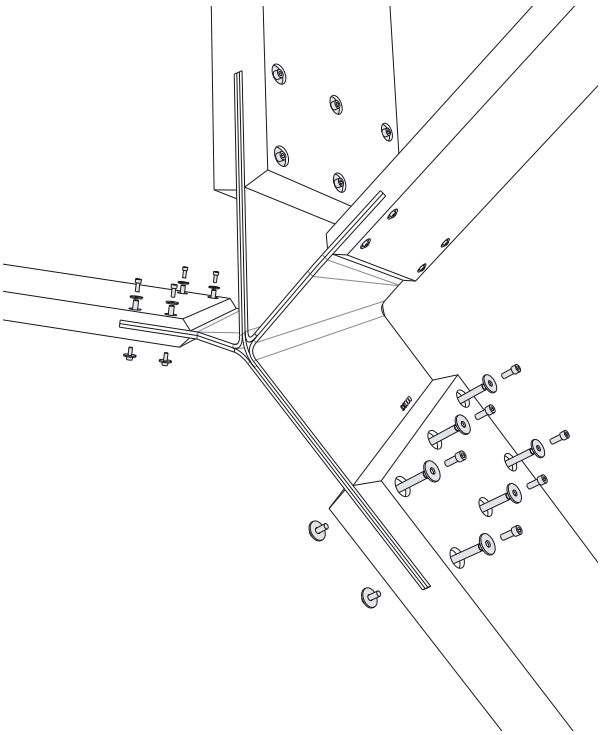


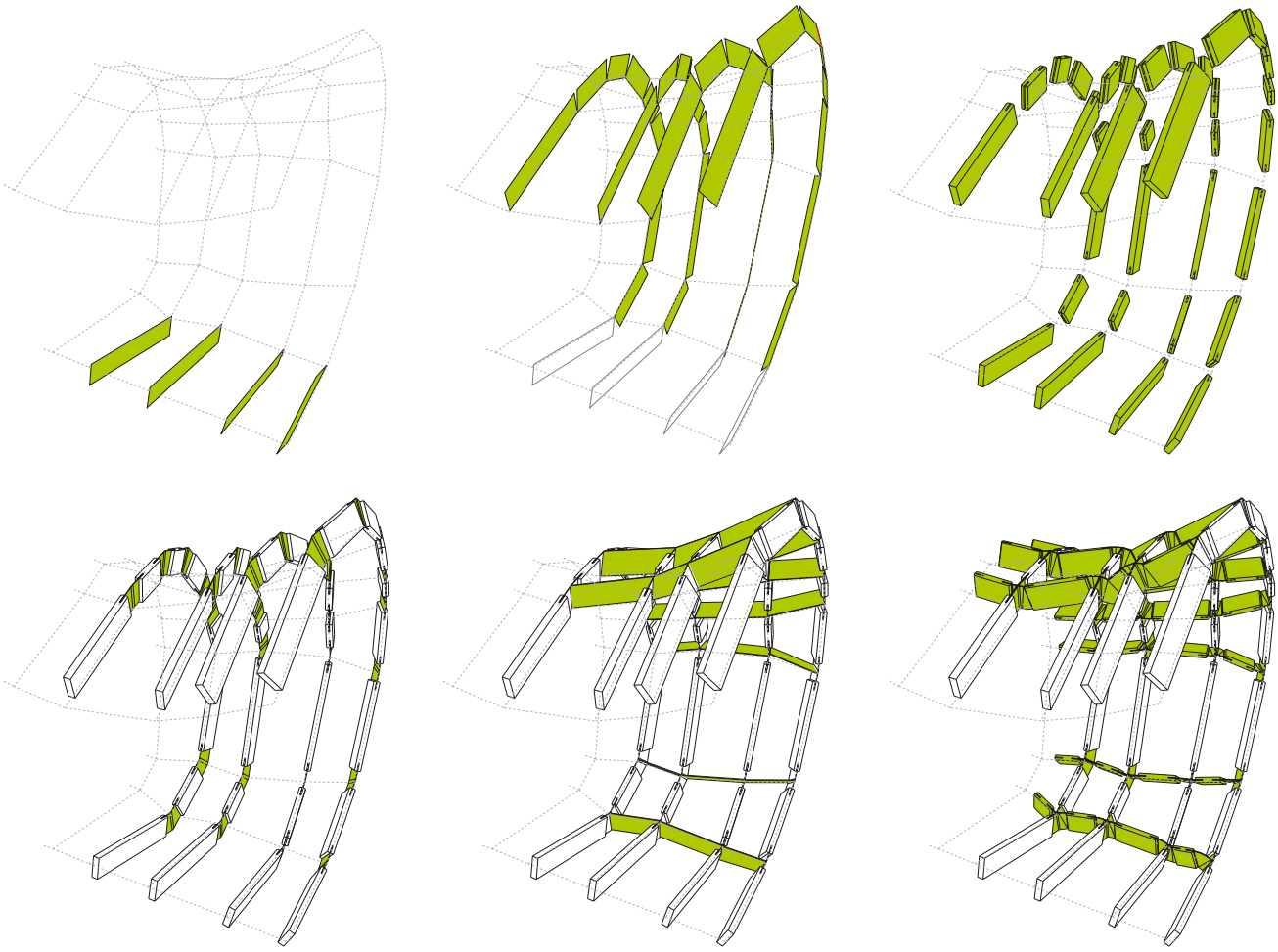
Abb. [34]) Verschraubung von Balken und Verbindern.

Ableitung der Konstruktion aus einem Flächennetz

Die einzelne Bauteile werden (vereinfacht dargestellt) nach folgender Systematik aus den Knoten des Flächennetzes entwickelt: Als erstes werden alle Bauteile der Richtung errechnet, die als Hauptrichtung zur Ableitung der Traglast definiert wurde – falls die Last gleichmäßig abgetragen wird, ist dies eine beliebige Richtung. Zunächst werden entlang der festgelegten Richtung zwischen den Knoten Konstruktionsebenen erstellt, die mittig und entlang der Höhe und Länge der späteren Balken liegen. Die jeweils erste Konstruktionsebene einer Reihe ist entlang der Normalen des ersten Knotens zur Ursprungsfläche ausgerichtet. Alle folgenden Ebenen sind so ausgerichtet, dass sie auf Gehrung aneinander stoßen. Aus diesen Ebenen werden dann alle Stäbe und Verbindungselement dieser Richtung errechnet [Abb. 35].

Anschließend werden die Bauteile entlang der zweiten Flächenrichtung erzeugt. Ausgangspunkt hierfür sind wieder Konstruktionsebenen, die der ‚Mitte' der Balken entsprechen. Diese Ebenen liegen ‚quer' zu den bereits erzeugten Elementen und sind so ausgerichtet, dass sie die Winkelabweichungen, die sich bei den Verbindungen in Querrichtung zwangsläufig ergeben, ausgleichen. Da durch diesen Ausgleich auf einer Seite der Fläche ein inhomogenes Bild entsteht, kann er optional auf der auf der ‚Innen-' oder ‚Aussenseite' der Ursprungsfläche erfolgen, je nach dem, was als Sichtseite definiert wird [Abb. 36]. Aus diesen Ebenen werden dann zunächst die fehlenden Verbinderelement und darauf aufbauend die restlichen Balken errechnet.

Abb. [35]) Ableitung der Konstruktion aus einem Flächennetz.



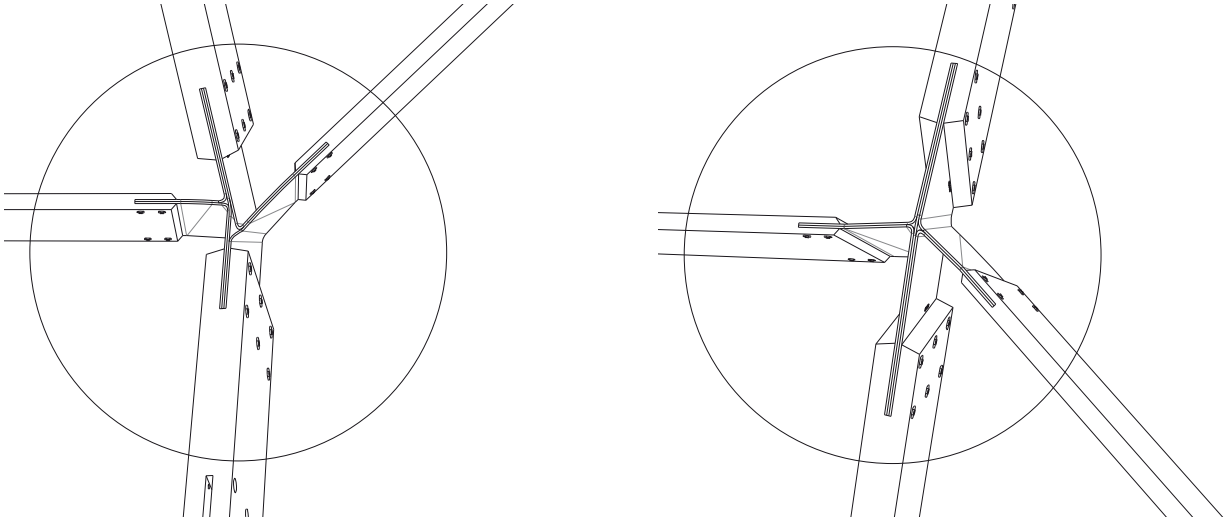


Abb. [36]) Ansicht ‚Innen-' und ‚Aussenseite' der selben Verbinderteile.

Veränderbare Parameter der Konstruktion

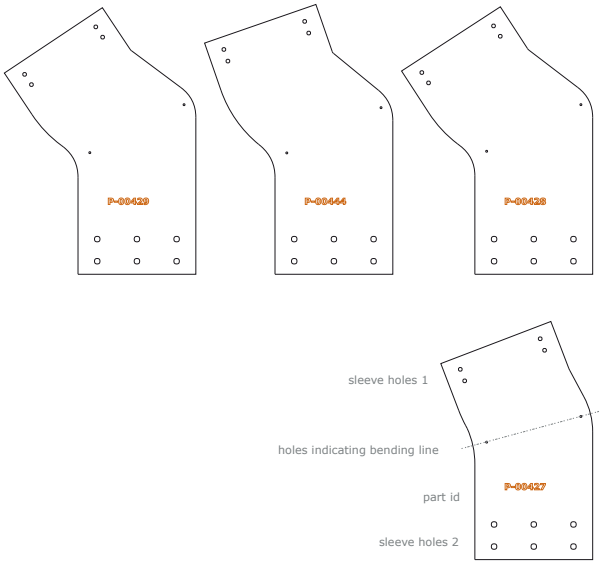
Auch bei diesem Konstruktionsmodul sind die wichtigsten Konstruktionsparameter variabel gehalten und untereinander in Beziehung gesetzt wie in der Abbildung auf der folgenden Seite [Abb. 38] gezeigt.

Ausgabedaten

Das Konstruktionsmodul kann die errechneten Bauteile als 3D-Modelle in verschiedenen Detaillierungsgraden visualisieren. Als Arbeitshilfe werden nach jedem Programmdurchlauf die verwendeten Konstruktionsparameter als Textdatei gespeichert. Als Daten für Fertigung und Montage werden ausgegeben:

- Teileliste für alle Schrauben, Gewindehülsen und Beilagscheiben
- Zuschnittliste für alle Holzbauteile
- Liste mit Biegewinkel aller Stahlteile
- Schnittkonturen für alle Stahlteile [Abb. 37]

Sollen die Holzbauteile mit einer *CNC*-Fräse gefertigt werden, können direkt die erzeugten 3D-Daten verwendet werden. Die Teilekennungen der Stahlteile werden separat augegeben, so dass sie gegebenenfalls auch graviert und nicht, wie die übrigen Pfade, geschnitten werden können.




```
(...)
Part ID: P-00427 --> Angle: -97.75933
Part ID: P-00428 --> Angle:  93.66456
Part ID: P-00429 --> Angle: -86.48427
(...)
Part ID: P-00444 --> Angle: -86.24165
(...)
```

Abb. [37]) Detailansicht Schnittkonturen und entsprechender Auszug aus der Textdatei mit Angaben zu den Biegewinkeln: Negativer Winkel bedeutet es wird in positiver, positiver Winkel es wird in negativer z-Richtung gebogen.

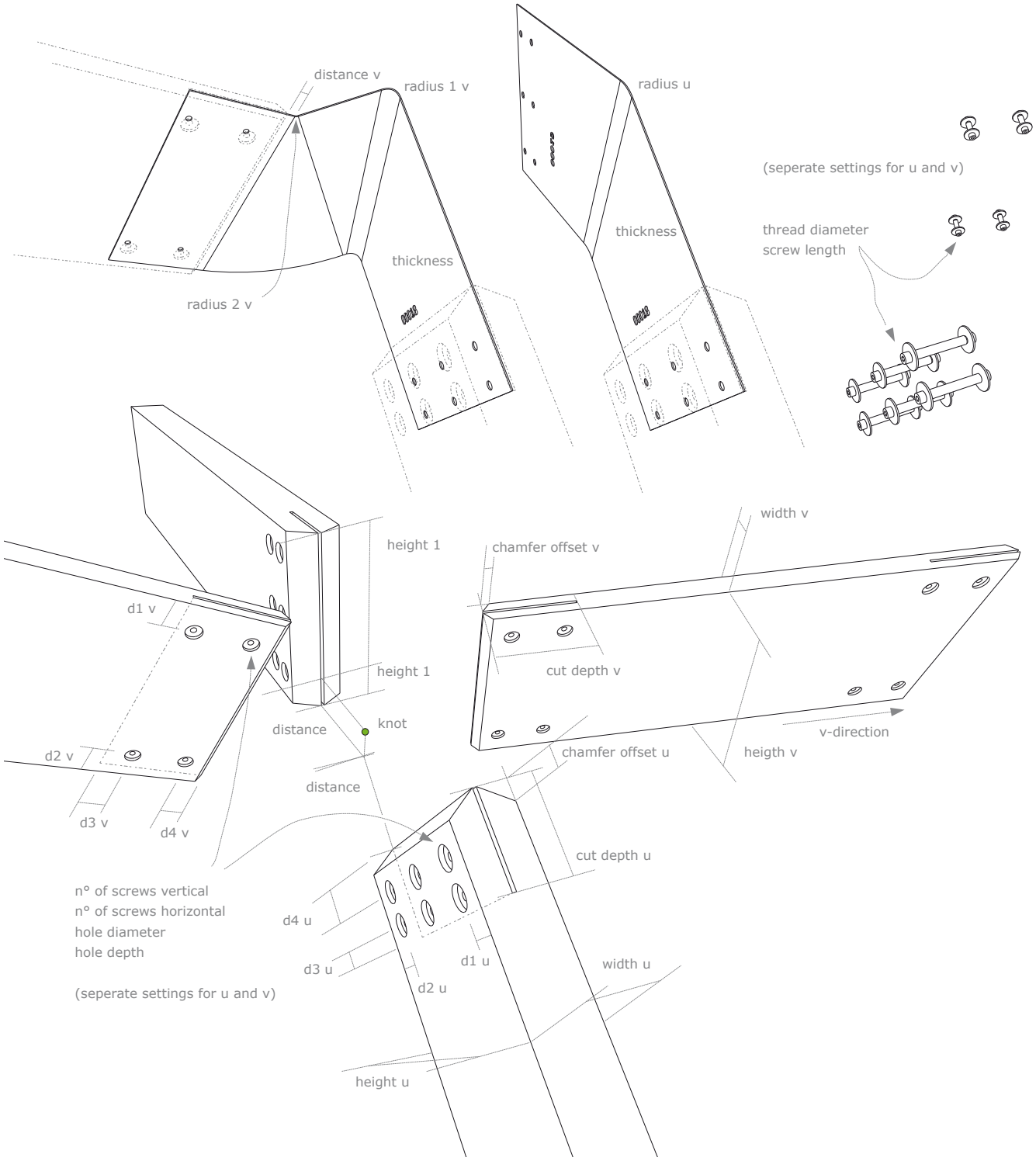


Abb. [38]) Variabel gehaltene Konstruktionsparameter

in Modul 1.

8.5.2 Konstruktionsmodul 2 – Gittertragwerke

Bei diesem Modul wird jedes einzelne Viereck des Flächennetzes als *‚offene Kiste'* aus Plattenmaterial (Holz) abgebildet [Abb. 39]. Dabei sind drei Konstruktionsvarianten möglich. Bei Variante 2 und Variante 3 stoßen die Platten an den Querkanten auf Gehrung aneinander. Bei Nummer 3 wird der Höhenversatz, der bei nicht planaren Vierecken entsteht, dadurch kaschiert, dass Ober- und Unterseite der Längskanten verkrümmt werden. Bei Nummer 2 sind diese Kanten rechtwinklig zu Plattenfläche ausgebildet, der Versatz wird sichtbar. Gleiches gilt für Variante 1, bei der zusätzlich die Querkanten rechtwinklig ausgebildet sind [Abb. 40].

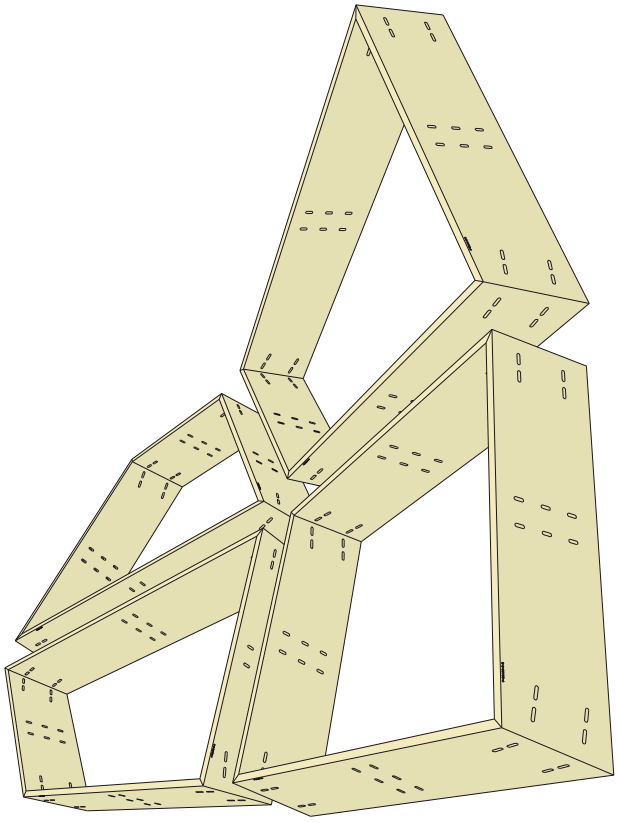


Abb. [39]) Stabelemente der Konstruktion.

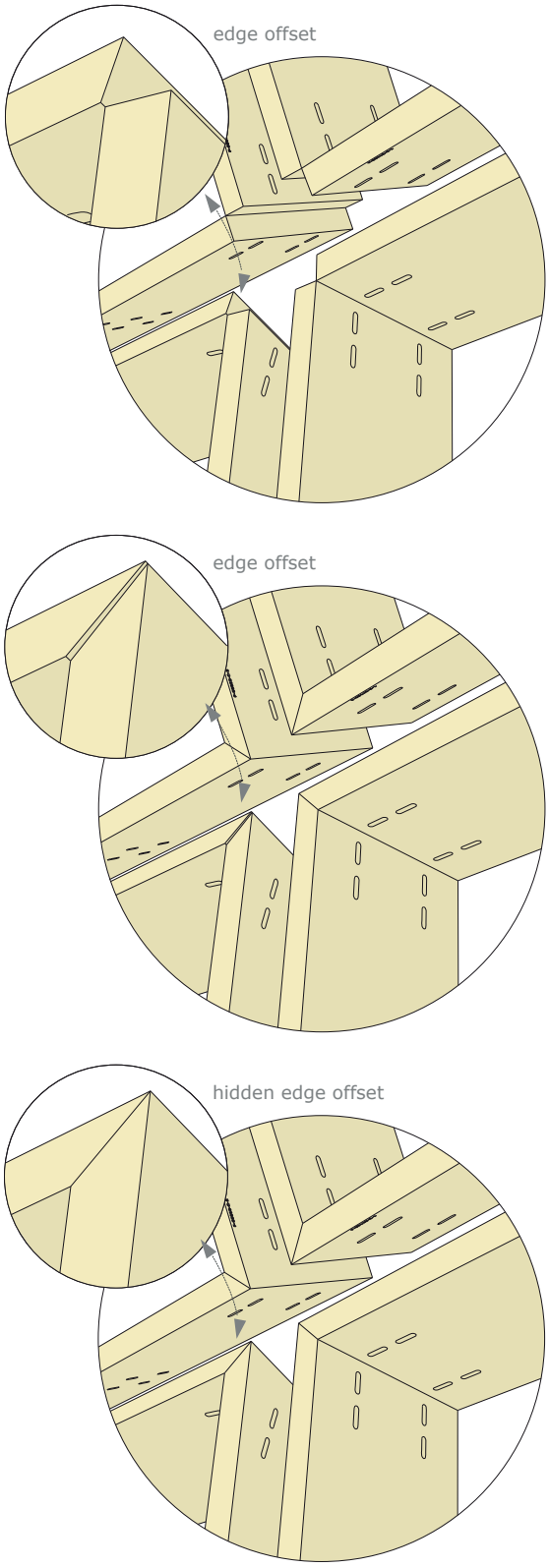


Abb. [40]) Konstruktionsvarianten Eckausbildung.

Die Platten jeder einzelnen Kiste werden über ihre Innenwinkel miteinander verbunden. Dazu werden Verbinderteile aus Holz benutzt, die mit dem Plattenmaterial verzapft werden [Abb. 42]. Die einzelnen Kisten werden über ihre Seiten mit der jeweils benachbarten Kiste verbunden, auch hier werden Bauteile aus Holz eingezapft [Abb. 44]. Bei dieser Konstruktionsart sollen alle Teile mit einer *CNC*-Fräse gefertigt werden. Je nach Konstruktionsvariante wäre dazu eine 3-Achs- oder ein 5-Achs-Maschine notwendig. Alle Verbinder- und Plattenteile haben eine Identifikationsnummer, damit diese 3D-Puzzle auch richtig montiert werden kann [Abb. 41 / 43]. Die Nummer wird direkt beim Fräsvorgang mit eingraviert.

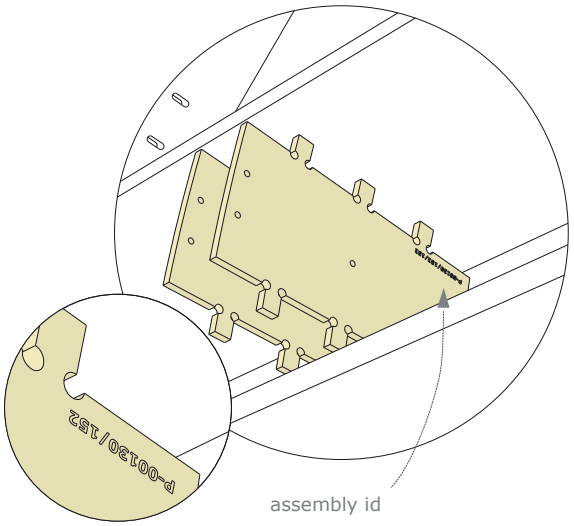


Abb. [43]) Außenverbinder mit Teilekennung.

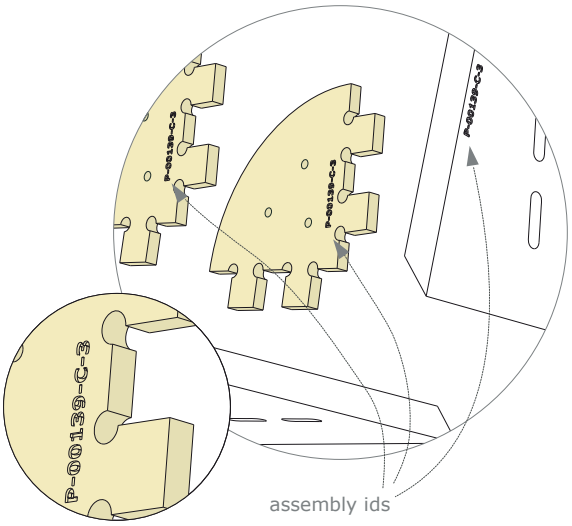


Abb. [41]) Innenverbinder mit Teilekennung.

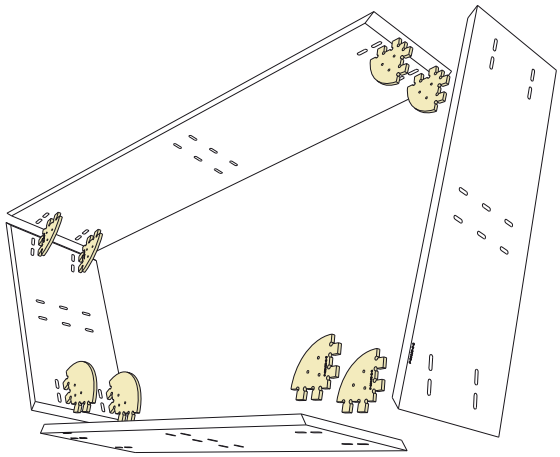


Abb [42]) Anbindung der Innenverbinder an die Plattenelemente.

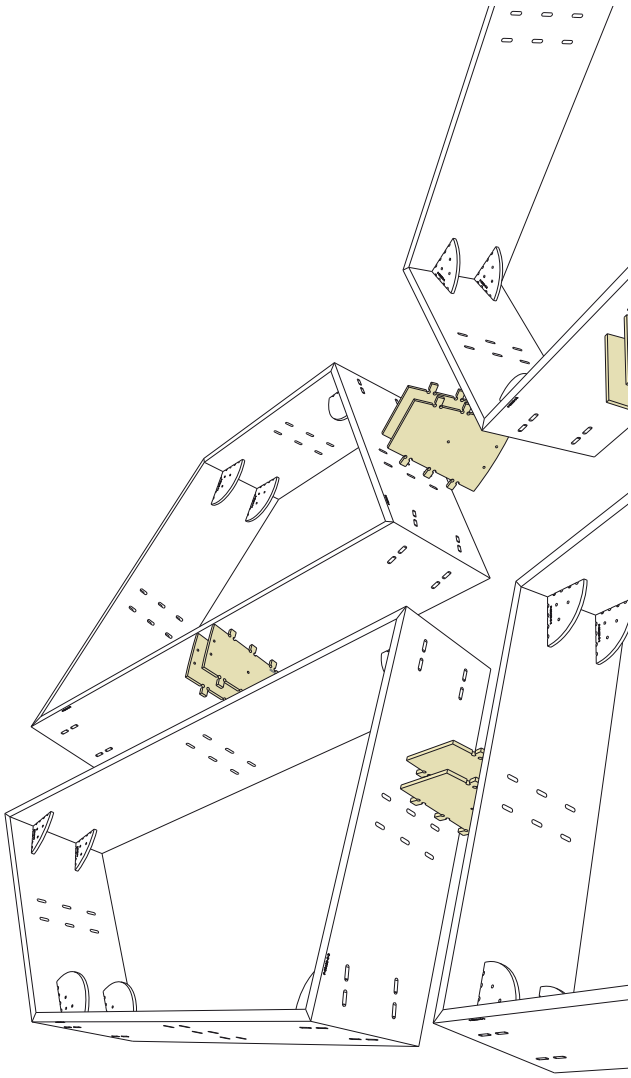


Abb. [44]) Anbindung der Außenverbinder an die Kastenelemente.

Ableitung der Konstruktion aus einem Flächennetz

Zum Errechnen der Bauteile wird hier jede Viereckmasche als zwei gegenüberliegende Dreiecke interpretiert. Im ersten Schritt werden auf den Aussenkanten Ebenen erstellt, die senkrecht zum jeweils zugehörigen Dreieck stehen. Parallel zu jeder Ebene wird eine zweite erstellt, die um die gewünschte Dicke des Materials in Richtung Mitte der jeweiligen Masche verschoben wird. Entsprechend dem definierten Abstand zur Aussenkante wandern nun alle Ebenenpaare noch weiter zur Mitte ihrer Viereckmasche hin. Die Schnittpunkte aller Ebenenpaare einer Masche entsprechen den Eckpunkten der einzelnen Platten eines Kastenelements, entsprechend der dritten Konstruktionsvariante. Für Variante zwei werden die Ober- und Unterseiten der Platten begradigt, für Variante eins noch die Gehrungen abgeschnitten. Dann werden weitere Ebenen erzeugt, die jeweils senkrecht zur Innenseite zweier angrenzender Plattenelemente stehen. Auf diesen Ebenen werden die Innenverbinder konstruiert. Jetzt fehlen noch die Aussenverbinder, sie werden auf Ebenen gezeichnet, die senkrecht zu den Aussenseiten der Platten zweier benachbarter Kastenelemente sind [Abb. 45].

Auch bei dieser Interpretationsart entsteht eine homogene und eine weniger homogen wirkende Seite. Entsprechend kann die Konstruktion entweder zur ‚Innen-' oder ‚Aussenseite' einer Ursprungsfläche angeordnet sein.

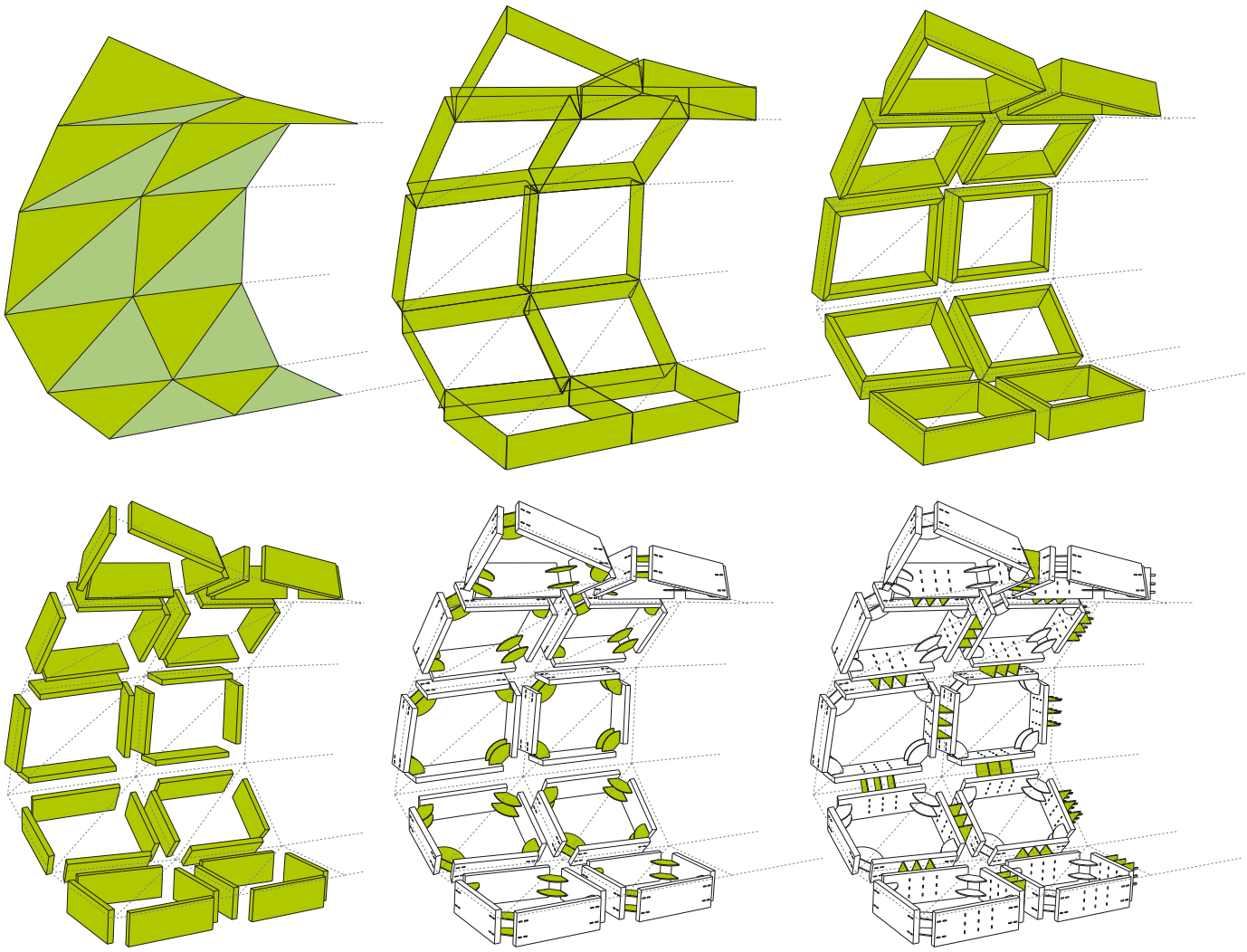


Abb. [45]) Ableitung der Konstruktion aus dem Flächennetz.

Variable Parameter der Konstruktion

Auch hier eine Skizze der wichtigsten variablen Werte der Konstruktion, dargestellt an Variante eins [Abb. 46 / 47].

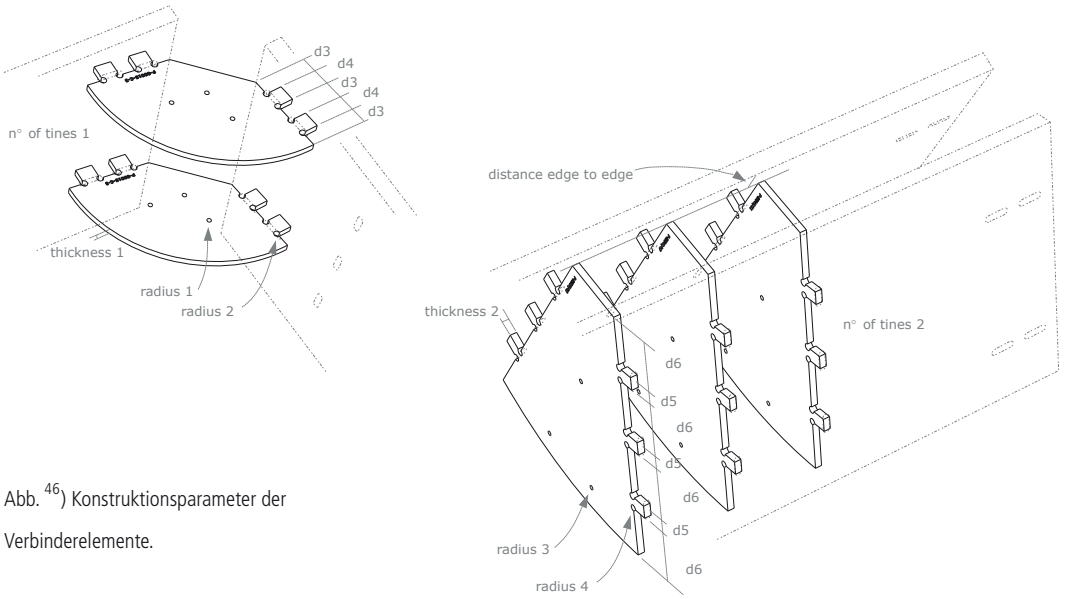


Abb. [46]) Konstruktionsparameter der Verbinderelemente.

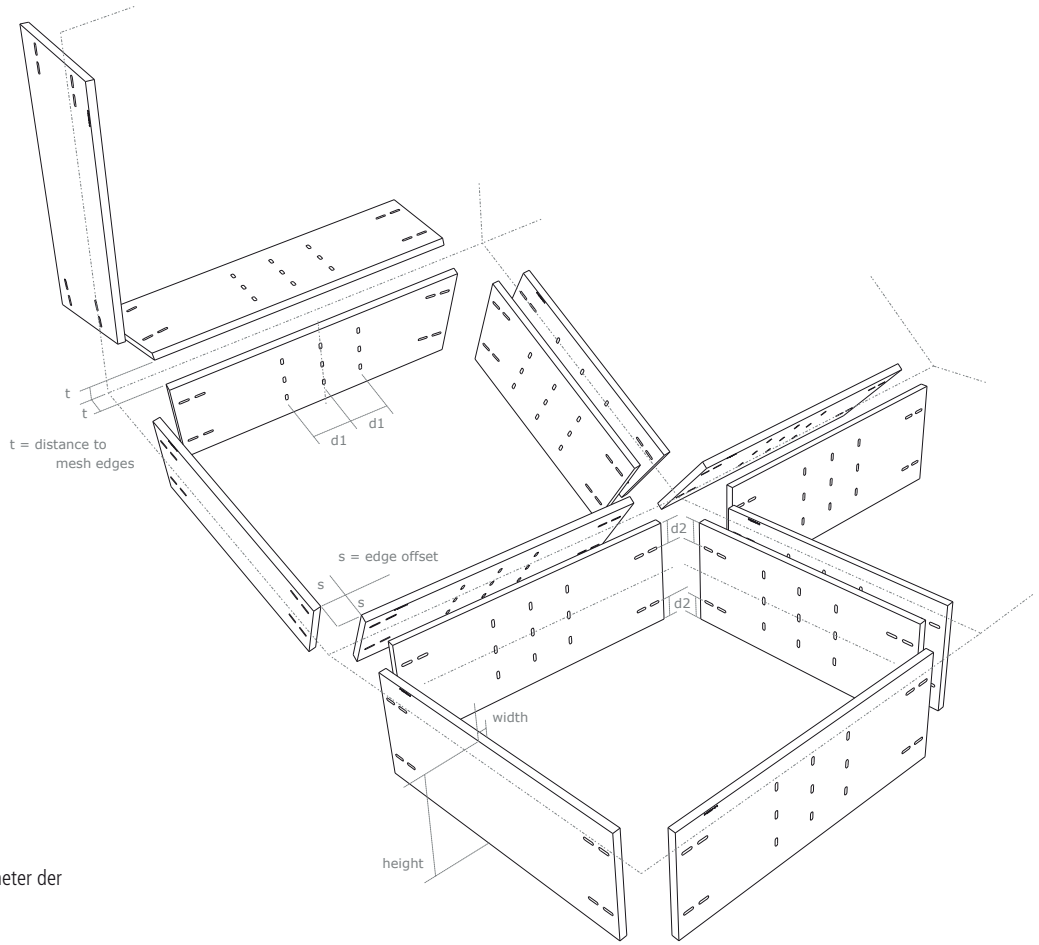


Abb. [47]) Konstruktionsparameter der Kastenelemente.

Ausgabedaten

Alle Bauteile, die für diese Konstruktion notwendig sind, können direkt aus den zugehörigen 2D- bzw. 3D-Daten gefertigt werden: Für die Verbinderteile genügen in jedem Fall 2D-Fräspfade [Abb. 48]. Ob für die Kastenelement Fräspfade oder 3D-Modelle erzeugt werden, hängt von der gewählten Konstruktionsvariante ab.

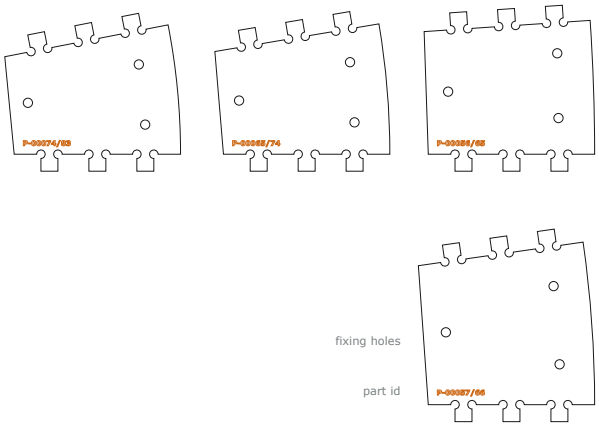


Abb. [48]) Fräspfade für Verbinderelement mit Teilekennung und den optionalen Löchern zum Fixieren kleiner Teile auf dem Frästisch.

8.6 Ergebnisse und Beispiele

8.6.1 Genetischer Algorithmus

Flächennetze aus gleichmäßig verteilten Viereckmaschen erwecken den Eindruck von geometrischer Homogenität, vermitteln eine Idee von korrekter Statik und auf den ersten Blick auch von einfacher konstruktiver Umsetzung mit bewährten Mitteln – kurz, solche Viereckmaschnetze sehen einfach vertrauenserweckend und entwerferisch beherrschbar aus. Diese Sichtweise stimmt, aber nur für manche Flächentypen, nicht aber in jedem Fall für Freiformfläche, bei denen ein regelmäßiges Netz aus viereckigen Maschen nicht unbedingt die geometrisch, konstruktiv oder statisch beste Lösung liefert. Die Viereckmaschnetze hingegen, die der genetische Algorithmus entwickelt hat, zeigen irritierende, unpräzise wirkende geometrische Muster, wirken teilweise instabil und konstruktiv schwierig in den Griff zu bekommen. Dieser Eindruck ist jedoch trügerisch. Evolutionär optimierte Flächennetze, sowohl von Freiform- als auch von mathematischen Flächen, können für bestimmte geometrische und mechanische Parameter bessere Eigenschaften aufweisen als die regelmäßigen Netze, wie die weiter unten beschriebenen Versuche zeigen. Was die Rationalisierung der Bauteile von Freiformflächen anbelangt, ist sehr häufig jedes Teil ohnehin ein Unikat, unabhängig davon, ob regelmäßige oder evolutionär entwickelte Netze der Ausgangspunkt sind.

Der genetische Algorithmus ist für zwei konkrete Fälle eingesetzt und getestet worden: Zur Untersuchung der Optimierungsleistung hinsichtlich statischer sowie geometrischer Kriterien, mit einem Blick darauf, wie das Werkzeug der evolutionären Optimierung auch als Entwurfswerkzeug einsetzbar wäre.

Optimierungsbeispiel 1 - Statische Optimierung

In dieser Versuchsreihe wurde untersucht, inwieweit sich die evolutionäre Optimierung verwenden läßt, um das statische Verhalten von Flächennetzen zu verbesseren. Messgröße war die maximale Knotenverschiebung im Netz, die über *RSTAB*® ermittelt wird und als Fitnesstest im genetischen Algorithmus implementiert ist. Alle Netze wurden unter Eigenlast, mit festen Auflagern (ohne Freiheitsgrade für eine Bewegung, also als feste Fundamente) berechnet. Die Stäbe waren als Stahlrundrohre (ST 37) mit 60mm Durchmesser und 2,3mm Wandstärke definiert, bei Flächenausdehnungen, die sich grob in einem Volumen von 25 x 10 x10 Metern bewegen. Die Stabknoten waren ebenfalls als feste Knoten, also wieder ohne Bewegungsspielraum definiert.

Die Kenngrößen für den Optimierungsprozess sind so festgelegt worden, dass sie zwar nach ‚Daumenpeilung' in einer realistische Größenordnung liegen, gleichzeitig aber noch Potential zur Verbesserung der statischen Struktur vorhanden ist. Wenn die Bauteile für die statische Evaluierung

bereits so definiert worden wären, dass keine oder nur noch marginale Knotenverschiebungen auftreten würden, könnte dieses Kriterium auch keinen selektiven Druck innerhalb des Evolutionsvorgangs ausüben. Die Festlegung auf Stahlrohre bedeutet auch nicht, dass optimierte Netze zwangsläufig damit umgesetzt werden müssten – es kann auch jedes andere Material verwendet werden, solange die drei Bedingungen ‚feste Auflager', ‚feste Knoten' und ‚Stabtragwerk' erhalten bleiben. Das Ergebnis der konstruktiven Umsetzung muss in jedem Fall noch einer statischen Überprüfung unterzogen werden, diesmal unter realistischen Lastfällen, mit den korrekten Materialkennwerten und Dimensionierungen. Die zur Umsetzung entwickelten Konstruktionsmodule erlauben es, die Größen der Bauteile in einem gewissen Maß, entsprechend einer solchen zweiten Überprüfung, anzupassen.

Neben dem Fitnesstest für die maximale Knotenverschiebung wurden noch drei weitere, geometrische Parameter evaluiert: Vorgegebene Mindestlängen für die Stäbe in beiden Flächenrichtungen und ein Maximalwert für die Längenabweichungen der Diagonalen. Die Zielgrößen für diese Fitnesstests variieren von Fläche zu Fläche und wurden aus regelmäßig aufgeteilten Flächennetzen, die gleichzeitig zum Vergleich der Stabverschiebung benutzt wurden, abgeleitet. Dazu wurde ein Modus in den Ablauf des genetischen Algorithmus implementiert, der bei der Evaluation von regelmäßigen Netzen die geometrischen Testkriterien in bestimmten Stufen ändert, bis sie von jeder Masche eines Netzes erfüllt werden. Damit nun bei der evolutionären Optimierung die Flächenmaschen einerseits etwas mehr Bewegungsspielraum haben, um sich in Richtung einer statischen Verbesserung entwickeln zu können, andererseits die Form der Ursprungsfläche noch erkennbar bleibt, wurden die so ermittelten Werte für die minimalen Kantenlängen halbiert, der Wert für die Diagonalenabweichung ist beibehalten worden.

Die Ergebnisse dieser Versuchsreihe, die an sechs verschiedenen Flächen mit jeweils einer Aufteilung von 8 mal 8 Maschen durchgeführt wurde, sind auf der gegenüberliegenden Seite zusammenfassend dargestellt [Abb. 49]. Bis auf den Kugelabschnitt war es bei allen Beispielen möglich, die maximale Knotenverschiebung in einer Größenordung von knapp 30 bis zu über 90 Prozent im Vergleich zu den regelmäßigen Netzen zu verringern. Dabei sind zwei Effekte deutlich erkennbar. Einmal entwickeln sich die Netze so, dass Auskragungen verkürzt werden, zum anderen werden Stäbe, die in einer Reihe angeordnet sind und potentielle Knicklinien bilden, in eine Art Zick-Zack-Anordnung gebracht, die ein Abknicken verhindert [Abb. 50]. Diese Effekte sind ebenfalls bei der Fläche zu sehen, die in nebenstehender Abbildung [Abb. 49] in der zweiten Reihe von unten gezeigt ist. Hier hat die evolutionäre Optimierung gegenüber der regelmäßigen Aufteilung, die ein eindeutiger Versagensfall ist (rund 9m maximale Knotenverschiebung bei etwa gleicher Höhe der Struktur), ein Netz hervorgebracht, das nur noch rund 1,8m maximale Knotenverschiebung aufweist. Dies ist sicher noch kein Stabtragwerk, das so baubar ist, es zeigt aber deutlich das Potential der evolutionären Strategie. Eine ähnlich große Verbesserung hat sich bei der untersten Fläche in der nebenstehenden Abbildung [Abb. 49] gezeigt, bei der eine maximalen Knotenverschiebung von knapp 1,8m im regelmäßigen Netz einem Wert von 13cm im evolutionär entwickelten Netz gegenübersteht.

Als deutliches Manko in den Versuchen hat sich die verwendete Programmierumgebung (*VisualBasic for Applications*®) und der notwendige, ständige Wechsel zwischen verschiedenen Softwareanwendungen erwiesen, was zu relativ langen Rechenzeiten geführt hat. Jeder einzelne der dargestellten Versuche ist mit 15 Individuen (eine Population) bei 500 Generationen gerechnet worden, was etwa einen Tag Zeit benötigte, für die sechs Beispiele also sechs Tage. Man kann an der Entwicklung der globalen Fitness [Abb. 51] erkennen, dass damit das Potential der evolutionären Optimierung nicht ausgeschöpft ist. Ein Verhältnis von 150 Individuen, aufgeteilt in drei Populationen und 1500 Generationen hat sich in vergleichbaren Vortests in der *VB-Script*-Umgebung als eine Größenordnung erwiesen, die ein annehmbares Verhältnis von Rechenzeit zu Optimierungsleistung bringt.

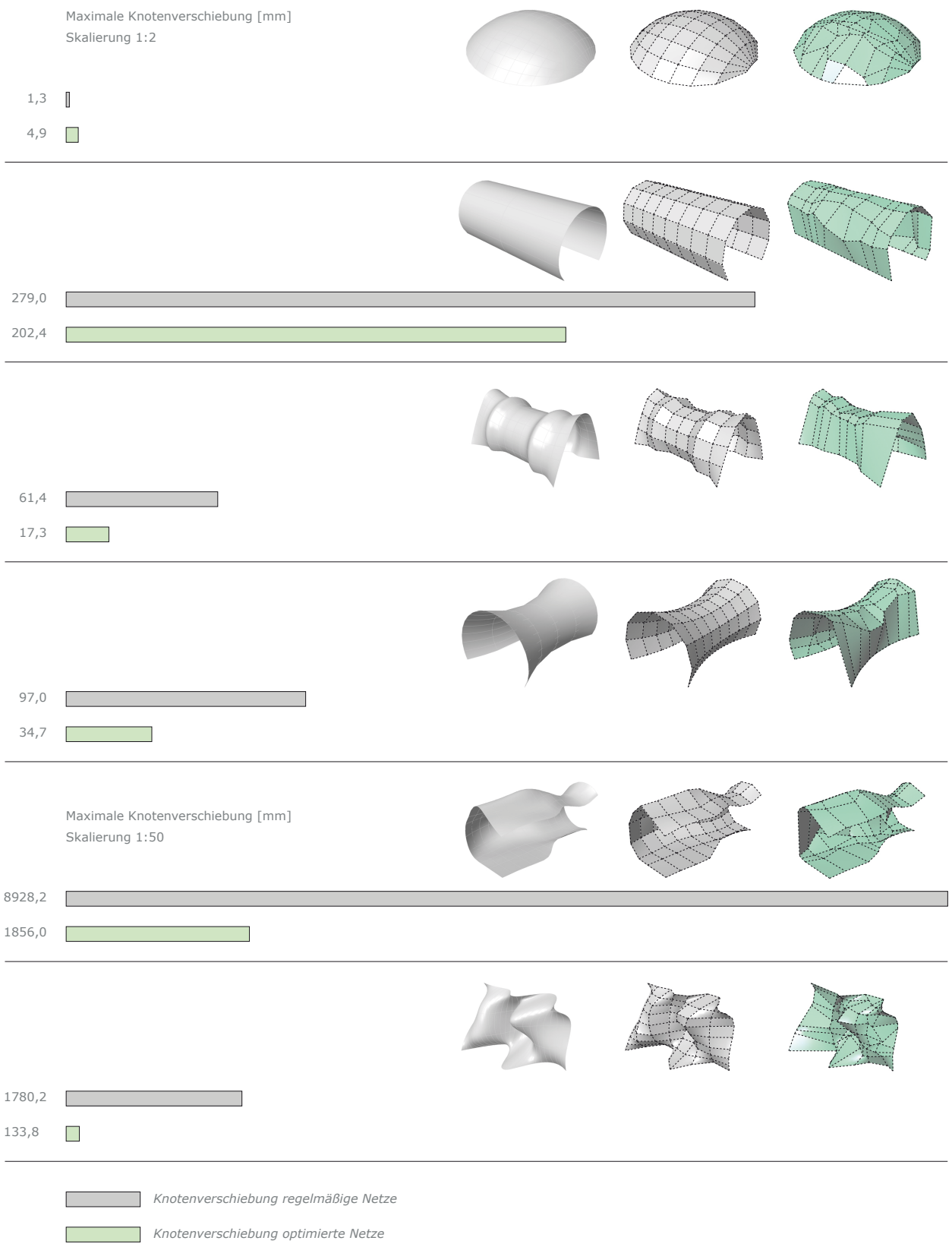


Abb. [49]) Ergebnisse einer Testreihe zur statischen Optimierung von Flächennetzen durch einen genetischen Algorithmus.

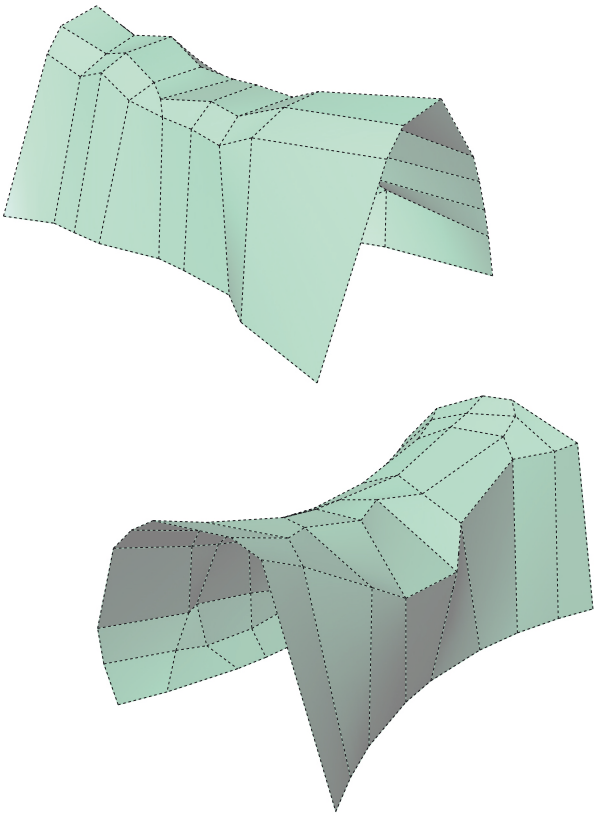


Abb. [50]) Ausbildung von Zick-Zacklinien entlang potentieller Knicklinien, Verkürzung von Auskragungen durch lange Maschenkanten.

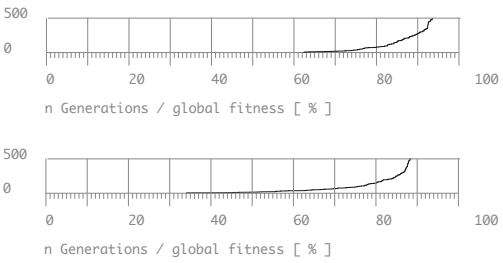


Abb. [51]) Entwicklung der globalen Fitness für die beiden Beispiele aus Abbildung 50.

Umgerechnet auf die Optimierung der Knotenverschiebung ergäbe dies eine Rechenzeit von rund 60 Tagen für die sechs Versuche. Damit wird nicht die Aussagekraft dieser Versuche geschmälert, ganz im Gegenteil: Entsprechend längere Rechenzeiten würden die erzielten Ergebnisse signifikant verbessern. Ein schneller, spielerischer Umgang mit dieser Methode, wie sie dem Entwerfen in frühen Phasen angemessen wäre, ist damit allerdings nicht möglich – im Vergleich: Eine der in *Kapitel 7* gezeigten Hockergeometrien ist im Durchschnitt, inklusive Topologieoptimierung, konstruktiver Interpretation und Ausgabe der **.stl*-Datei, in rund 15 Minuten erzeugt.

Optimierungsbeispiel 2 - Geometrische Optimierung

In diesem zweiten Beispiel wurde die evolutionäre Optimierung eingesetzt, um Flächennetze zu generieren, bei denen eine möglichst große Anzahl an Maschen durch viereckige Scheiben aus planarem Isolierglas belegt werden kann. Dieser Ansatz hat zunächst einen wirtschaftlichen Hintergrund. Planare Scheiben sind wesentlich günstiger herzustellen als gekrümmte. Im Vergleich zu Dreiecksscheiben, die natürlich immer planar sind, haben Viereckscheiben einmal den Vorteil, dass bei gleicher belegbarer Fläche und ungefähr gleicher Kantenlänge auch nur die Hälfte an Scheiben benötigt wird. Dies bekommt zusätzlich Gewicht dadurch, dass beim Schneiden von Dreieckscheiben die Bruchgefahr höher ist, insbesondere bei spitzen Winkeln. Nicht zuletzt sind Viereckснetze wegen der gleichmäßigen Knotenwertigkeit auch insgesamt konstruktiv günstiger als Dreiecksnetze.

Die Berechnung der maximalen Durchbiegung der Maschen erfolgt näherungsweise, wie in *Kapitel 8.4.2* beschrieben. Die Versuchsreihen, die hier durchgeführt wurden, haben einen ähnlichen Aufbau wie die zur statischen Optimierung. Zum Vergleich dienen ebenfalls regelmäßige Netze, als weitere Fitnesstests wurden wieder die minimalen Kantenlängen sowie die Diagonalenabweichung genutzt. Gegenüber den regelmäßigen Flächennetzen wurde der Zielwert für die Kantenlängen bei jeder getesten Fläche um jeweils 20% reduziert, der Wert für die Diagonalenabweichung um 10% erhöht. Bei diesen Experimenten wurde auch die Option genutzt, die Maschenknoten entlang ihrer Normalen zur Ursprungsfläche zu bewegen. Der Wert für die maximale Verschiebung wurde aus den regelmäßigen Netzen ermittelt. Er entspricht dem Durchschnittswert der

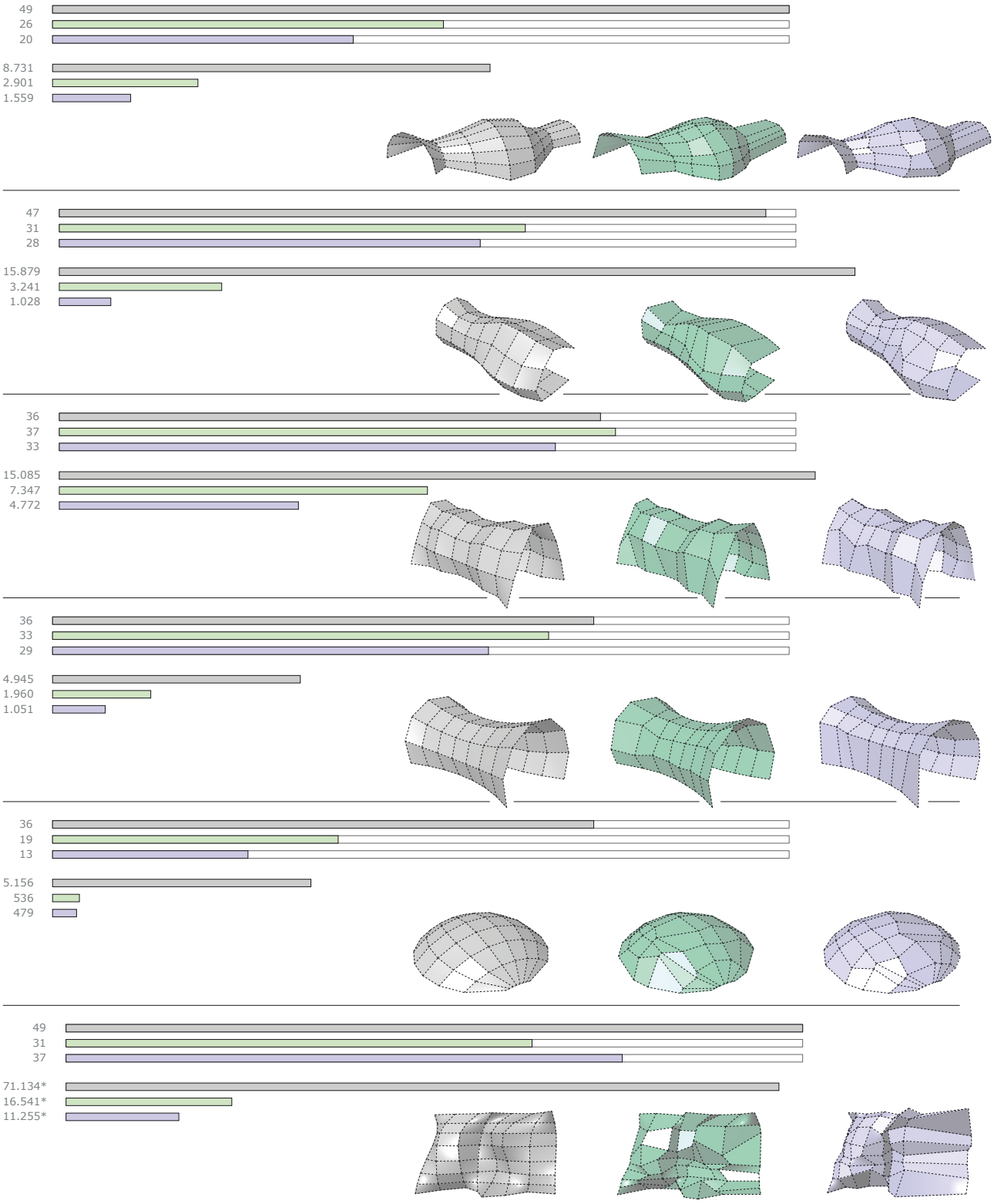


Abb. [52]) Ergebnisse einer Testreihe zur geometrischen Optimierung von Flächennetzen durch den genetischen Algorithmus.

Abweichung der Flächenmaschen, die außerhalb der zulässigen Biegetoleranz liegen. Es wurden drei Varianten erprobt: keine Verschiebung, Verschiebung um 50% sowie um 100% des ermittelten Durchschnittswerts.

Die evolutionäre Vorgehensweise hat durchgehend Flächennetze hervorgebracht, die bezüglich einer möglichst geringen Durchbiegung der einzelnen Maschen bessere Eigenschaften aufweisen als regelmäßig aufgeteilte Netze. Bei den auf der vorhergehenden Seite gezeigten Beispielen Abb. 52, die alle eine Flächenaufteilung von 7 x 7 Maschen haben, hatten die evolutionär optimierten Netze immer mehr Maschen innerhalb der gewünschten Biegetoleranz. Auch war die Summe der Abweichungen der Netzmaschen, die noch außerhalb der Toleranz lagen, immer geringer. Ebenfalls gut in dieser Abbildung Abb. 52 zu sehen ist, dass die Ergebnisse besser werden, je größer die erlaubte Abweichung der Knoten von der Ursprungsfläche ist. Dieser Vorteil wird aber ab einem gewissen Punkt wieder dadurch aufgehoben, dass bei zunehmender Abweichung auch die Gefahr steigt, dass sich Maschen überschneiden und dadurch nicht mehr baulich umsetzbar sind – hier ist fallabhängig eine Balance zu finden.

Über die dargestellten Ergebnisse hinaus wurde noch eine Reihe weiterer Versuchsreihen mit verschiedenen Maschenzahlen durchgeführt (11x11; 15x15; 23x11). Bei allen Versuchen hat sich das obige Ergebnis bestätigt – die evolutionäre Optimierung erzeugt immer mehr Flächenelemente innerhalb der Krümmungstoleranz als die mit gleicher Maschenzahl aufgeteilten regelmäßigen Flächennetze. Dies gilt auch für eine besonder Aufteilungsvariante, die an zwei Flächen erprobt wurde – allerdings mit einigen Einschränkungen. Hier wurde zunächst ein regelmäßiges Netz mit 15 x 15 Maschen erzeugt, dann jede Masche, die außerhalb der Biegetoleranz war, in vier weitere Maschen unterteilt usw., solange, bis schließlich jede Masche durch eine Viereckscheibe belegbar war Abb. 53. Bei der auf der Folgeseite Abb. 54 / links gezeigten Fläche ergab dieses Verfahren eine Zahl von 16.701 Flächen, wobei die kleinsten Kantenlängen bei rund 80mm, die größten bei rund 1350mm lagen. Die zweite Fläche Abb. 54 / rechts wurde gar in Mosaiksteinchen zerlegt (Längen von 15mm bis 3.250mm), bei einer Maschenzahl von 57.643. Die erzeugte Zahl an Maschen ist absurd hoch und außerhalb jeder wirtschaftlichen Fertigungsperspektive. Dennoch zeigt sich bei beiden Beispielen der sehr schöne Effekt, dass die Elementierung die Kurvatur der Fläche abbildet – je größer die lokale Krümmung, desto mehr Flächen werden in diesem Bereich erzeugt.

Abb. 53) Regelmäßige Zerlegung einer Fläche nach einer fraktalen Logik – jede Einzelfläche, die nach der Aufteilung außerhalb der Krümmungstoleranz liegt, wird im nächsten Schritt in vier weitere Flächen unterteilt.

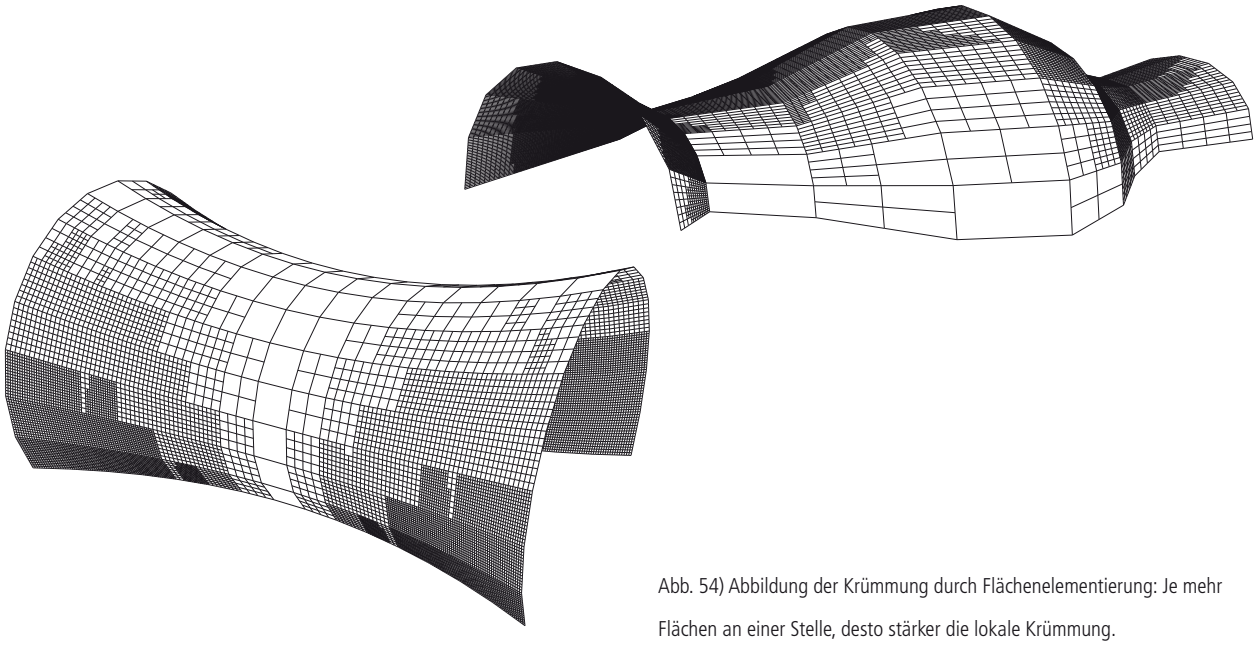


Abb. 54) Abbildung der Krümmung durch Flächenelementierung: Je mehr Flächen an einer Stelle, desto stärker die lokale Krümmung.

Die Flächennetze, die einer evolutionären Optimierung unterzogen wurden, sind nach der gleichen Systematik wie oben aufgeteilt, nur dass eben nach jedem neuen Unterteilungsschritt versucht wurde, die Krümmung der resultierenden Maschen durch den genetischen Algorithmus zu minimieren. Die Abbildung auf der folgenden Seite [Abb. 56] zeigt, wie hier, im Gegensatz zu den regelmäßig aufgeteilten Netzen, ein unregelmäßiges Muster entsteht, in dem sich der Krümmungsverlauf der Ursprungsfläche nur noch erahnen läßt. Dieser Effekt kommt dadurch zu Stande, dass jede weniger stark gekrümmte Masche an anderer Stelle eine stärker gekrümmte Masche erzeugt. Die Gesamtkrümmung der Fläche kumuliert also auf einzelnen Maschen, der nächste Iterationsschritt zeigt den gleichen Effekt usw., was schließlich ein Flächennetz mit unregelmäßigen Mustern erzeugt. Die Kehrseite davon ist, dass so sehr kleine, stark gekrümmte Maschen entstehen, die sich kaum mehr in annähernd planare Elemente auflösen lassen. Bei den in den Abbildungen [Abb. 55 / 56] gezeigten Beispielfläche ist die evolutionäre Optimierung bis zur 4. Iteration im Vorteil: Hier sind dann rund 92% der Gesamtfläche in rund 7.000 planare Elemente aufgelöst, während bei der regelmäßigen Aufteilung knapp 80% in rund 5.000 planare Elemente aufgeteilt sind. Dannach beginnt das Ganze zu kippen: Im nächsten Schritt ist bei der regelmäßigen Aufteilung die Gesamtfläche in fast planare Elemente aufgelöst, während die evolutionäre Optimierung auch im 8. Schritt noch Lücken enthält – bei einer Maschengröße, die in etwa der von Dichtungsprofilen entsprechen dürfte.

Obwohl mit diesen Experimenten der Anspruch aufgegeben wurde, auch baulich umsetzbare Netze zu erzeugen, markieren sie gleichzeitig den Übergang von einem Optimierungs- zu einem Entwurfswerkzeug. Natürlich ist es aus entwerferischer Perspektive wünschenswert, Netze zu erzeugen, die in irgendeiner Form mit der Logik eines Ursprungsfläche spielen, etwa mit der Krümmung oder einem simulierten Kräftefluss. Gleichzeitig kommt man mit den hierfür entwickelten Werkzeugen wieder in einen Bereich, in dem man Tage bis Wochen auf ein Ergebnis warten muss – was für einen Vorgang, der am Anfang einer entwerferischen Prozesskette steht, schlicht zu lange ist. *Rechenberg (1994)* [11] stellt fest, dass an einer Optimierungsstrategie, bei der die meiste Zeit dafür aufgewendet werden muss, den Algorithmus selbst zu optimieren, etwas nicht ganz in Ordnung ist. In diesem Sinne und im Kontext der Flächennetze hat sich der genetische Algorithmus als Optimierungswerkzeug bewährt. Als Entwurfswerkzeug zum schnellen Experimentieren und Varianten bilden ist er, so wie er hier eingesetzt wurde, jedoch nur bedingt geeignet.

Abb. [55]) Fraktale Aufteilungslogik in Kombination mit evolutionärer Optimierung.

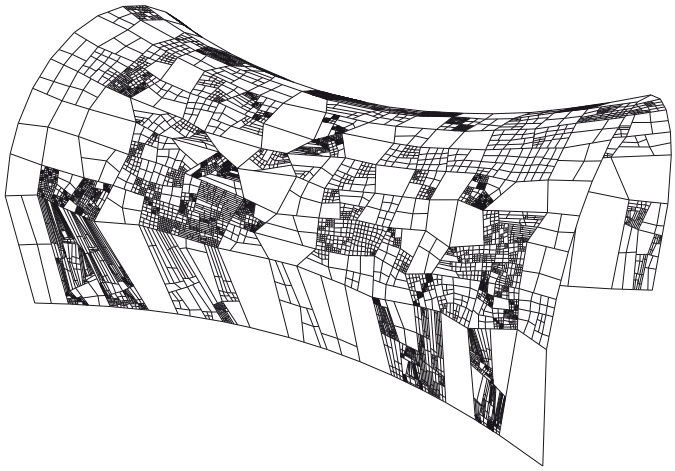


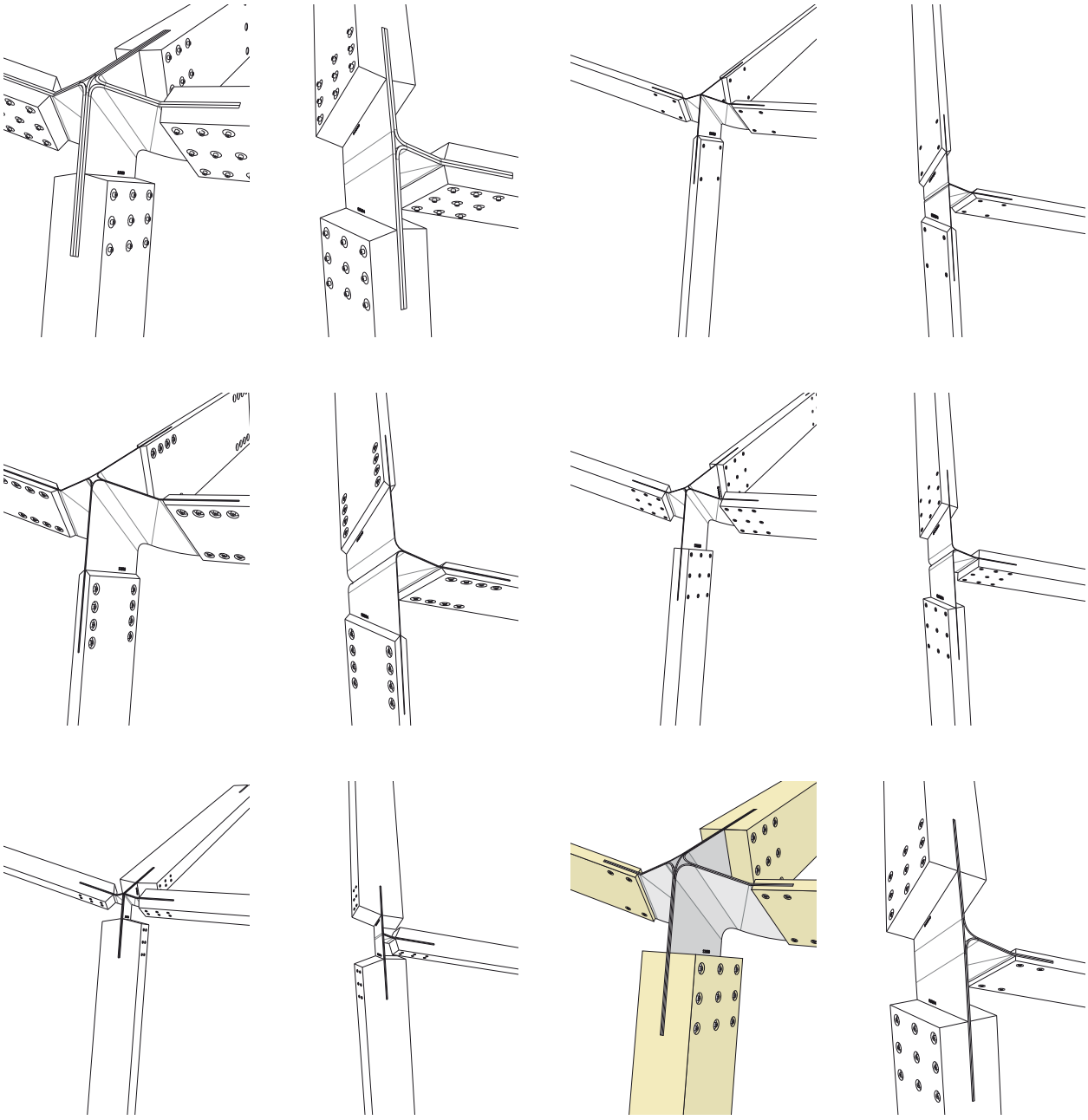
Abb. [56]) Der evolutionäre Optimierungsvorgang entwickelt eine eigene Krümmungslogik, bei der lokale Krümmungen auf einzelnen Flächenelementen kummulieren und ein unregelmäßiges Muster erzeugen.

8.6.2 Konstruktionsmodule

Die Konstruktionsmodule, die hier entworfen wurden, unterscheiden sich von denen, die für die Topologieoptimierung entwickelt wurden, durch drei Aspekte: Erstens ist die grundlegende Programmstruktur einfacher, da wegen der anderen Aufteilungssystematik die konstruktive Interpretation in weniger komplizierten, doppelten Schleifen organisiert werden konnte, ohne dass dabei die Nachbarschaftsbeziehungen zwischen den Netzelementen geprüft werden müßten. Zum zweiten basiert die konstruktive Umsetzung hier ausschließlich auf geometrischen Parametern, die Logik der Optimierung ist hier bereits durch die Form des Netzes abgebildet und wird nicht durch weitere Werte repräsentiert. Abgewandelte Formen der Konstruktionsmodule, die in der Lage sind, die im bereits optimierten Netz auftretenden Kräfte direkt in eine entsprechende Dimensionierung der Bauteile umzusetzen sind in der Entwicklung, aber noch nicht soweit gediegen, dass sie für diese Arbeit präsentabel wären. Der dritte Unterschied besteht darin, dass die hier verwendeten Module geometrisch komplexer sind, was durch die eingangs definierte Anforderung entsteht, dass alle Bauteile auf planare Elemente zurückgeführt werden müssen.

Die folgenden Abbildungen zeigen zunächst eine Reihe von Variationen, die über verschiedene Konstruktionsparameter erzeugt wurden. Die Darstellungen eines Moduls sind jeweils im selben Größenverhältnis gehalten. Gezeigt werden jeweils die selben Elemente aus der Fläche: Bei Modul 1 [Abb. 57] ein vierwertiger sowie ein dreiwertiger Knoten, teilweise als Gittertragwerk, teilweise als Rippentragwerk ausgebildet, bei Modul 2 [Abb. 58] vier komplette Maschen.

Abb. 57) Parametrisches Design – sechs Konstruktionsvarianten aus Programmmodul 1, in gleicher Skalierung dargestellt.



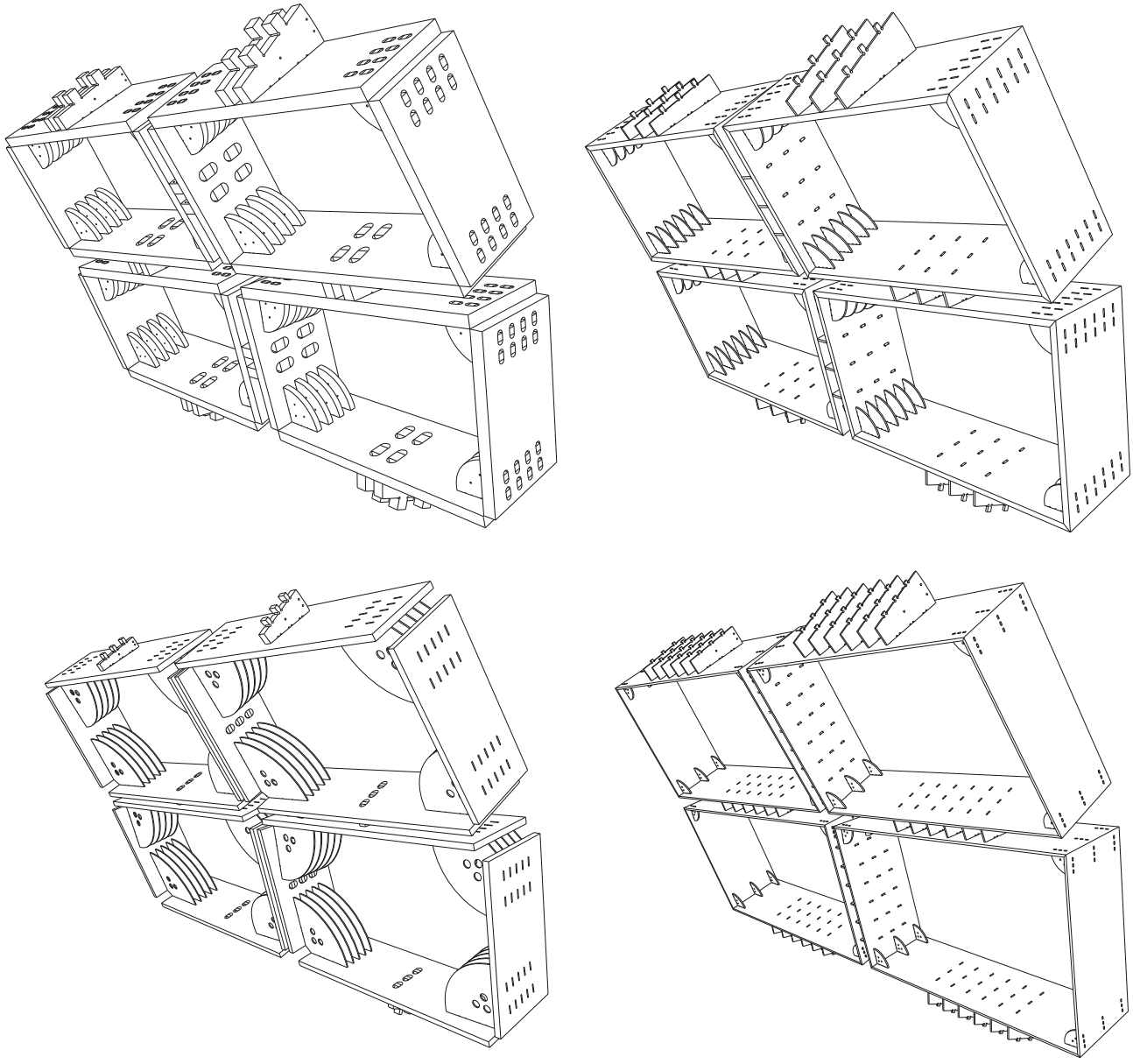


Abb. [58]) Parametrisches Design – vier Konstruktionsvarianten aus Programmmodul 2, wieder in gleicher Skalierung und Ansicht.

Die folgenden Beispiele zeigen verschiedene Anwendungen der Konstruktionsmodule auf regelmäßige sowie evolutionär optimierte Flächennetze. In nebenstehender Abbildung [Abb. 59] wurde die parametrische Konstruktion zunächst auf zwei verschiedene Flächen angewandt, die nach einer regelmäßigen Aufteilung in 9 x 14 Maschen unterteilt wurden.

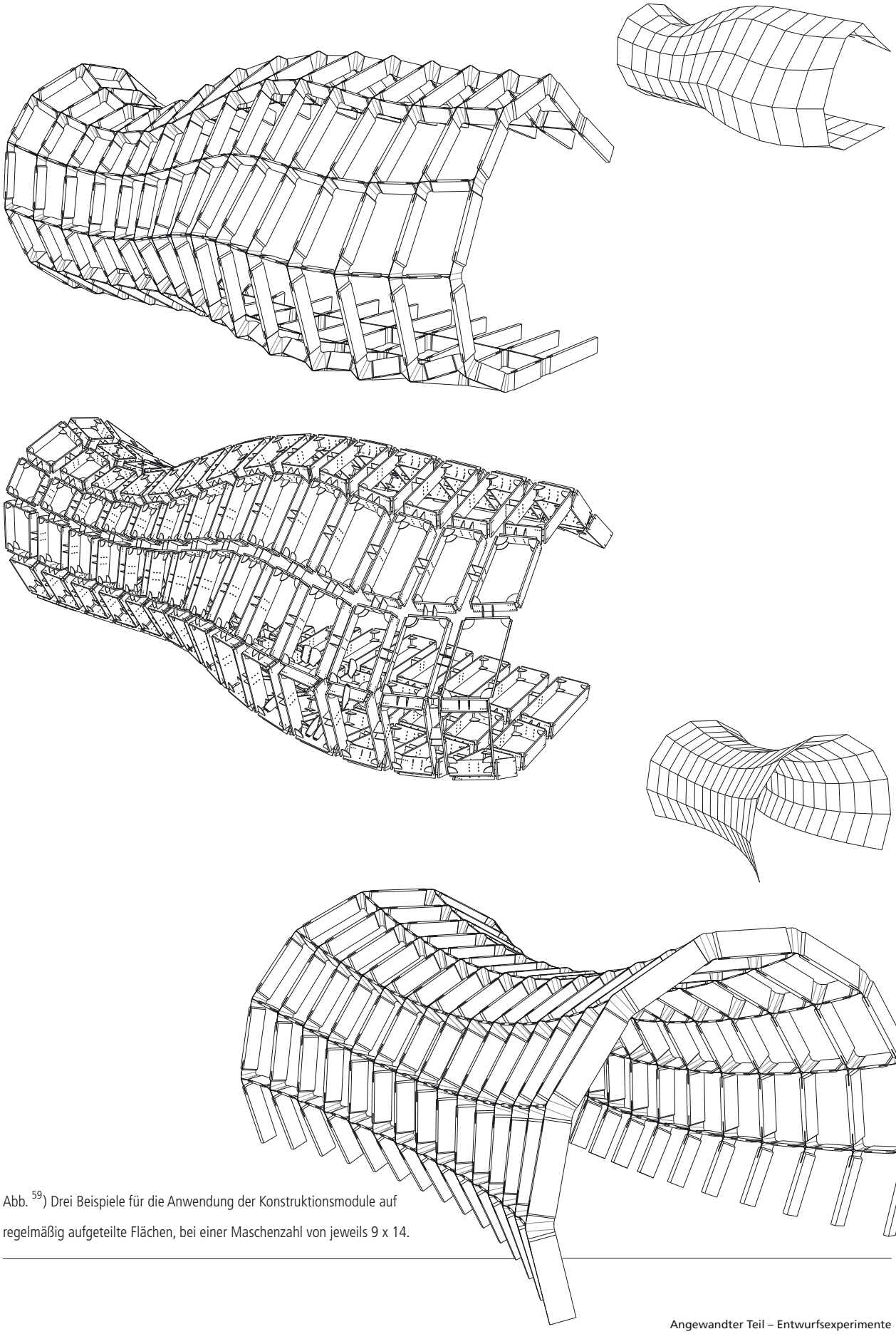


Abb. [59]) Drei Beispiele für die Anwendung der Konstruktionsmodule auf regelmäßig aufgeteilte Flächen, bei einer Maschenzahl von jeweils 9 x 14.

Die nachfolgende Abbildung [Abb. 60] zeigt die Anwendung des ersten Konstruktionsmoduls auf eine optimierte Fläche, die den evolutionären Prozess zweimal mit identischen Einstellungen durchlaufen hat. Die resultierenden Strukturen haben eine ganz andere Anmutungsqualität als die aus den regelmäßigen Netzen. Zudem zeigt sich auch hier die Besonderheit des evolutionären Vorgehens, bei dem jedes erzeugte Ereignis ein Unikat ist und auch nicht wiederholt werden kann – sind die erzeugten Daten verloren, ist damit auch der Entwurf nicht mehr reproduzierbar.

Die letzten Abbildungen [Abb. 61 / 62] schließlich geben noch einen Eindruck der Komplexität, die durch eine Entwurfsanssatz entsteht, der nur wenige oder gar keine sich wiederholenden Bauteile hervorbringt – und von den entsprechenden Schwierigkeiten, die hier bei einer Montage auftreten können. Dargestellt ist ein kleiner Teil von Verbinderelementen aus beiden Modulen, wie sie bei den hier gezeigten Beispielen typischerweise entstehen, sowie die Überprüfung der virtuellen algorithmischen Entwurfsräume am physischen Modell.

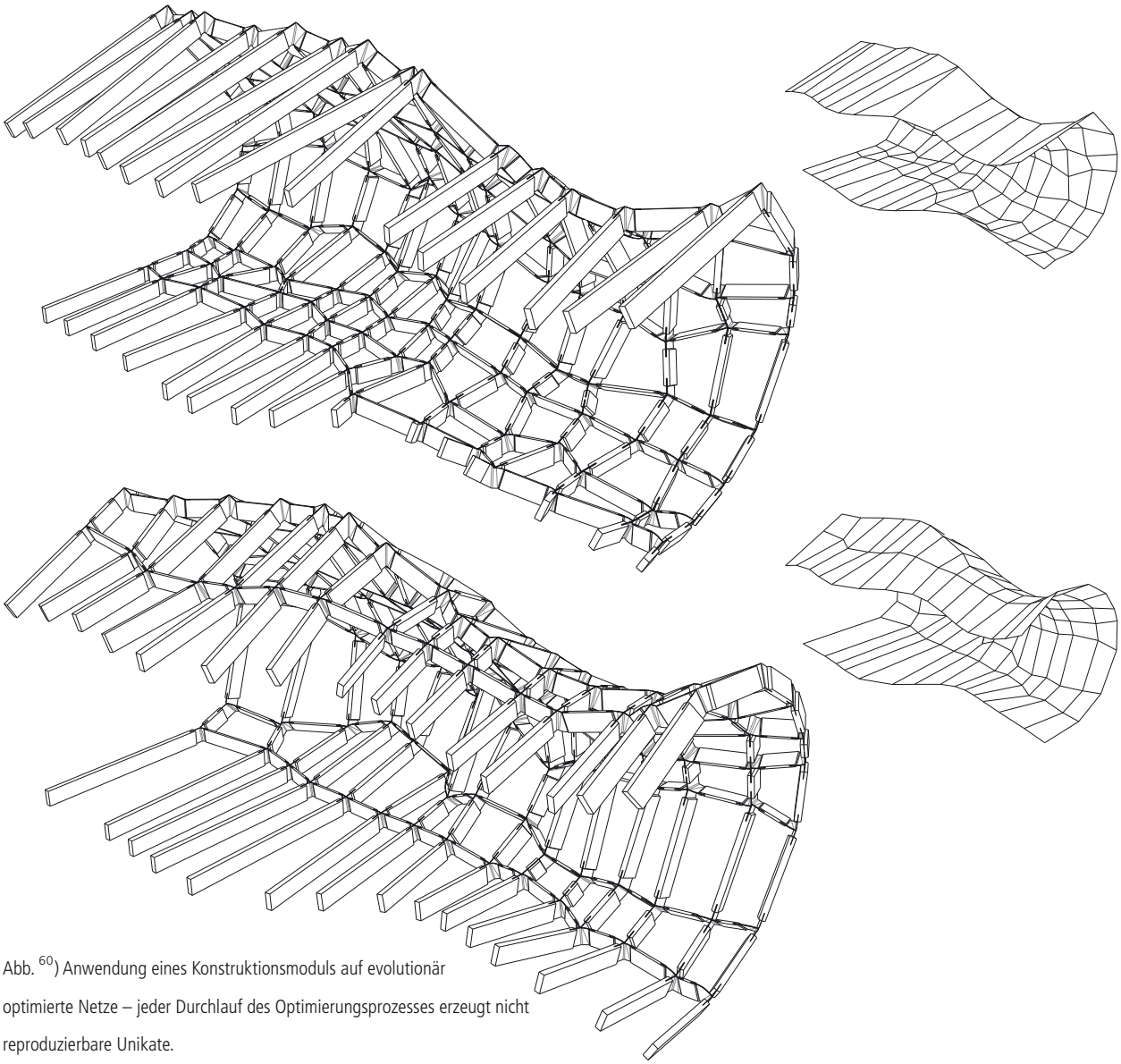
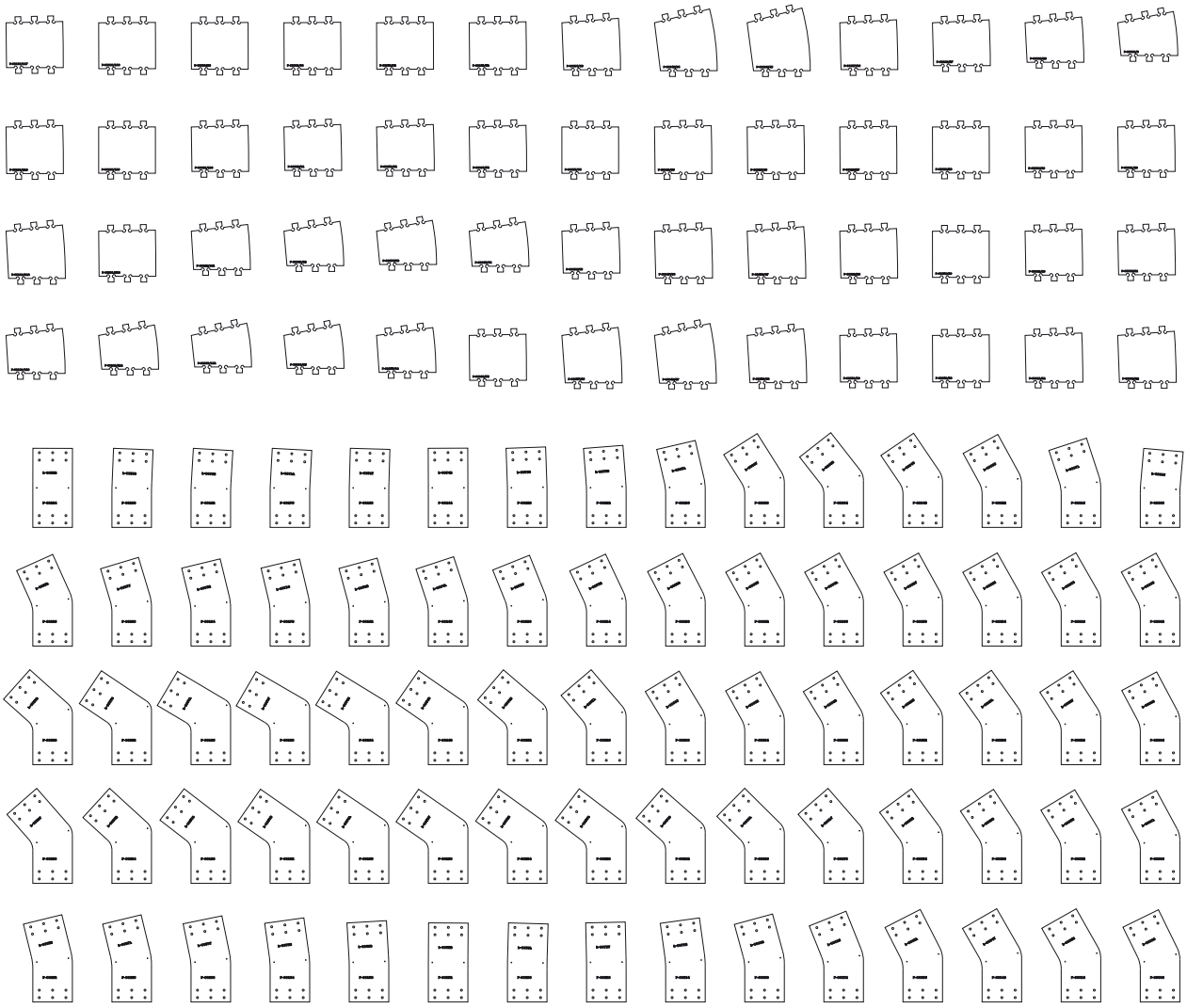
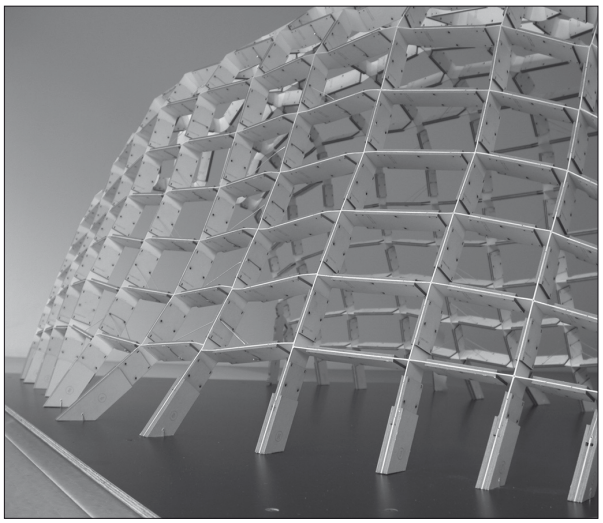


Abb. [60]) Anwendung eines Konstruktionsmoduls auf evolutionär optimierte Netze – jeder Durchlauf des Optimierungsprozesses erzeugt nicht reproduzierbare Unikate.





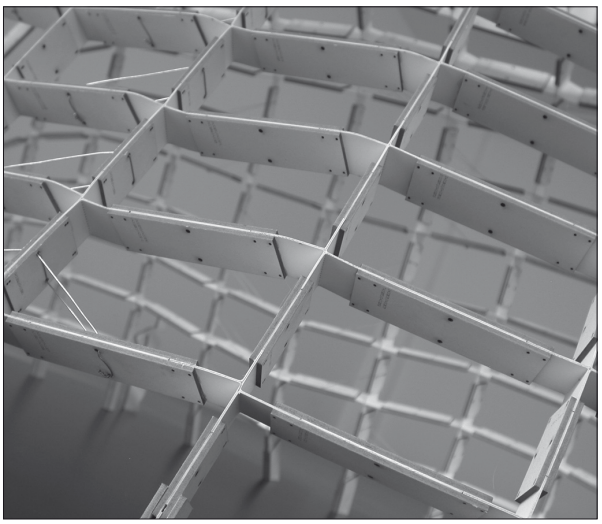


Abb. [61]) Bauteile aus den gezeigten Beispielen. Jedes Teil als Unikat, einmal als Darstellung der Laserpfade sowie ähnliche Teile im Modell.

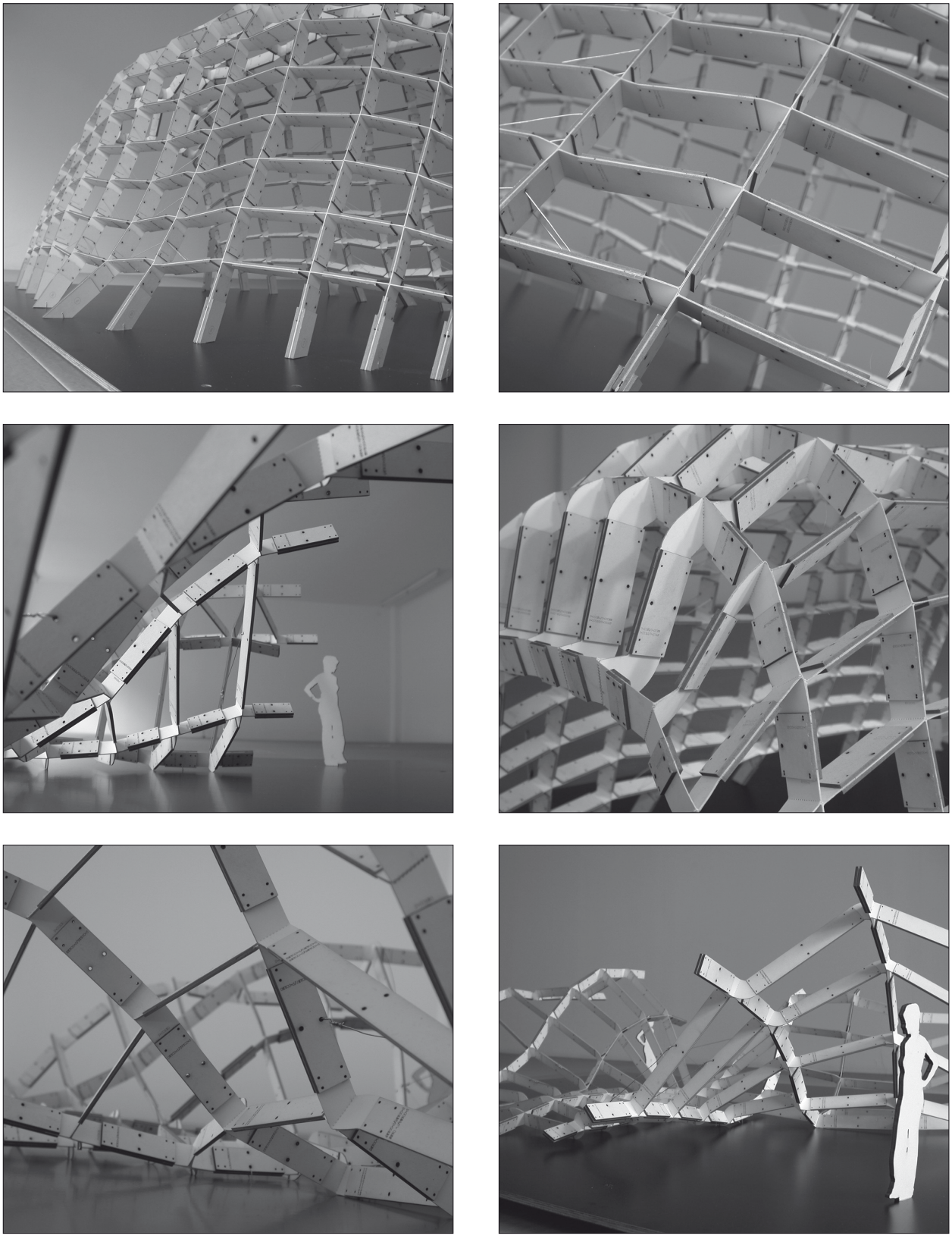


Abb. [62]) Überprüfung der virtuellen Entwurfsräume
an physischen Modellen.

9. Entwurfsräume 3 – Erweiterungen und Rekombinationen

9.1 Kurzes Resümee algorithmischer Entwurfsräume

Die Entwicklung und das Arbeiten mit den oben vorgestellten algorithmischen Werkzeugen hat bestätigt, was eingangs der Arbeit als Hypothesen zu den algorithmischen Entwurfsräumen formuliert und später verfeinert wurde. Die entwerferische Arbeit fokussiert sehr schnell auf das Aufdecken und Festlegen einiger (überwiegend geometrischer und mechanischer) Rahmenbedingungen. Auf dieser Basis werden Konzepte für Algorithmen entwickelt und damit ist der eigentliche entwerferische Prozess auch bereits weitgehend abgeschlossen. Gravierende Änderungen am Entwurfskonzept werden im Fortlauf der Programmierung (u.a.) aus ökonomischen Gründen sehr schnell unwahrscheinlich, der Ideenreichtum des Entwerfers manifestiert sich immer weniger am ursprünglichen Gegenstand des Design, sondern mehr und mehr im Entwurf von Algorithmen und deren Implementierung, bis letztlich nur noch Handwerk (Fehlersuche, Effizienzsteigerung u.ä.) übrigbleibt – das ursprüngliche Entwurfskonzept bleibt davon in weiten Teilen unberührt.

Die hier entwickelten algorithmischen Entwurfsräume verfügen, wie viele vergleichbare Ansätze, über eine beeindruckende Leistungsfähigkeit bei der Erzeugung von Entwurfsvariationen, die weit über das hinausgeht, was ein menschlicher Entwerfer ohne diese Mittel zu leisten im Stande wäre, dank der Geschwindigkeit und der Automatisierungsmöglichkeiten der digitalen Informationsverarbeitung. Gleichzeitig bleibt das im Programmcode manifestierte Designwissen konserviert und kann jederzeit wieder abgerufen werden. Was jedoch die Erzeugung von Varietät anbelangt, sind einzelne algorithmische Entwurfsräume als Mittel kaum geeignet. Auch dies ist der digitalen Informationsverarbeitung geschuldet, in diesem Fall der Programmierung, die Entwurfskonzepte durch die Anforderungen formaler Sprachen in ein enges und unflexibles Netz spannt. Kurz gesagt, sind einzelne algorithmische Entwurfsräume hervorragende Mittel zur Variantenbildung, nicht aber zur Erzeugung entwerferischer Vielfalt.

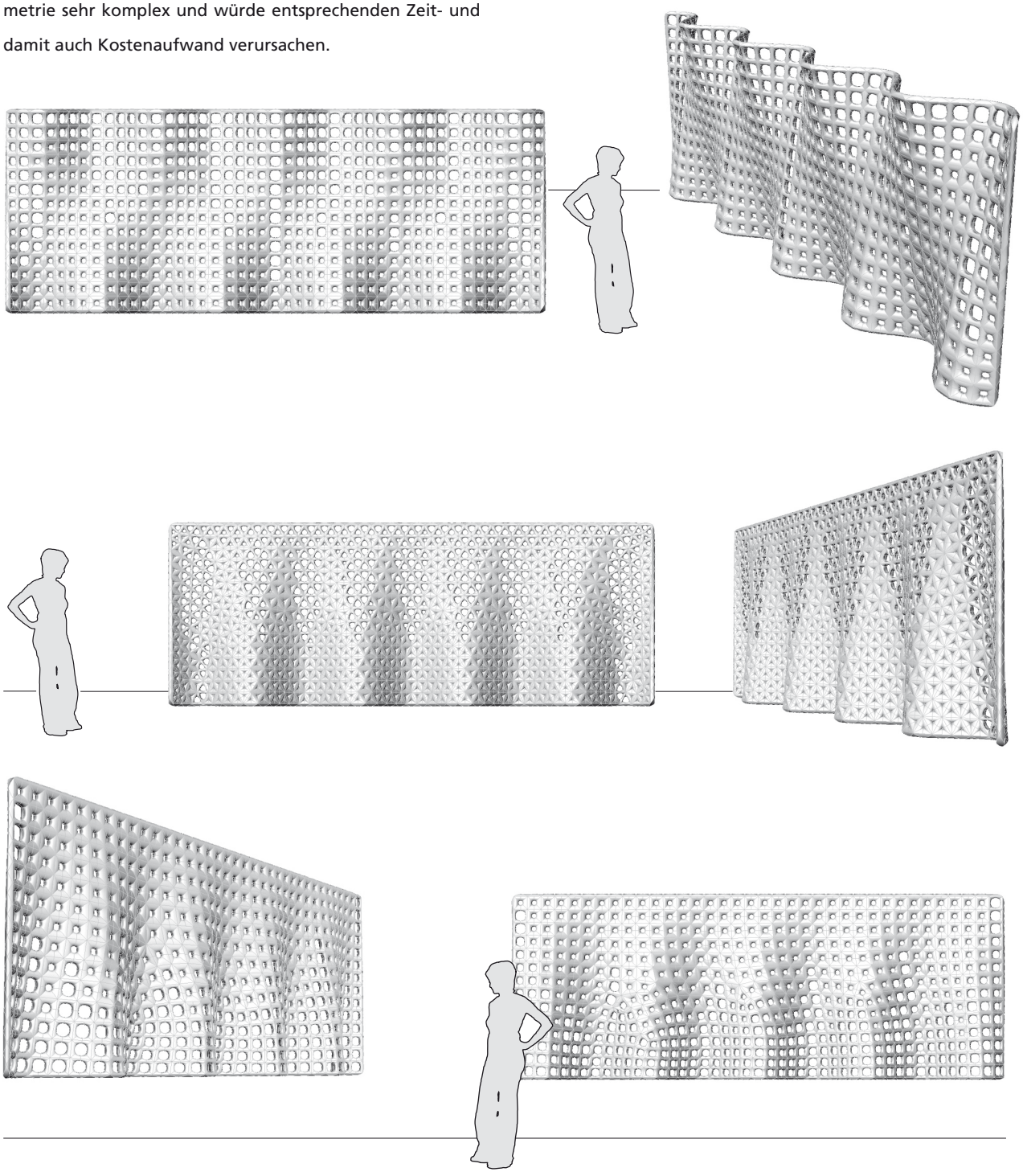
Um dieses Manko etwas auszugleichen, sind bereits zwei mögliche Wege skizziert worden: Der Weg der ständigen Pflege, Erweiterung und Rekombination algorithmischer Entwurfsräume, was letztlich zum Entwurf generativer Entwicklungsumgebungen führt und Entwerfen weiter auf die virtuelle, symbolische Metaebene des Algorithmendesigns bringt. Der zweite, dem angewandten gegenständlichen Design etwas nähere Weg, ist die Idee, eine Fläche als Designträger zu nutzen und damit ein generatives Entwurfskonzept in einen anderen Kontext zu transportieren. *De facto* wird angewandtes algorithmisches Entwerfen immer ein Spagat zwischen beiden Möglichkeiten sein. Die folgenden Seiten geben zum Abschluss des praktischen Teils noch einige kurze Beispiele für beide Strategien. Sie sind als Indizien, nicht aber als Belege für potentielle Entwicklungslinien zu sehen, die algorithmisches Entwerfen in Richtung breiterer Anwendungsfelder führen kann.

9.2 NURBS-Flächen als Designträger

Die Anwendung algorithmischer Werkzeuge, die für einen bestimmten Entwurfshintergrund konzipiert und entwickelt wurden, in einen neuen Kontext zu übertragen, bedeutet sich gedanklich von den Rahmenbedingungen zu lösen, von denen ausgehend die Werkzeuge entworfen wurden. Es ist dabei nicht mehr wichtig, für was die generativen Instrumente ursprünglich entwickelt wurden, sondern welche Themenstellungen mit ihnen sonst noch zu bearbeiten wären. Am naheliegendsten ist es hier, zunächst die festgelegte Größendimension zu verlassen, wie an den folgenden Beispielen [Abb. 01] zu sehen. Hier wurden Flächen modelliert, die als Raumteiler dienen sollen und die mit der entwickelten Methode aus Topologieoptimierung und parametrischem Design ausgearbeitet wurden. Auch bei diesen Beispielen wurden die Dichtewerte gestalterisch interpretiert, zweimal als stehende sowie einmal als hängende Struktur. Dabei haben die erzeugten Entwürfe keine wei-

tere statische Funktion zu erfüllen, als ihr eigenes Gewicht zu tragen. Der sichtbare Verlauf in den Strukturen, der den Kraftfluss wiedergibt, wird zweckentfremdet und zum Mittel einer leichten optischen Abtrennung und zur Lichtmodulation. Die Frage, die sich durch diesen Maßstabssprung neu stellt, ist die der Realisierung der Gebilde: Für eine Fertigung mit Hilfe eines 3D-Druck oder Lasersinterverfahrens sind sie jetzt zu groß, für eine Fräsanwendung ist die Geometrie sehr komplex und würde entsprechenden Zeit- und damit auch Kostenaufwand verursachen.

Abb. [01]) Maßstabssprung in der Anwendung der algorithmischen Entwurfsräume aus Topologieoptimierung und parametrischem Design: Drei Raumteiler, bei denen der visualisierte Kraftfluss zum Mittel der Lichtmodulation wird.



Um die Frage der Fertigung in größerem Maßstab zu lösen, wurde eine weitere Anfangsbedingung verworfen, nämlich die der Anwendung auf Freiformflächen. Auf bestimmten Flächentypen können auch problemlos Vierecksmaschen erzeugt werden, die alle die gleiche Größe haben. Das folgende Beispiel Abb. 02 zeigt einen zylindrischen Querschnitt, der durch sich wiederholende Bausteine aus *UHPC (Ultra High Performance Concrete)* umgesetzt werden soll. Durch die Reduktion auf bestimmte Flächentypen und gleiche Bauteile geht nun allerdings die geometrische Vielfalt und damit auch einiges von der besonderen Anmutungsqualität, die diese Entwurfsräume bieten, verloren. Deshalb ist in einem weiteren Beispiel, dessen Umsetzung aus Modulbausteinen aus *UHPC* gedacht ist, das selbe Flächennetz aus einer Rechtecksfläche mit drei verschiedenen Parametereinstellungen nacheinander interpretiert worden. Aus den daraus resultierenden, drei verschiedenen Bauteilen ist dann die gezeigte Wand Abb. 03, per Hand am *CAD*-System, zusammengesetzt worden.

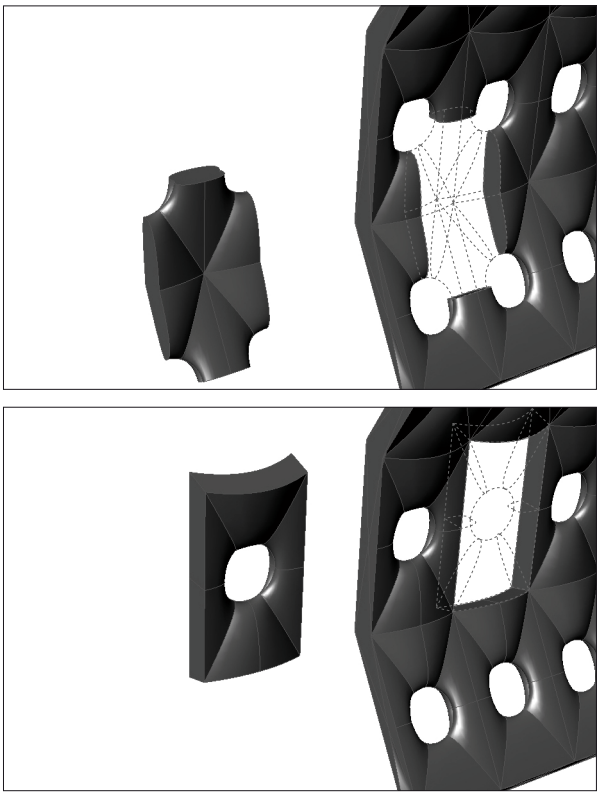


Abb. 02) Die Reduktion der Anwendung auf Flächen, die in exakt gleiche Viereckmaschen aufgeteilt werden können, schafft erweiterte Anwendungsmöglichkeiten innerhalb des algorithmischen Entwurfsraums: Tragstruktur, zusammengesetzt aus wenigen, sich wiederholenden Elementen.

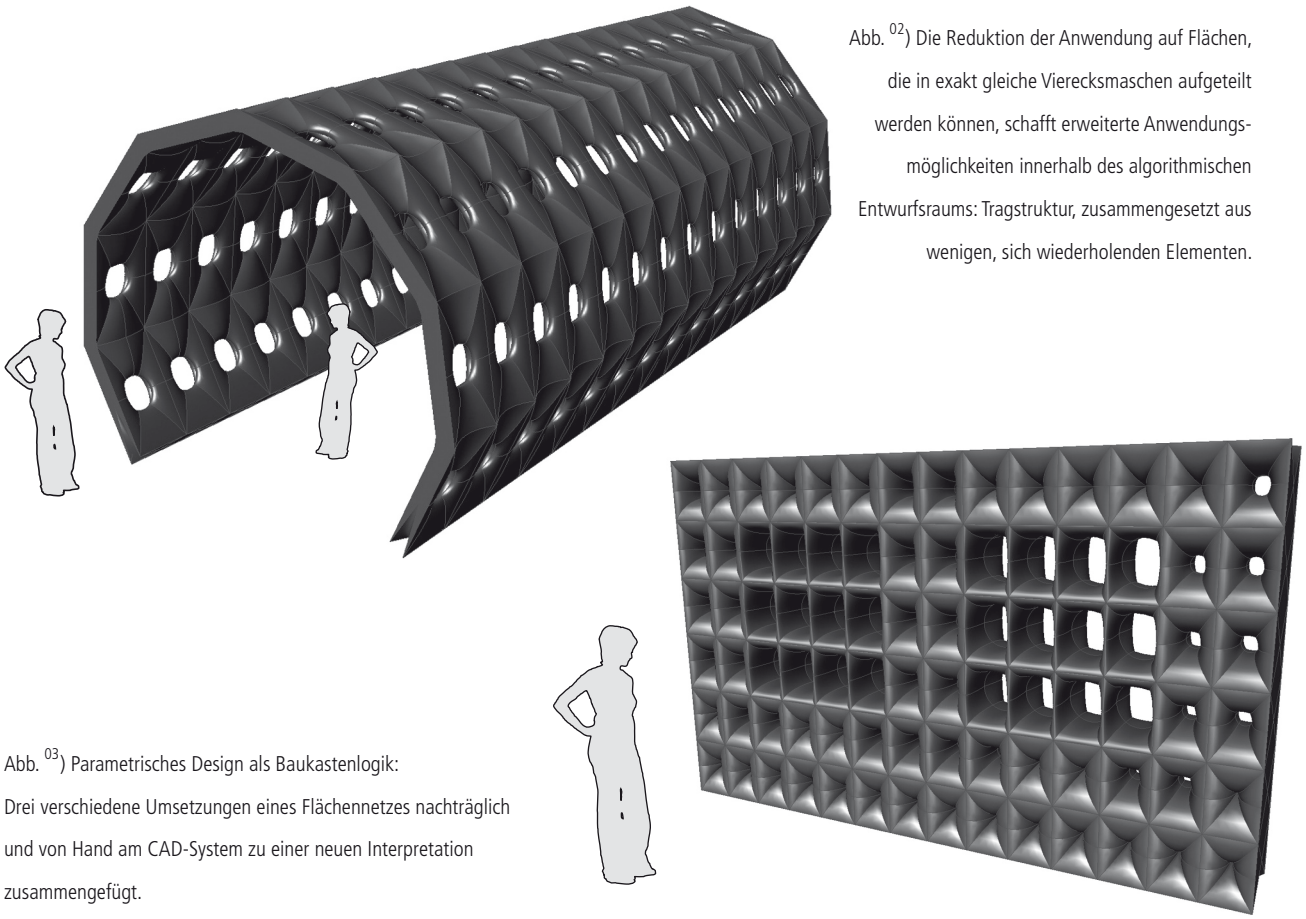


Abb. 03) Parametrisches Design als Baukastenlogik:
Drei verschiedene Umsetzungen eines Flächennetzes nachträglich und von Hand am CAD-System zu einer neuen Interpretation zusammengefügt.

Nun ist zwar wieder ein wenig von der geometrischen Vielfalt zurückgewonnen, die Zusammensetzung der drei Bauteile aber beliebig, ohne dass die Logik eines Kraftflusses in der so umgesetzten Wandfläche sichtbar würde. Auch hier ist eine einfache Lösung innerhalb des Entwurfsraums möglich: In der bisherigen Version der Konstruktionsmodule für die Interpretation der Topologieoptimierung werden die aus *ANSYS*® zurückgegebenen Dichtewerte direkt interpretiert. Durch eine entsprechende Rasterung dieses Wertes auf beispielsweise fünf, zehn oder 20 Stufen würde auch eine entsprechende Anzahl unterschiedlicher Bauteile entstehen, aber nicht mehr. So wird das Anwendungsfeld dieses Entwurfraums deutlich erweitert, es können sowohl verschiedene Standardbausteine für Betonwände entwickelt als auch die Wände durch ein modulares Baukastensystem einfach und unter Beibehaltung wirtschaftlicher Fertigungsperspektiven an verschiedene statische Bedingungen angepasst werden. Dieses System wird in Zusammenarbeit mit der Firma *gTecz*®, eine in Kassel angesiedelte Beratungsfirma, die auf Hochleistungsbeton spezialisiert ist, in einer verfeinerten Variaten umgesetzt und experimentiert.

Nach der Skalierung in größere Dimensionen, noch ein Sprung in kleinere. Hier wurden Flächen mit den Mitteln der Topologieoptimierung und des parametrischen Designs einmals als Schale Abb. 04 und einmal als Leuchtenkörper Abb. 05 interpretiert und über ein 3D-ABS-Druckverfahren hergestellt. Bei diesem Maßstab der Anwendung kommt zu der Realisierung freier Formen noch ein weiterer Gestaltungsaspekt – die Auflösung des Druckers, die, bei entsprechender Positionierung der Datei, in der Art von Höhenlinien sichtbar wird und eine feine Rhythmisierung der Oberfläche schafft, die sich bei größeren Flächen wieder verliert. Mit diesen beiden Entwürfen sind, abgesehen von der Größe, wieder alle ursprünglich definierten Rahmenbedingungen erfüllt. Bei beiden spielt jedoch die Idee des Kraftflusses keine Rolle mehr, sie wird zum ornamentalen bzw. lichtmodulierenden Element. Die Leuchte ist der Protoyp einer Serie aus verschiedenen Flächen, bei der Schale wird noch die Bedingung der direkten Fertigung verworfen: Sie dient als Urform zur Produktion einer Kleinserie aus feinem Porzellan. Die plastikhafte, etwas billig scheinende Anmutung des ABS-Drucks ist für diesen Entwurfszweck nicht geeignet.

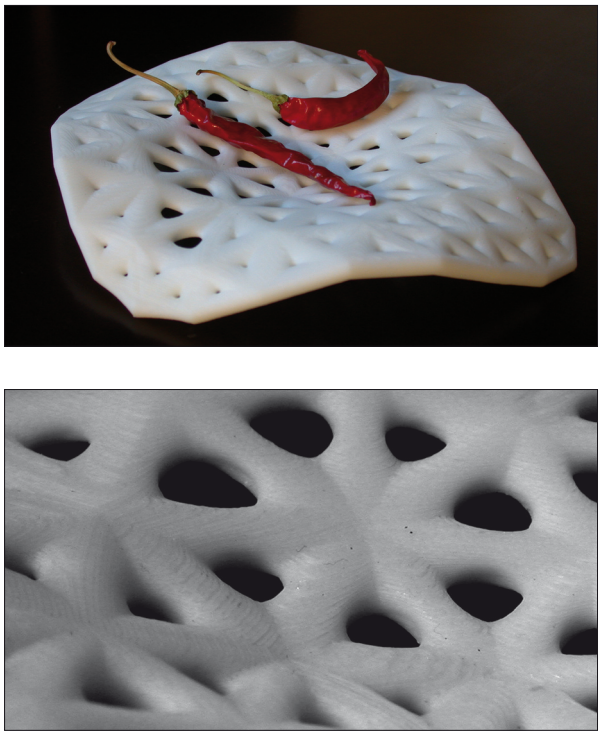


Abb. 04) Urform für eine Herstellung mit keramischen Materialien - eine kleine, im 3D-ABS-Druckverfahren hergestellte Schale.

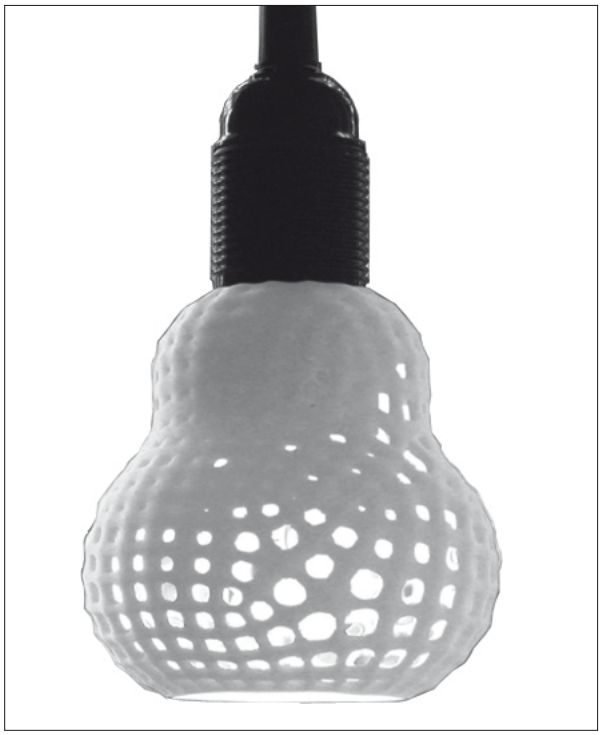
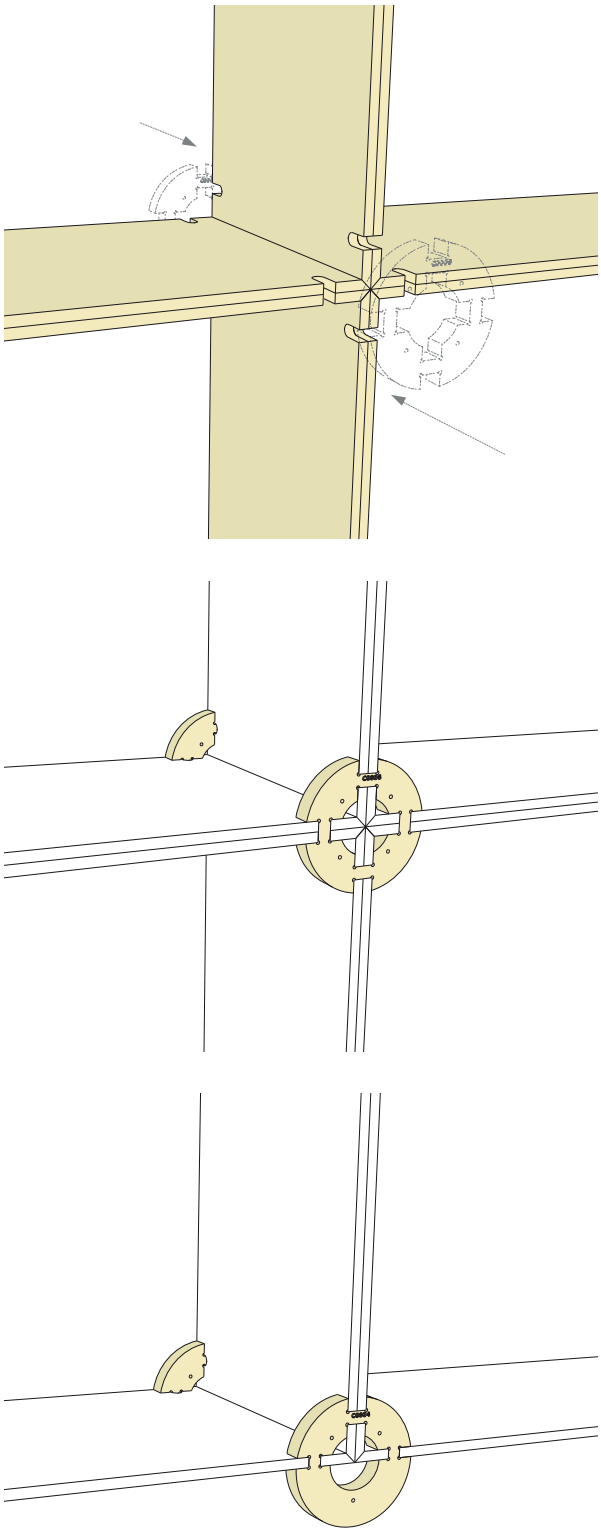


Abb. 05) Leuchtenkörper, ebenfalls im 3D-ABS-Druckverfahren hergestellt. (Krüger; Schein, 2006)

9.3 Erweiterung und Rekombination von Entwurfsräumen

Neben dem Ausloten der eingeschriebenen Möglichkeiten algorithmischer Entwurfsräume durch die Erprobung der enthaltenen Werkzeuge in verschiedenen Entwurfszusammenhängen, ist es natürlich jeder Zeit möglich, die instrumentelle Basis dieser Entwurfsräume zu erweitern, indem neue Designalgorithmen entworfen und ergänzt werden. Das hierfür gezeigte Beispiel ist eine Zweckentfremdung der Flächenaufteilungssystematik, die für die evolutionäre Optimierung verwendet wurde: Für diese Systematik wurde ein zusätzliches Konstruktionsmodul entwickelt, das aber nur dann anwendbar ist, wenn die Ursprungsflächen bereits planar sind und mit dem sich Regale erzeugen lassen. Diese Regale basieren auf einem einfachen Platten-Verbinder-Steckprinzip, entsprechend der drei verschiedenen Wertigkeit der Knoten im Viereckesnetz sind auch drei unterschiedlich ausgeprägte Verbinderteile nötig Abb. 06. Die wesentlichen Konstruktionsparameter lassen sich, wie bei allen anderen Modulen auch, verändern Abb. 07. Ebenso werden alle zur Fertigung notwendigen Daten automatisch erzeugt.



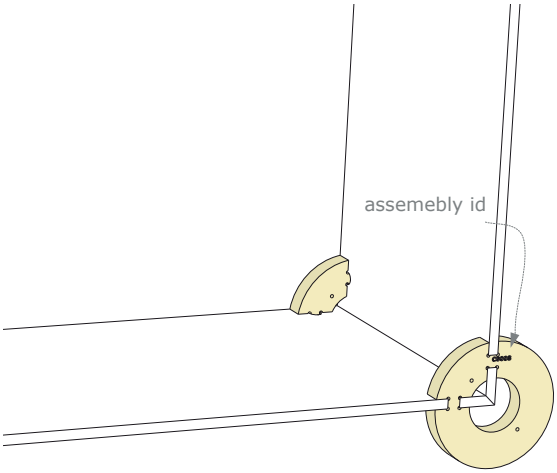


Abb. 06) Modul zum Erzeugen von Regalen aus planaren Flächennetzen: Steckprinzip und die drei verschiedenen Verbinderelemente.

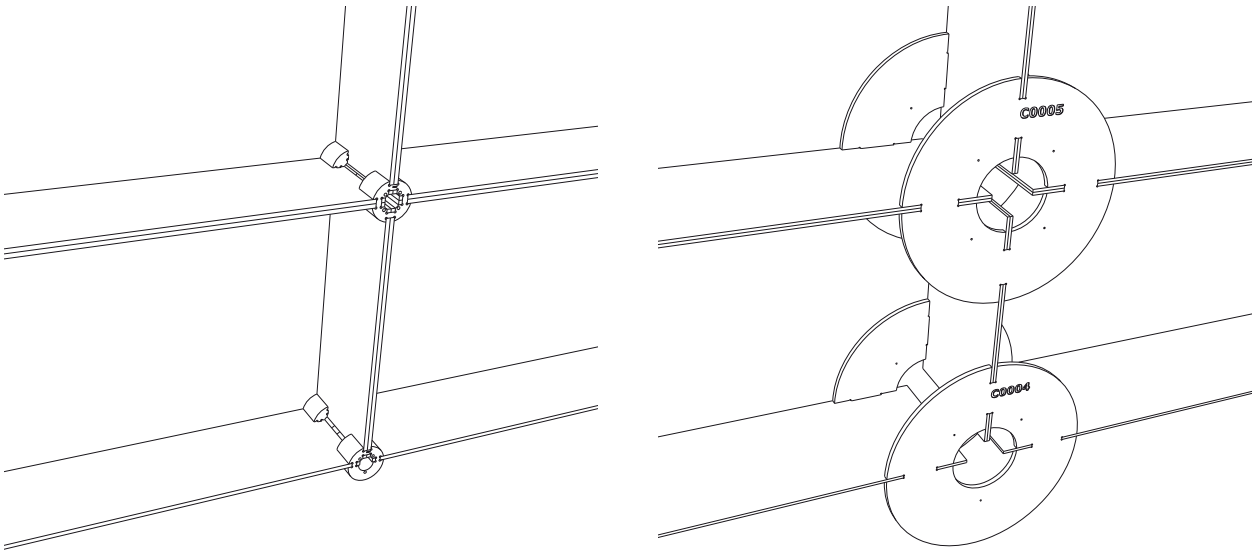


Abb. [07]) Zwei weitere Varianten der Umsetzung.

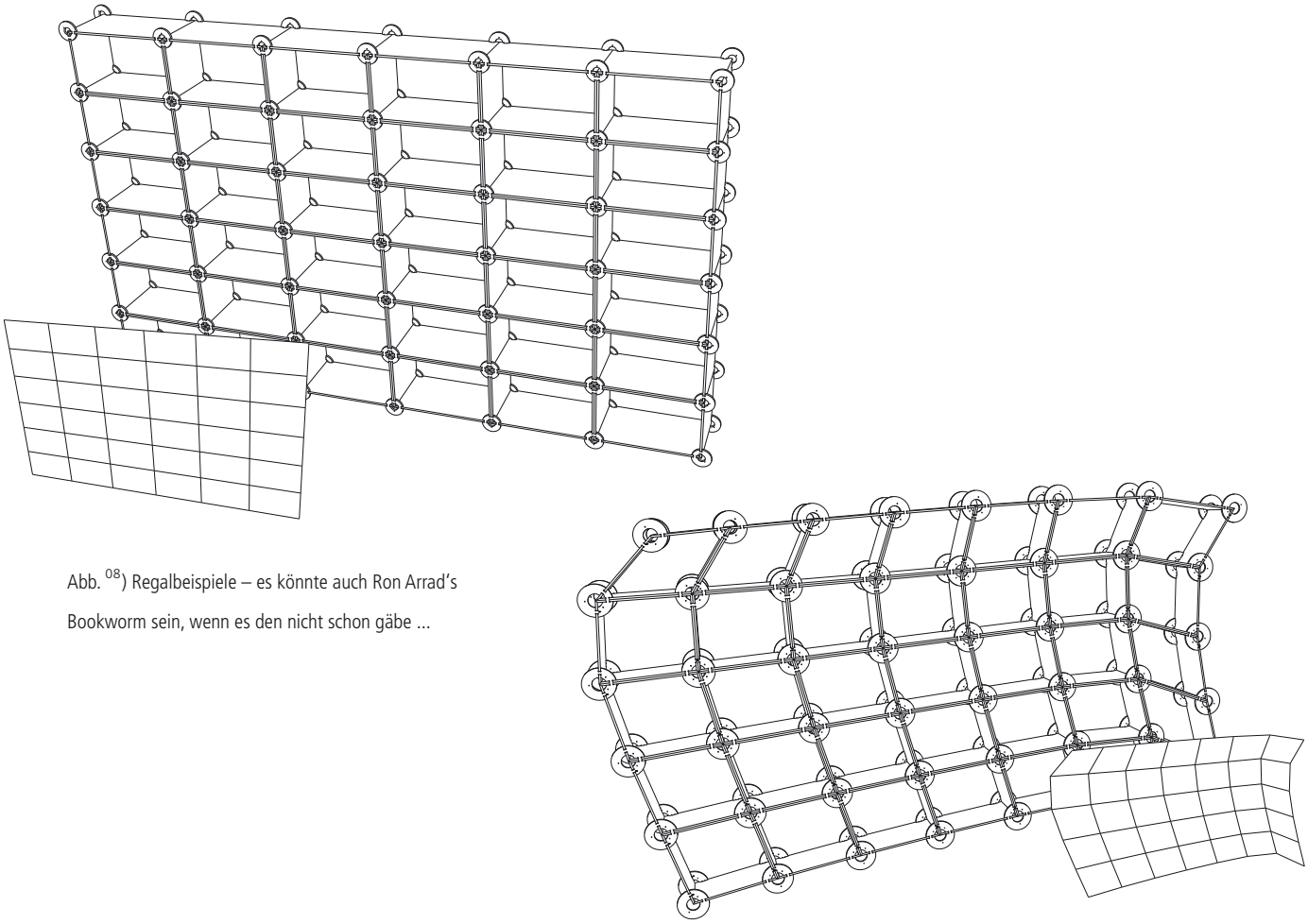


Abb. [08]) Regalbeispiele – es könnte auch Ron Arrad's Bookworm sein, wenn es den nicht schon gäbe ...

Das folgende Beispiel beschließt den praktischen Teil der Arbeit. Es ist das aus Autorensicht wichtigste Experiment, das im Rahmen dieser Arbeit durchgeführt wurde. Äußerlich, von der Seite seiner Beschreibung, der Methodik und Ergebnisse betrachtet, unterschiedet es sich wenig von den vorangegangenen Beispielen. Trotzdem markiert es aber einen Punkt, in dem das intellektuell geprägte Verständnis in der Handhabung generativer Methoden und die Beherrschung der instrumentellen Grundlagen mit einem Gefühl zusammengefallen sind, wie ein selbstverständlicher, entwerferischer Umgang mit solchen generativen Methoden beschaffen sein kann: Als ein ständiger, unvermeidlicher Spagat zwischen der virtuellen Welt des problemorientierten Algorithmendesigns einerseits und der direkten und unmittelbaren Überprüfung und Anwendung dieser Entwurfsebene in der Welt des physischen und lösungsorientierten Designs auf der anderen Seite.

Das Experiment adressiert nochmals das Problem der planaren Flächenelemente, diesmal allerdings mit einer anderen Herangehensweisen. Dazu wurde zunächst eine Aufteilungssystematik entwickelt, die von vorneweg planare Viereckmaschen erzeugt, orientiert an dem, was *Schober (2002)* [01] *„Den Dreh mit den Geraden“* nennt.

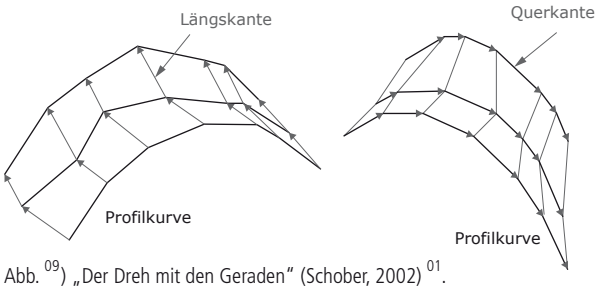


Abb. [09]) „Der Dreh mit den Geraden“ (Schober, 2002) [01].

Diese Methode basiert auf dem geometrischen Prinzip, dass zwei parallele Vektoren im Raum immer eine ebene Viereckfläche aufspannen, deren Kanten durch die Anfangs- und Endpunkte der Vektoren definiert werden. Wird dieses einfache Prinzip bereits bei der Erzeugung einer Ursprungsfläche berücksichtigt, kann damit bereits eine relativ große Vielfalt an Formen realisiert werden. So kann beispielsweise eine beliebige Translationsfläche, die dadurch entsteht, dass eine Erzeugende normal entlang einer Leitlinie geführt wird, durch dieses Prinzip nachgebildet werden [Abb. 09]. Möglichkeiten der Translation und Skalierung erweitern dieses Prinzip, wie *Schober* und *Schlaich* bereits an einigen beeindruckenden Projekten gezeigt haben.

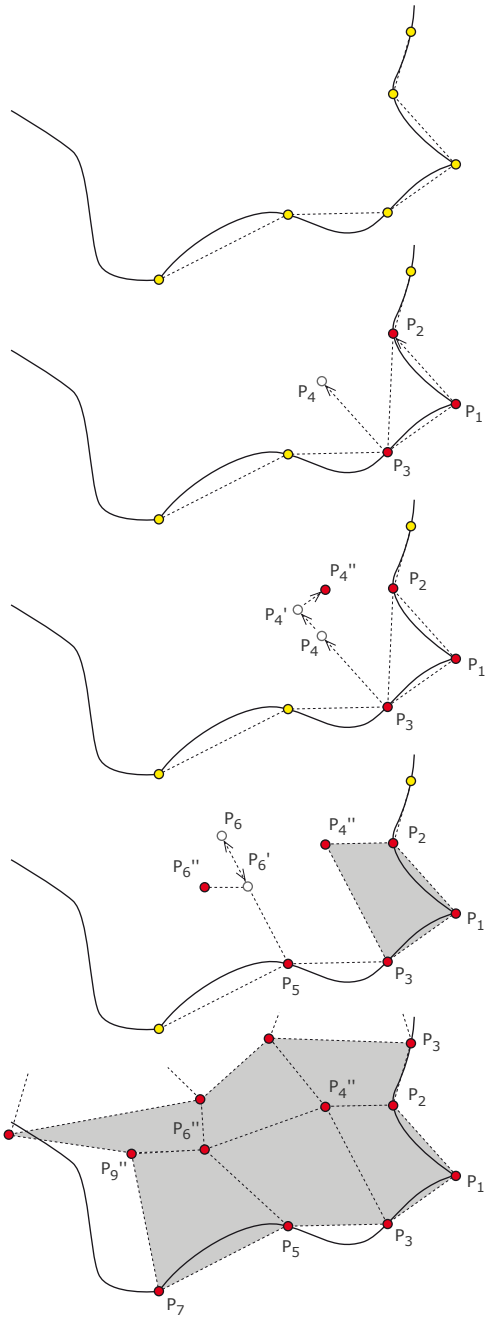


Abb. [10]) Elementierungsverfahren als Erweiterung des ‚Geraden-Drehs‘.

Ausgehend von dieser Methode, die einer vorgeschalteten Rationalisierung eines Flächenentwurfs entspricht, wurde ein Elementierungsverfahren entwickelt, das sich im nachhinein auf beliebige Flächen anwenden läßt: Zunächst wird eine Längs- und eine Querkante einer *NURBS*-Fläche in eine bestimmte Anzahl zufälliger Segmente unterteilt Abb. 10 / 1. Aus den so erzeugten Punkten, die bis jetzt alle genau auf der Ursprungsfläche liegen, werden nun die planaren Viereckmaschen entwickelt. Als nächster Schritt wird eine erste Ebene gebildet, die durch einen Eckpunkt der Ursprungsfläche (*P1*) sowie den jeweils nächsten Punkten auf der Längs- sowie der Querkante (*P2* und *P3*) definiert ist Abb. 10 / 2. Der vierte Punkt (*P4*) wird nach dem Schober-Prinzip erzeugt, durch Verschiebung des Punktes auf der Querkante (*P3*) um den Vektor entlang der Längskante Abb. 10 / 3. Ist der Punkt plaziert, wird er jetzt noch innerhalb eines definierten Wertebereichs zufällig verschoben, und zwar nochmals entlang des Längsvektors (*P4'*) und entlang des Quervektors (*P4''*) Abb. 10 / 4 – die erste Viereckmasche ist bestimmt. Die zweite, durch den vierten Punkt definierte Kante, ist jetzt der neue Längsvektor für die anschließende Viereckmasche, der anschließende Punkt auf der Querkante der Ursprungsfläche (*P5*) bildet zusammen mit dem vorigen Punkt (*P3*) den neuen Quervektor Abb. 10 / 5. Der vierte Punkte für diese neue Masche wird erzeugt (*P6*), wieder verschoben (*P6'*) (*P6''*) usw. Nach diesem Vorgehen entsteht das gesamte Flächennetz.

Als Option ist, wie bei den anderen Aufteilungsvarianten, eine Verschiebung der Knotenpunkte entlang ihrer Normalen zur Fläche enthalten – wobei sich dies Möglichkeit hier nur auf Netzknoten bezieht, die an den Rändern der Fläche liegen: Ändert sich die Postion eines solchen Knotens, ändert sich auch der entsprechende Vektor und damit die gesamte folgende Reihe von Maschen. Die Normalenverschiebung ist in den Genotyp des genetischen Algorithmus eingeschrieben, ebenso die uv-Parameter der Punkte auf den beiden Flächenkanten und die jeweiligen Verschiebungen des vierten Punkts jeder Masche – diese Parameter können demnach auch durch die genetischen Operatoren verändert werden.

Neben den üblichen, schon beschriebenen Fitnesstests, die dazu dienen, die Homogenität eines Flächennetzes zu kontrollieren, wurde noch ein weiterer Test implementiert, der prüft, wie weit die Netzknoten in Summe von der Ursprungsfläche entfernt sind. Dieser Test sorgt also im Ablauf des Evolutionsprozesses dafür, dass sich das Netz aus bereits planaren Viereckmaschen so weit als möglich der Form der Ursprungsfläche angleicht. Um mit diesem Verfahren gute Ergebnisse zu bekommen, musste der genetische Algorithmus erneut erweitert werden. Normalerweise werden die Fitnesstests immer auf alle Elemente eines Flächennetzes angewandt – damit waren hier aber keine brauchbaren Resultate zu erzielen. Deshalb wurde die Methode der sequentiellen Optimierung implementiert.

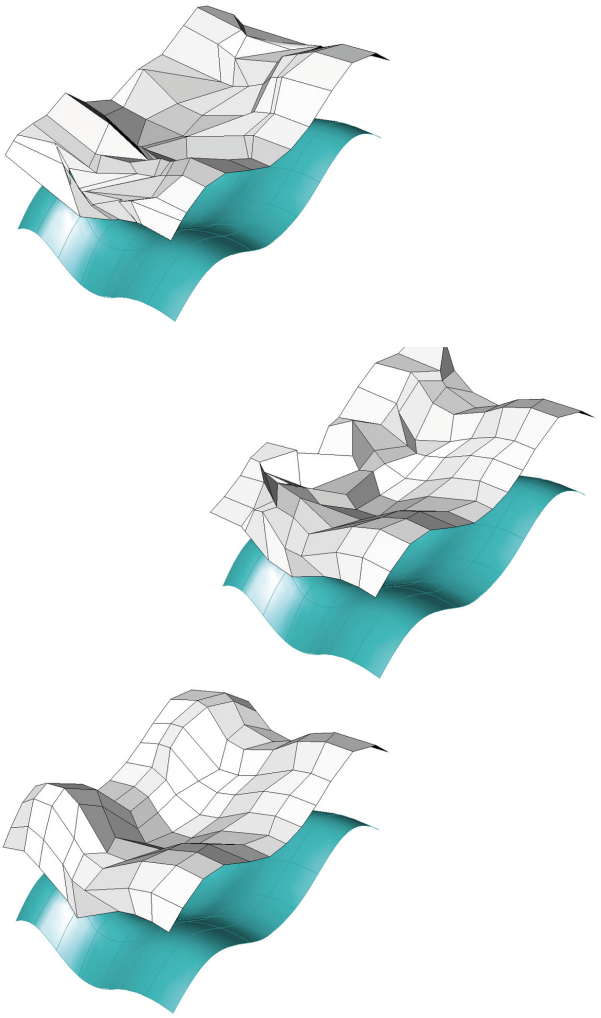


Abb. [11]) Sequentielle Optimierung durch Anwendung der Fitnesstests auf immer größere Teilbereiche des Flächennetzes.

Dies bedeutet, dass die Fitnesstests zunächst nur auf eine Reihe von Flächenmaschen angewendet werden, nach einer bestimmten Anzahl Generationen wird die zweite Reihe in die Bewertung miteinbezogen, dann die dritte usw. bis schließlich in der letzten Sequenz von Durchläufen alle Maschen evaluiert werden Abb. 11. Mit Hilfe dieses Kniffs können nun Flächennetze erzeugt werden, die sich weitgehend der Form der Ursprungsfläche annäheren, gleichzeitig homogen anmuten und aus völlig planaren Viereckmaschen gebildet werden.

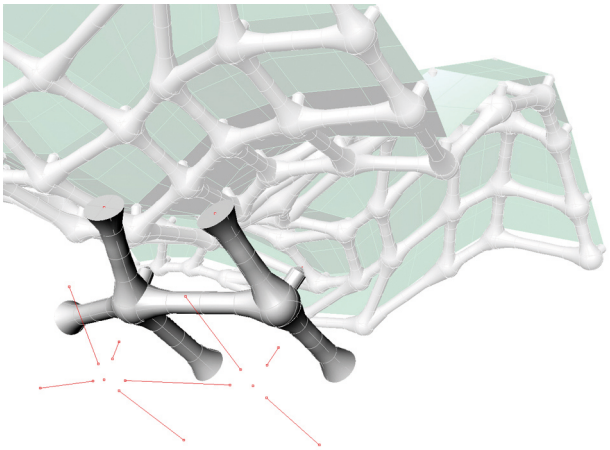
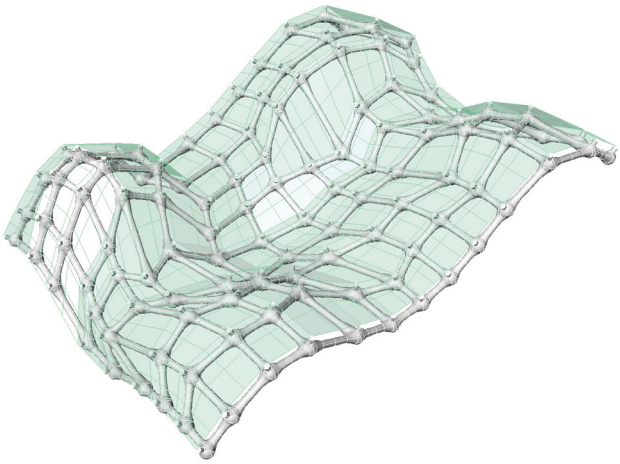


Abb. 12) ‚Knochenmodul' aus dem Hockerexperiment, angewandt auf evolutionär optimierte Netze.

Der letzte Schritt in diesem Prozess war die Kombination von Programmmodulen aus beiden Entwurfsexperimenten. Ohne das dies zuvor geplant gewesen wäre, wurde gesehen, dass das Konstruktionsmodul, das für die Erzeugung der Hocker aus knochenähnlichen Strukturen entworfen wurde, bereits parametrisch aufbereitet, alle relevanten geometrischen Daten zur Erzeugung einer Standard-Knoten-Stab-Konstruktion mit Glashalter enthält Abb. 12. Um das Modul nutzen zu können, musste eine zusätzliche Schnittstelle programmiert werden, die die Ausgabedaten des genetischen Algorithmus in eine Form bringt, die von diesem Modul auch gelesen werden kann. Die Weiterentwicklung des ‚Knochenmoduls' selbst in eine Form, dass es detaillierte Knoten, Stäbe und Glashalter ausgibt, inklusive aller Bohrungen und Verbinder für die Anbindung der Glasflächen, wäre eine Fleißaufgabe, die in grob einer Woche erledigt sein dürfte.



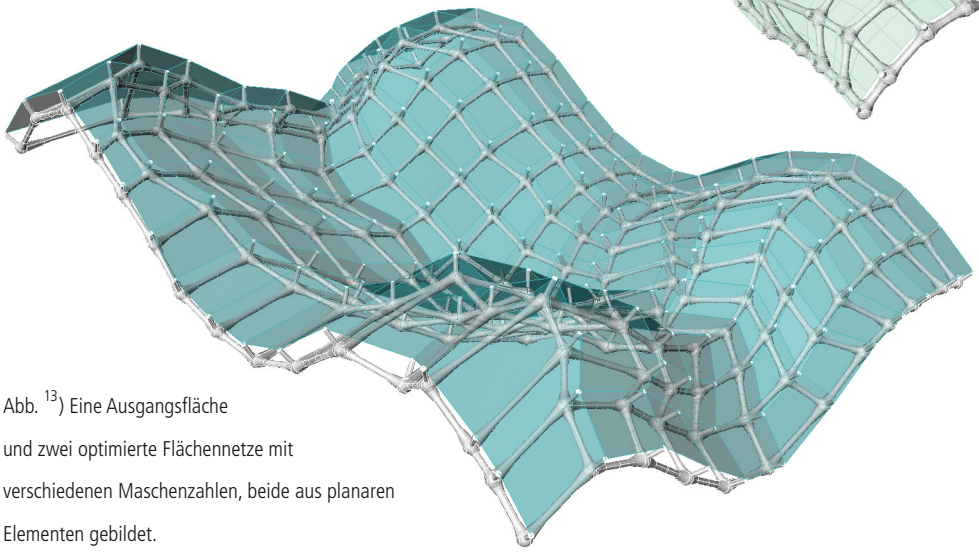
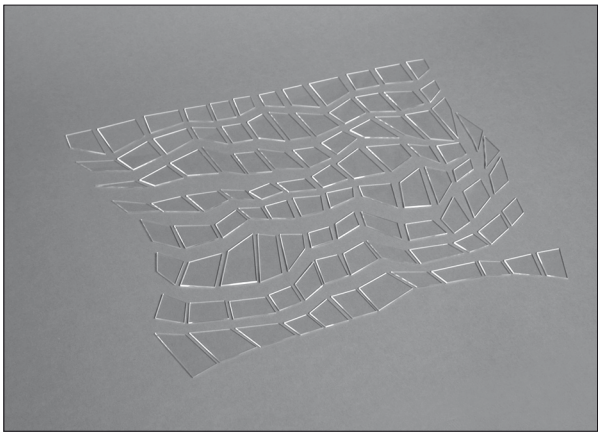
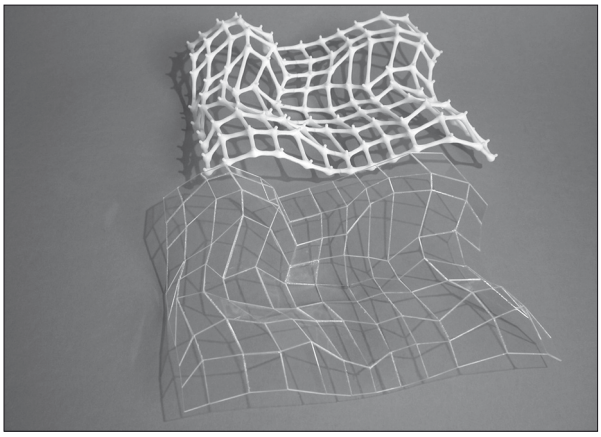
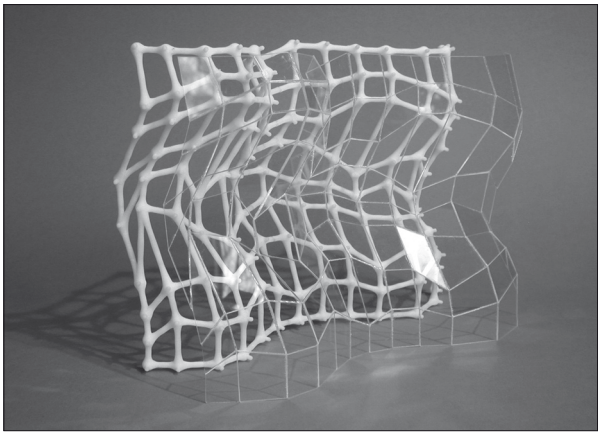


Abb. 13) Eine Ausgangsfläche und zwei optimierte Flächennetze mit verschiedenen Maschenzahlen, beide aus planaren Elementen gebildet.







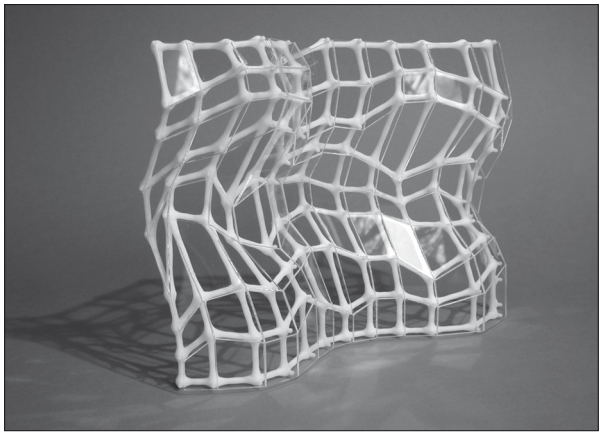


Abb. [14]) Modellhafte Umsetzung der rechten, gröber aufgelösten Fläche aus Abbildung 13: Scheiben, aus Plexiglas gelasert, vereinfachte Tragstruktur, im 3D-ABS-Druckverfahren hergestellt.

10. Kritik und Nachtrag

10.1 Kritik und Ausblick

In einer persönlichen Nachschau der Arbeit wurde klar, dass ihre Schwächen aber vielleicht auch ihre Stärken darin begründet sind, dass sie – im Grunde bis zu den letzten Zeilen – zwischen drei Polen hin und her pendelt. Der erste Pol ist das intellektuelle, rationale Durchdringen des Themenkomplexes des generativen und algorithmischen Entwerfens – die Theorie. Und der zweite der Anspruch, das so erworbene Verständnis auch im eigenen Handeln erfahrbar zu machen und nachzuvollziehen – die Praxis. Das sich diese beiden Bereiche nicht unberührt lassen können und sollen ist selbstverständlich – *„Theorie ohne Praxis ist leer, Praxis ohne Theorie ist blind" (Maser, 2001)* [01]. Der dritte Pol schließlich ist die eigene entwerferische Haltung, die sich lange vor der Beschäftigung mit dem algorithmischen Design in der Praxis herausgebildet hat und eher an einem traditionellen, modernistisch geprägten Designverständnis orientiert war.

Entsprechend dieser Pole zeigt die Arbeit auch unterschiedliche Erkenntnisniveaus, die nicht unbedingt in der Reihenfolge der Kapitel ausgedrückt sind: Einige Gedanken, die in der Theorie als fruchtbar erschienen sind, wurden durch die Praxis nicht bestätigt, einiges, was die praktische Erfahrung lehrte, hat zu einer Vertiefung der Theorie geführt. Und schließlich konnten auch einige der Ansprüche, die theoretisch begriffen und formuliert wurden nicht im Entwurfsexperiment realisiert und ausgedrückt werden. Es ist eine Sache, sich klar zu sein, welche Denkstrukturen zum algorithmischen Entwerfen nötig sind, die andere sich tatsächlich und selbstverständlich darin wiederzufinden – und an der eigenen Lebenszeit zu erfahren, wie aufwendig die instrumentelle Beherrschung komplexer generativer Systeme ist. Die so entstandene Inhomogenität in der Arbeit ist aber in gewisser Weise auch ein Ausdruck der gesamten algorithmischen Entwurfskultur, die gerade entsteht – als Thema ist es, wie im Englischen so treffend formuliert, ein *moving target*, und noch dazu ein sehr Schnelles. Als ich mit meiner Diplomarbeit in 2001 die ersten Schritte in dieses Gebiet unternommen habe, existierte der größte Teil der Beispiele nicht, die sich heute, nur sechs Jahre später, in dieser Arbeit wiederfinden, genausowenig, wie viele Gedanken, die sich heute selbstverständlich in Bücherregalen und im Netz versammeln.

Die drei Arbeitshypothesen zu den algorithmischen Entwurfsräumen haben sich als gute Instrumente erwiesen, Inhalte zu bündeln und gleichzeitig Erkenntnisse aus den unterschiedlichen Themengebieten zu hinterfragen. Was die erste Hypothese anbelangt, nach der Entwurfsthemen so gedeutet, *Constraints* so definiert werden, dass sie den algorithmischen Methoden zugänglich werden, hat sich voll und ganz bestätigt – nicht nur in der eigenen Erfahrung sondern auch in der Auseinandersetzung mit meinen Fachkollegen. Natürlich ist es so, dass Designwerkzeuge und -methoden, die man selbst unter großem Aufwand entworfen und entwickelt hat, auch eingesetzt sein wollen. Da werden dann schon einmal Themen angegangen, für die andere Entwurfsstrategien sicher weitaus besser geeignet wären. Im Grunde gilt dies aber für jede praktisch orientierte, entwerferische Arbeit, bei der gewissen Werkzeuge und Methoden eingesetzt werden. Nur wird dies dort nicht so deutlich, weil mehr der praktische denn der erkenntnistheoretische Aspekt des Entwerfens im Vordergrund steht und die verwendeten Methoden eher selten externalisiert und explizit gemacht werden. Und selbstverständlich spielt auch die ungeheure Faszination, mit der generative Prozesse immer wieder scheinbar mühelos komplexe Geometrien zaubern, eine nicht unbedeutende Rolle, wenn Entwurfsthemen in die entsprechende Richtung gedeutet werden. Wenn man sich dessen bewußt ist, verringert sich auch die Gefahr der Verführung, algorithmisches Entwerfen als Selbstzweck zu betreiben.

Die zweite Hypothese, nach der einzelne algorithmische Entwurfsräume – in der Anschauung – nur eher kleine Lösungsräume abbilden können, ist zu revidieren. Zwar ist es richtig, dass hier nicht mit *Constraints* umgegangen werden kann, wie aus anderen Entwurfszusammenhängen gewohnt: Definieren, hinterfragen, wieder vergessen, neue

Rahmenbedingungen setzen, andere, bereits gesetzte ignorieren usw. Es muss von Anfang an sehr sorgfältig überlegt werden, welche Bedinungen überhaupt verwendet werden können, wie sie in der algorithmischen Konzeption gefasst und in Beziehung gesetzt werden können, um daraus schrittweise ein generatives Programm zu entwickeln. Die Lösungsräume, die so definiert werden, erscheinen nur dann klein, wenn man als Maßstab Varietät nimmt, etwa der Art, wie sie ein geübter Entwerfer mit einem Zeichenstift erzeugen kann. Kurzum, zur Varietätserzeugung in diesem Sinne halte ich algorithmische Methoden eher für ungeeignet. Anders sieht es aus, wenn man beginnt zu untersuchen, wie sich kleine Differenzierungen von Mustern erzeugen und als form- und funktiongebende Elemente nutzen lassen. Dann eröffnet sich auf einmal ein ganz neuer und umfassender Gestaltungshorizont, der sich aus der Suche nach Varietät so nicht erschließen läßt.

Was die dritte Hypothese betrifft, nach der sich algorithmische Entwurfsräume auf der Ebene der Algorithmen zu neuen und umfassenderen Lösungsräumen zusammen führen lassen, muss hier differenziert zwischen der anwendungsbezogenen und der allgemeingültigen Ebene unterschieden werden. Zunächst sind sowohl gedankliche Vorformatierungen als auch instrumentelle Hürden zu überwinden. Der wesentliche Aspekt dabei ist tatsächlich, dass man Algorithmen abstrakt betrachtet und sie auf der Ebene informationsverarbeitender Muster sieht, also als generische Werkzeuge und nicht als spezifisch, für einen bestimmten Entwurfszweck konzipiert. Genau das aber fällt Entwerferherzen nicht leicht, die eher an Lösungen interessiert sind als an der grundsätzlichen Bewältigung allgemeiner Probleme. Diese Trennlinie macht es ebenfalls schwierig, informationsverarbeitende Muster zu entwickeln, sie als solche zu betrachten und in neuen Bedeutungszusammenhängen zu erproben – auch das eine Erfahrung aus den praktischen Experimenten und der Zusammenarbeit mit Fachkollegen. ‚Jeder für sich und für jedes Projekt' könnte das Motto sein, nachdem Programme entworfen und implementiert werden. Im Ergebnis entsteht ein Salat von Programmcode, der schwer entwirrbar ist und das gemeinsame Arbeiten an Projekten nicht unbedingt fördert, auf der Bedeutungsebene aber auch der Ebene konkreter Schnittstellen zur Datenübergabe. Realistischerweise muss man wohl eingestehen, dass dieses Wechseln zwischen theoretischer Problemlösung und praktischer Anwendbarkeit, zwischen dem Generischen und dem Generativen, nicht mehr als Einzelkämpfer zu bewältigen sein wird – es braucht hier das Zusammenspiel verschiedener disziplinärer Haltungen mit ihren unterschiedlichen Problemverständnissen und Arbeitsweisen. Im konkreten Projekt immer wieder zu entscheiden, welcher Teil der Programme nun als allgemeingültig zu betrachten ist und welcher als projektspezifisch, erfordert große intellektuelle Disziplin, die einer zusätzlichen Überprüfung aus anderer Sichtweise bedarf. Und schließlich ist es mit einer Unterscheidung entlang dieser Trennlinien nicht getan – das Designwissen, das in Programmcode explizit gemacht wurde, ist auch so zu dokumentieren und in Programmbibliotheken abzulegen, dass es wieder auffindbar ist und so erst erneut zur Anwendung kommen kann.

Die Entwicklung und Pflege solcher Datenbanken ist wohl der Schlüssel für die Weiterentwicklung algorithmischer und generativer Entwurfsmethoden – und eröffnet neue Perspektiven für das Design. Einmal sind es Entwerfer bisher wenig gewohnt oder vielleicht auch nicht in der Lage, ihre ‚Betriebsgeheimnisse', also die Werkzeuge und Methoden, die ihre entwerferische Tätigkeit konstituieren, anderen direkt zugänglich zu machen. Ganz anders ist dies in der Informatikergilde mit ihrer Tradition des lebhaften und kostenlosen Wissensaustausches, manifestiert in einer Vielzahl von *blogs*, *newsgroups* und Datenbanken mit frei verfügbarem Programmcode. Hier zeichnet sich eine fruchtbare Verbindung zweier sehr unterschiedlicher Arbeitsweisen ab. Und die zweite Perspektive ist die Herausforderung des Credos, dass sich Wissen im Design nicht kumulativ verhält. Sicher bietet jede neue Problemstellung neue Herausforderungen und verlangt die Aneignung neuen Wissens. Genauso sicher spielen dabei aber alle vorher gelösten Probleme eine Rolle: Jedes Projekt erweitert das Designwissen und damit das Handlungsspektrum des Entwerfenden und wirkt sich so auf alle künftigen Projekte aus. Wird dieses Wissen in

Bibliotheken gesammelt, erhält man eine Vielzahl von Kategorien an algorithmischen Problemlösern, auf generischer wie auch generativer Basis. Dabei sind diese Problemlöser nicht statisch, sondern können ständig geändert, erweitert und untereinander kombiniert werden.

Als eher persönliches Resümee und gleichzeitig wesentliche Erkenntnisse sollen hier noch einige weitere Dinge kurz angesprochen werden. Als Wesen des Entwerfens betrachte ich den Blick in die Zukunft. Informiert und detailliert wird dieser Blick durch unterschiedliche Sprachen auf verschiedenen Modellierebenen. Gleichzeitig wird wieder das Medium der Sprache benutzt, um die formulierten Modelle zu überprüfen – Sprache ist also in der Lage, sich selbst zu reflektieren. Computer führen Programme aus, Programme bestehen aus Code und Code ist Sprache. Demnach können sich auch Programme, innerhalb der vom Programmierer definierten Grenzen selbst reflektieren und gegebenenfalls ihre Ausführung ändern. An dieser Stelle verlieren die üblichen Werkzeugmetaphern, die der Computernutzung unterlegt sind, ihre Gültigkeit. Der Computer wird zur nicht-trivialen Maschine und führt damit zu Entwurfsmöglichkeiten, die noch in abstrakten Kategorien fassbar, aber nicht mehr in ihrer detaillierten Ausprägung kontrollierbar sind. Ein zweiter wichtiger Aspekt scheint mir zu sein, dass das algorithmische Entwerfen das Potential bietet, ein ökologisches Designverständnis zu befördern – im systemischen Sinne. Die Sozialisierung zeitgenössischer Entwerfer findet in einem kulturellen Kontext statt, der geprägt ist durch philosophische Einflüsse, die sich mit Schlagworten wie positivistisch, rational, ökonomisch oder mechanistisch charakterisieren lassen – was die Grundlagen für die Entwicklung unserer Wissenschaften und dem Verständnis der materiellen Welt sind. Trotz eines umfassenden Wissens über die Komplexität von Systemen aller Art, zeigt sich dem entsprechend eine gestaltete Welt, die in vielerlei Hinsicht eher auf diskreter Abgrenzung basiert – in den Objekten wie den Köpfen und damit auch die Limitierungen, die diese Denkweisen mit sich bringen. Es war für mich verblüffend zu erkennen, dass gerade die harte, auf schwarz und weiß basierende Sprache der Algorithmen hier Potentiale für neue Designphilosophien öffnet, auf zwei Weisen. Zum einem macht die formale Logik eben genau die Grenzen diskreter Betrachtungen klar – was wir in Computermodellen beschreiben ist eben nicht die Realität, nicht das künftige Produkt, sondern über Geometrie simuliertes Material, eine Abstraktion der geplanten Realität. Genau diese Simulation von Materie über Geometrie aber kann mit Computerhilfe weitaus komplexer und differenzierter entwickelt werden als bisher. Generatives und algorithmisches Entwerfen ist also die Basis einer zutiefst materiellen Entwurfshaltung, inspiriert von den Möglichkeiten der digitalen Datenverarbeitung. Nicht was die Dinge bedeuten ist hier die Frage, sondern was sie tun und wie sie sich zueinander verhalten, wie dies *Reiser* und *Umemoto (2006)* [02] sinngemäß formulieren. Möglicherweise führt also diese Art des Entwerfens zu einer Transzendenz des Konkreten durch das Abstrakte, zu einem besseren intuitiven Verständnis von Differenzierung und Verflechtung durch das Rationale.

Um dies im praktischen Sinn zu denken, stehen für mich als nächste Arbeitschritte an, zunächst die vielen tausend Zeilen Programmcode, die in den letzten Jahren entstanden sind zu entflechten, sie soweit als möglich von den Kontexten, mit denen sie verwoben sind zu entblättern und sie zu einem algorithmischen Entwurfswerkzeug zu formen – um neue gedankliche und tatsächliche Schnittstellen sowie Entwurfsanlässe zu schaffen. Dann geht es vor allem daran, die Vielzahl von Projekten, die im Kopf schweben, im praktischen Entwurf weiter auszuloten und (nach Möglichkeit) auch umzusetzen. Kurzum, die ganzen Mühen, die mit dieser Arbeit verbunden waren, haben sich für mich letztendlich doch gelohnt, sie haben den Enthusiasmus für das algorithmische Design nicht geschmälert. Ganz im Gegenteil – es ist zum selbstverständlichen Bestandteil meiner ganz persönlichen Designphilosophie geworden.

10.2 Kurzer Nachtrag

Trotz einer gewissen Zufriedenheit bleibt bei der Beschäftigung mit der algorithmischen Entwurfskultur doch etwas Merkwürdiges haften. Es ist ein leises Gefühl, in etwa

so, wie es seltsam erscheinen mag, dass die Amphibien an Land gekrochen sind, sich zu Säugetieren entwickelt und als Wale wieder ins Meer zurückgekehrt sind: Erst gehen freie Formen in einem Prozess der ökonomischen Rationalisierung und Aufteilung in immer kleiner Arbeitseinheiten verloren und kehren dann ausgerechnet wieder durch digitale Prozesse zurück. Das Physische wird über den Umweg des Virtuellen entwickelt, die Bedeutungen von Entwürfen werden in pure Information aufgelöst um daraus wieder semantische und pragmatische Zusammenhänge zu erzeugen. Das Prinzip des Natürlichen wird über das Technische begriffen, es wird über die Ökologie des Entwerfens geredet, während man Jahre seines Lebens vor dem Rechner sitzend verbringt und versucht, über die Abstraktheit des Codes wieder konkrete Substanz zu erzeugen. So gesehen scheint dieser ganzen digitalen Entwurfskultur beinahe etwas Nostalgisches anzuhaften, fast so, als ob sie in einem großen Umweg versuchen würde, etwas wiederzuentdecken, was im Laufe der kulturellen Evolution verloren gegangen ist.

Euklid hat seine Geometrie geschaffen um damit die Welt zu begreifen, gleichzeitig aber auch eine neue, symbolische Welt entworfen. Letztlich kann man nur hoffen und das Seinige dazu tun, dass sich algorithmisches Design nicht doch nur als ein digitaler *Hype* entwickelt, zu einem Symbol technologischer Machbarkeit, hinter dem die vielfältigen Bezüge unterschiedlicher Realitäten noch weiter verschwinden. Vielleicht ist es aber auch wirklich so, dass die Umwege doch notwendig sind, zur Transzendenz führen, zu einem wirklich erweiterten Designverständnis, dass den Menschen und seine biologische Natur dabei nicht vergisst und den immensen Reichtum der überaus komplexen Verflechtungen, in denen Entwerfen stattfindet, besser begreift und zu würdigen weiß.

11. Quellennachweise

Literatur

Prolog

[01] Dahl, Roald (1953): Der große automatische Grammatisator. In: Dahl, Roald (1996): Eine Kleinigkeit. Vier ungewöhnliche Geschichten. Rowohlt-Verlag, Reinbek.

Kapitel 1

[01] Turkle, Sherry (1998): Leben im Netz – Identität in Zeiten des Internet. Rowohlt Verlag, Reinbek. S. 137

[02] Kilian, A. (2006): Design Exploration through Bidirectional Modeling of Constraints. MIT, Cambridge.

[03] Frayling, C. (1993): Research in Art and Design, Royal College of Art Research Papers, Vol.1 No.1, London: Royal College of Art. S. 5.

[04] ebenda. S. 5.

Kapitel 2

[01] Jones, C., J. (1970): Design Methods. Wiley, Chichester, UK.

[02] Rittel, Horst W.J., Webber, Melvin M. (1972): Dilemmas in einer allgemeinen Theorie der Planung. In: Reuter, Wolf.D. (Hrsg.) (1992): Horst W. J. Rittel: Planen Entwerfen Design. Ausgewählte Schriften zu Theorie und Methodik. Kohlhammer. Stuttgart, S. 22.

[03] Evans, R. (1995): The Projective Cast: Architecutre and It's Three Geometries. MiT-Press, Cambridge.

[04] Maser, Siegfried (1972): Einige Bemerkungen zum Problem einer Theorie des Design, Manuskript, Braunschweig.

[05] Simon, H. A. (1973). The Structure of Ill-structured Problems. Artificial Intelligence, 4. S. 181-200.

[06] Rittel, Horst W.J., Webber, Melvin M. (1972): Dilemmas in einer allgemeinen Theorie der Planung. In: Reuter, Wolf.D. (Hrsg.) (1992): Horst W. J. Rittel: Planen Entwerfen Design. Ausgewählte Schriften zu Theorie und Methodik. Kohlhammer. Stuttgart, S. 21.

[07] ebenda. S.27

[08] Rittel, Horst W. J (1970): Der Planungsprozess als iterativer Vorgang von Varietätserzeugung und Varietätseinschränkung. In: Reuter, Wolf.D. (Hrsg.) (1992): Horst W. J. Rittel: Planen Entwerfen Design. Ausgewählte Schriften zu Theorie und Methodik. Kohlhammer. Stuttgart, S. 77f.

[09] Rittel, Horst W.J., Webber, Melvin M. (1972): Dilemmas in einer allgemeinen Theorie der Planung. In: Reuter, Wolf.D. (Hrsg.) (1992): Horst W. J. Rittel: Planen Entwerfen Design. Ausgewählte Schriften zu Theorie und Methodik. Kohlhammer. Stuttgart, S. 24.

[10] Mitchell, W. J. (1990): The Logic of Architecture. Design, Computation, and Cognition. MIT-Press, Cambridge. S. 109.

[11] Vitruv (~30 v.Chr): De Architectura libri decem.

[12] Aristid Lindenmayer (1968): „Mathematical models for cellular interaction in development." J. Theoret. Biology, 18. S. 280-315.

[13] Sei Watanabe, Makoto (2002): Induction Design - A Method for Evolutiononary Design.Birkhäuser Verlag, Basel. S .7

[14] Terzidis, Kostas (2003): Expressive Form. A Conceptual Approach to Computational Design. Spon Press, New York. S. 6.

[15] Jonas, W. (1994): Design - System - Theorie. Überlegungen zu einem systemtheoretischen Modell von Design-Theorie. Essen.

[16] Chouchoulas, Orestes (2003): Shape Evolution An Algorithmic Method for Conceptual Architectural Combining Shape Grammars and Genetic Algorithms. Centre for Advanced Studies in Architecture, Department of Architecture and Civil Engineering, University of Bath. S. 2f.

[17] Lawson, B. (2006): How Designers Think: The Design Process Demystified. 2nd. London: Butterworth Arthitecture. S. 42ff.

[18] Kolarevic, Branko (2003): Architecture in the Digital Age. Design and Manufacturing. Taylor & Francis, New York. S. 13.

[19] Terzidis, Kostas (2003): Expressive Form. A Conceptual Approach to Computational Design. Spon Press, New York.

[20] Andrasek, A (2006): www.biothing.org/genware. (Stand: November 2006)

[21] Negroponte, Nicholas (1995): Being Digital. Hodder and Stoughten, London.

[22] Dritsas, Stylianos (2004): Design Operators. S. 11ff. http:www.dritsas.net (Stand: November 2006)

[23] Menges, Achim (2006): Polymorphism. In: Hensel, M.; Menges, A.; Weinstock, M (Guest-Editors): Techniques and Technologies in Morphogenetic Design. AD, March/April 2006. S. 78-87.

[24] Lawson, B. (2006): How Designers Think: The Design Process Demystified. 2nd. London: Butterworth Arthitecture.

[25] de Bono, Edward (1971): Laterales Denken: Ein Kursus zur Erschließung ihrer Kreativitätsreserven. Rowohlt, Reinbek.

[26] Burgos, Nate; Kallish, Adam (2006): www.designmethods.org (Stand: April 2006)

[27] Kilian, A. (2006): Design Exploration through Bidirectional Modeling of Constraints. MIT, Cambridge.

[28] Lawson, B. (2006): How Designers Think: The Design Process Demystified. 2nd. London: Butterworth Arthitecture. S. 141f.

[29] Alexander, Christopher (1964): Notes on the Synthesis of Form. Harvard-University-Press, Cambridge.

[30] Kalay, Yehuda E. (2004): Architectures New Media. Principles, Theories and Methods of Computer-Aided Design. MIT-Press, Cambridge. S. 66.

[31] Turkle, Sherry (1984): Die Wunschmaschine. Vom Entstehen der Computerkultur. Rowohlt-Verlag, Reinbek.

[32] von Randow, Gero (1998): Roboter – Unsere nächsten Verwandten. Rowohlt-Verlag, Reinbek. S. 15f.

[33] Bürdek, Bernhard (1991): Design. Geschichte, Theorie und Praxis der Produktgestaltung. DuMont-Verlag, Köln. S. 308ff.

[34] Maeda (1999): Design by Numbers. MIT-Press, Cambridge. S. 20.

[35] ebenda. S. 20.

[36] Moneo, Rafael (2001): The Thing Called Architecture. In: Davidson, Cynthia (Hrsg.)(2001): Anything. Anyone Coorporation, New York. S. 120-123.

[37] Cache, Berhard (1995): Earth Moves. MIT-Press, Cambridge.

[38] Glaeser (2005): Geometrie und ihre Anwendungen in Kunst, Natur und Technik. Spektrum-Verlag. S. 231f.

[39] ebenda. S. 228.

[40] Kilian, A. (2006): Design Exploration through Bidirectional Modeling of Constraints. MIT, Cambridge.

[41] Turing, Alan (1950): Computing Machinery and Intelligence. Mind 59. S. 433-460.

[42] Turkle, Sherry (1995): Leben im Netz. Identität im Zeitalter des Internet. Rowohlt. Reinbek. S. 137ff.

[43] Searle (1982): The Myth of Computers. The NewYork Review of Books.

[44] Breitenberg (1972): Kybernetische Vehikel. Rowohlt-Verlag, Reinbek.

[45] www.spiegel.de (Orginallink nicht mehr verfügbar)

[46] Maeda (1999): Design by Numbers. MIT-Press, Cambridge. S. 23.

Kapitel 3

[01] Bolz, Norbert (2006): bang design. Design-Manifest des 21. Jahrhunderts. Trend-Verlag, Köln.

[02] von Förster, Heinz (2003): Understanding Understanding. Springer-Verlag, New York.

[03] http://www.gehrytechnologies.com

[04] http://www.kaisersrot.com

[05] Neufert, Ernst (2005): Bauentwurfslehre. Vieweg-Verlag, Braunschweig (38. Auflage).

[06] Vitruv (~30 v.Chr): De Architectura libri decem.

[07] Alexander, Christopher et al (1977)(1995): Eine Muster-sprache. Löcker-Verlag, Wien.

[08] Lindinger, Herbert (1983): Kriterien einer guten Industrieform. In: Die gute Industrieform, Hannover 1983.

[09] Burckhardt, Lucius (1977): Kriterien für ein neues Design. In: Burckhardt, Lucius (1995): Design ist unsichtbar. Cantz-Verlag, Ostfildern.

[10] Eco, Umberto (1962): La definizione dell'arte. In: Introduzione al catalogo della mostra Arte programmata, negozio Olivetti. Garzanti, Mailand. S.23.

[11] Galanter, Philip (2003): What is Generative Art? Complexity Theory as a Context for Art Theory. In: Soddu, Celestino (2003) : Generative Art 2003. Proceedings. Ohne Verlag, Mailand. S. 219.

[12] Schuhmacher, Patrik (1999):Rational in Retrospect - Reflections on the Logic of Rationality in Recent Design. Unedited version. Edited in: AA files 38, Annals of the Architectural Association School of Architecture.

[13] Schischkoff, Georgi (Hrsg.) (1991): Philosophisches Wörterbuch. Kröner-Verlag, Stuttgart.

[14] Lawson, Bryan (2006): How Designers Think – The design process demystified. Architectural Press, London. S. 90-106.

[15] Rittel, Horst W. J (1970): Der Planungsprozess als iterativer Vorgang von Varietätserzeugung und Varietätseinschränkung. In: Reuter, Wolf.D. (Hrsg.) (1992): Horst W. J. Rittel: Planen Entwerfen Design. Ausgewählte Schriften zu Theorie und Methodik. Kohlhammer. Stuttgart, S. 85ff.

[16] Kilian, A. (2006): Design Exploration through Bidirectional Modeling of Constraints. MIT, Cambridge.

[17] Lynn, Greg (1999): Animate Form. Princeton Architectural Press, New York. S. 31f.

[18] Turing, Alan (1936): On Computable Numbers, With an Application to the Entscheidungsproblem, Proceedings of the London Mathematical Society, Series 2, Volume 42.

[19] Mitchell, William, J. (2003): Design Worlds and Fabrication Machines. In: Kolarevic, Branco (Hrsg.)(2003): Architecture in the Digital Age – Design and Manufacturing. Spon-Press, New York., S. 75.

[20] Dawkins, Richard (1986): The Blind Watchmaker. Norton, London. S.46ff.

[21] Mitchell, William, J. (2003): Design Worlds and Fabrication Machines. In: Kolarevic, Branco (Hrsg.)(2003): Architecture in the Digital Age – Design and Manufacturing. Spon-Press, New York., S. 75.

[22] Gell-Mann, Murry (1995): What is Complexity? Complexity, John Whiley and Sons, Vol. 1 nr. 1. S. 16-19.

[23] http://de.wikipedia.org/wiki/Algorithmus (Stand: April 2005)

[24] Dehlinger, Hans (2003): Generative Prozesse und Verfahren. Vorlesungsskript. Nicht veröffentlicht.

[25] Nake, Frieder (1967): Teamwork zwischen Künstler und Computer. In: Büscher, Barbara et al (Hrsg.)(2004): Ästhetik als Programm, Max Bense/Daten und Streuungen. Kaleidoskopen, Heft 5. S.220-225.

[26] Dritsas, Stylianos (2004): Design Operators. S. 16. http:www.dritsas.net (Stand: November 2006)

[27] http://de.wikipedia.org/wiki/Church-Turing-These (Stand: Oktober 2005)

[28] Kalay, E. Yehuda (2004): Architecture's New Media. Principles, Theories and Methods of Computer-Aided Design. MIT Press, Cambridge. S. 3.

[29] Wittgenstein, Ludwig (1921): Tractatus Logico-Philosophicus. Vorrede. http://www.gutenberg.org/etext/5740 (Stand: April 2007)

[30] ebenda, 5.62.

Kapitel 4

[01] Schein, Markus (2003): Next Gerneration(s) Computer Aided Design – Next Generation CAD Education?. In: iF International Forum Design GmbH: ICSID 2nd Education Conference, Critical Motivations and New Dimensions. Ohne Verlag, Hannover. S. 169-176.

[02] von Kleist, Heinrich (1805): Über die allmähliche Verfertigung der Gedanken beim Reden. http://www.kleist.org/texte/index.htm (Stand: April 2007)

[03] Bauer, Hans; Dehlinger, Hans; Precht, Hans J.(1991): Design, Kunst, Computer. Jenior & Preßler, Kassel. S. 134f.

[04] Kalay, E. Yehuda (2004): Architecture's New Media. Principles, Theories and Methods of Computer-Aided Design. MIT Press, Cambridge. S. 99f.

[05] deLanda, Manuel (2002): Deleuze and the Use of the Genetic Algorithm in Architecture. http://www.cddc.vt.edu/host/delanda/pages/algorithm.htm (Stand: April 2007)

[06] Franken, Bernhard (2003): Real as Data. In: Kolarevic, Branco (Hrsg.)(2003): Architecture in the Digital Age – Design and Manufacturing. Spon-Press, New York., S. 128ff.

[07] Beermann, Jana; Seeland, David (2007): Space Layers. In: Smart Structures. Kooperationprojekt der Universitäten Kassel, Delft und Knoxville. Veröffentlichung geplant für Mitte 2008.

[08] Speaks, Michael (2002): Design Intelligence: Or Thinking at the End of Metaphysics. In: Sharples, Holden, Pasquarelli (Hrsg.)(2002): Versioning – Evolutionary Techniques in Architecture. Architectural Design, Vol. 72. S. 5

[09] ebenda, S. 6

[10] Willems, Daan (2006): Kunstmatig Evolutionair Ontwerpen. Masterarbeit im Fachbereich Architektur der TU Eindhoven. Nicht veröffentlicht.

[11] Schein, Markus; Werner, Ole; Zimmermann, Gregor (2004): Sweet-Greens – A Computergenerated Landscape. Books on Demands, Norderstedt, o.V.

[12] Borges, Jorge, L. (1941): Die Bibliothek von Babel. Reclam-Verlag, Stuttgart.

[13] Watanabe, Makoto Sei (2002): Induction Design - A Method for Evolutionary Design. Birkhäuser-Verlag, Basel. S. 7.

[14] deLanda, Manuel (2002): Deleuze and the Use of the Genetic Algorithm in Architecture. http://www.cddc.vt.edu/host/delanda/pages/algorithm.htm (Stand: April 2007)

[15] Mayr, Ernst (2001)(2005): Das ist Evolution. Goldmann-Verlag, München. S. 25.

[16] Lawson, Bryan (2006): How Designers Think – The design process demystified. Architectural Press, London. S. 41ff.

[17] deLanda, Manuel (2002): Deleuze and the Use of the Genetic Algorithm in Architecture. http://www.cddc.vt.edu/host/delanda/pages/algorithm.htm (Stand: April 2007)

[18] Reiser + Umemoto (2006): Atlas of Novel Tectonics. Princeton Architectural Press, New York. S. 20.

[19] Kolarevic, Branco (Hrsg.)(2003): Architecture in the Digital Age – Design and Manufacturing. Spon-Press, New York., S. 8f.

[20] Scheurer Fabian (2003): The Groningen Twister. An experiment in applied generative design. In: Soddu, Celestino: Generative Art 2003. Proceedings. Ohne Verlag, Mailand. S.90-99.

[21] Werner, Ole (2003): Konstruktion von Entwurfsräumen. Diplomarbeit aus dem Studiengang Produktdesign.Kunsthochschule der Universität Kassel. Nicht veröffentlicht.

[22] Menges, Achim (2005): Polymorphism. In: Hensel, M.; Menges, A.; Weinstock, M (Guest-Editors): Techniques and Technologies in Morphogenetic Design. AD, March/April 2006. S. 78-87.

[23] Rittel, Horst W. J (1976): Sachzwänge – Ausreden für Entscheidungsmüde. In: Reuter, Wolf.D. (Hrsg.) (1992): Horst W. J. Rittel: Planen Entwerfen Design. Ausgewählte Schriften zu Theorie und Methodik. Kohlhammer. Stuttgart, S. 271-281.

[24] Krausse, Joachim; Lichtenstein, Claude (1999): Your Private Sky – R. Buckminster Fuller. Verlag Lars Müller, Baden. S. 402.

[25] Lawson, B. (2006): How Designers Think: The Design Process Demystified. 2nd. London: Butterworth Arthitecture. S. 46f.

[26] Kolarevic, Branco (Hrsg.)(2003): Architecture in the Digital Age – Design and Manufacturing. Spon-Press, New York., S. 8ff.

[27] Werner, Ole (2003): Konstruktion von Entwurfsräumen. Diplomarbeit aus dem Studiengang Produktdesign.Kunsthochschule der Universität Kassel. Nicht veröffentlicht.

[28] Etscheit, Georg (2002): Maßgeschneidertes vom Fließband. In: DIE ZEIT, Ausgabe 05/2002. Zeitverlag, Hamburg.

[29] Speaks, Michael (2002): Design Intelligence: Or Thinking at the End of Metaphysics. In: Sharples, Holden, Pasquarelli (Hrsg.)(2002): Versioning – Evolutionary Techniques in Architecture. Architectural Design, Vol. 72. S. 5

[30] Lynn, Greg (1999): Animate Form. Princeton Architectural Press, New York.

[31] Werner, Ole (2005): Persönliche Kommunikation.

[32] Soddu, Celestino (2002): La Citta Ideale – Generative Codes Design Identity. In: Soddu, Celestiono (Hrsg.)(2002): Generative Art 2002. Ohne Verlag, Polytecnico Milano.

[33] Willems, Daan (2006): Persönliche Kommunikation.

[34] deLanda, Manuel (2001): Deleuze and the Use of the Genetic Algorithm in Architecture. http://www.cddc.vt.edu/host/delanda/pages/algorithm.htm (Stand: April 2007)

[35] Reiser + Umemoto (2006): Atlas of Novel Tectonics. Princeton Architectural Press, New York.

[36] Zitiert aus: McNeill, Daniel; Freiberger, Paul (1996): „Fuzzy Logic – Die ‚unscharfe' Logik erobert die Technik. Droemersche Verlagsanstalt Th. Knauer Verlag, München. S. 118.

[37] Reiser + Umemoto (2006): Atlas of Novel Tectonics. Princeton Architectural Press, New York. S. 50f.

Kapitel 5

[01] Dawkins, Richard (1977)(1989): The Selfish Gene. Oxford University Press, Oxford. S.192.

[02] Dawkins, Richard (1983): Universal Darwinsim. In: Bendal, D.S. (Hrsg.)(1983): Evolution from Molecules to Men. Cambridge University Press, Cambridge.

[03] Gatherer, Derek (1999): The Memetics of Design. In: Bentley, Peter (Hrsg.) (1999): Evolutionary Designs by Computer. Morgan Kauffmann, San Franscisco. S. 102.

[04] ebenda. S. 98.

[05] Darwin, Charles (1859)(1985): The Origin of Species. Penguin Books, London.

[06] French, Michael (1999): The Interplay of Evolution and Insigth in Design. In: Bentley, Peter (Hrsg.) (1999): Evolutionary Designs by Computer. Morgan Kauffmann, San Franscisco. S. 77-89.

[07] Langrish, John (2006): Evolutionary Design – Ten Years On: Memes and Natural Selection. In: Jonas, Wolfgang; Chow, Rosan; Verhaag, Niels: Proceedings of the 6th International Conference of the European Academy of Design. Ohne Verlag, Bremen.

[08] Gatherer, Derek (1999): The Memetics of Design. In: Bentley, Peter (Hrsg.) (1999): Evolutionary Designs by Computer. Morgan Kauffmann, San Franscisco. S. 102.

[09] Rechenberg, Ingo (1973): Evolutionsstrategie. Optimierung technischer Systeme nach den Prinzipien der biologischen Evolution. Frommann-Holzboog. Stuttgart.

[10] Holland, J. H. (1973): Genetic Algorithms and the Optimal Allocations of Trials. In: SIAM Journal of Computing 2:2.. S. 88-105.

[11] Holland, J. H. (1975): Adaption in Natural and Artificial Systems. University of Michigan Press, Ann Harbor.

[12] Goldberg, D.E. (1989): Genetic Algorithms as a Computational Theory of Conceptual Design.Addison-Wesley..

[13] Fogel, Lawrence, J. (1963): Biotechnology: Concepts and Applications. Prentice Hall, Englewood Cliffs.

[14] Fogel, David (1992): Evolving Artificial Intelligence. PhD thesis, University of California, San Diego.

[15] Koza, J. (1992): On the Programming of Computers by Means of Natural Selection. MIT Press, Cambridge.

[16] Bentley, Peter (1999): An Introduction to Evolutionary Design by Computers. In: Bentley, Peter (Hrsg.) (1999): Evolutionary Designs by Computer. Morgan Kauffmann, San Franscisco. S. 25-34.

[17] Koza, J.R. et al (1997): Automated Synthesis of an Analogue Electric Circuit to implement the robot controller by means of genetic programming. Proceedings of IEEE International Symposium on Computational Intelligence in Robotics and Automation. Computer Society Press, S. 340-346.

[18] Bruns, J.; Kaps, L. (2006): Optimierung von Bauteilverbindungen in PKW-Karosserien mit genetischen Algorithmen unter Betriebsfestigkeitsrandbedingungen. In: Nafems Magazin, 1/2006.

[19] Eby, David et al (1999): The Optimization of Flywheels using an Injection Island Genetic Algorithm. In: Bentley, Peter (Hrsg.) (1999): Evolutionary Designs by Computer. Morgan Kauffmann, San Franscisco. S. 167-188.

[20] Sims, Karl (1994): Evolving Virtual Creatures. In: Sigraph94 - Proceedings. S. 15-22

[21] Kilian, Axel (2003): 6.836 Embodied Intelligence - Using Genetic Algorithms To Generate Developable Strips From Free formed Surfaces. http://destech.mit.edu/akilian/ (Stand: April 2006)

[22] Willems, Daan (2006): Kunstmatig Evolutionair Ontwerpen. Masterarbeit im Fachbereich Architektur der TU Eindhoven. Nicht veröffentlicht.

[23] Menges, Achim: (o.a.): morphogenetic design experiment 01 - mulit planar. http://www.achimmenges.net/ (Stand: April 2007)

[24] Bentley, Peter (1999): From Coffee Tables to Hospitals: Generic Evolutionary Design. In: Bentley, Peter (Hrsg.) (1999): Evolutionary Designs by Computer. Morgan Kauffmann, San Franscisco. S. 405-423.

[25] ebenda, S. 413.

[26] ebenda, S. 412.

[27] Hornby, Gregor; Pollack, Jordan (2001): The Advantages of Generative Grammatical Encodings für Physical Design. Congress On Evolutionary Computation. http://www.demo.cs.brandeis.edu/papers/long.html#hornby_cec01 (Stand: April 2006)

[28] Frazer, John (1995): An Evolutionary Architecture. Architectural Association, London.

[29] Sims, Karl (1991): Artificial Evolution for Computer Graphic. In: Computer Graphics, 25(4), S. 319-328.

[30] Dawkins, Richard (1986): The Blind Watchmaker. Norton, London. S. 335-358.

[31] Dean, Lionel; Atkinson, Paul; Unver, Ertu (2005): Evolving Individualised Consumer Products. In: Jonas, Wolfgang; Chow, Rosan; Verhaag, Niels: Proceedings of the 6th International Conference of the European Academy of Design. Ohne Verlag, Bremen.

[32] Hung, Kwai Chan; Frazer, John H.; Xi, Tang Ming (2002): Interactive Evolutionary Design in a Hierachical Way. Design Technology Research Center, School of Design, Hong Kong Polytechnic University, ohne Verlag.

[33] Dawkins, Richard (1986): The Blind Watchmaker. Norton, London. S. 73.

[34] Sims, Karl (1991): Artificial Evolution for Computer Graphic. In: Computer Graphics, 25(4), S. 319-328.

[35] deLanda, Manuel (2002): Deleuze and the Use of the Genetic Algorithm in Architecture. http://www.cddc.vt.edu/host/delanda/pages/algorithm.htm (Stand: April 2007)

[36] Mitchell, W. J. (1990): The Logic of Architecture. Design, Computation, and Cognition. MIT-Press, Cambridge.

[37] Burry, Mark (2003): Between Intuition and Process: Parametric Design and Rapid Prototyping. In: Kolarevic, Branco (Hrsg.)(2003): Architecture in the Digital Age – Design and Manufacturing. Spon-Press, New York., S. 149f.

[38] De Luca, F. and Nardini, M. (2002): Behind the Scences: Avant-Garde Techniques in Contemporary Design. Birkhäuser Verlag, Basel.

[39] Kolarevic, Branco (2003): Digital Morphogenesis. In: Kolarevic, Branco (Hrsg.)(2003): Architecture in the Digital Age – Design and Manufacturing. Spon-Press, New York., S. 18f..

[40] Burry, Mark (2003): Between Intuition and Process: Parametric Design and Rapid Prototyping. In: Kolarevic, Branco (Hrsg.)(2003): Architecture in the Digital Age – Design and Manufacturing. Spon-Press, New York., S. 154ff.

[41] De Luca, F. and Nardini, M. (2002): Behind the Scences: Avant-Garde Techniques in Contemporary Design. Birkhäuser Verlag, Basel.

[42] Franken, Bernhard (2003): Real as Data. In: Kolarevic, Branco (Hrsg.)(2003): Architecture in the Digital Age – Design and Manufacturing. Spon-Press, New York., S. 131.

[43] De Luca, F. and Nardini, M. (2002): Behind the Scences: Avant-Garde Techniques in Contemporary Design. Birkhäuser Verlag, Basel.

[44] Schein, Markus (2005): Parametrisches Design. Lehrprojekt an der Kunsthochschule der Universität Kassel. Digitalpool. Nicht veröffentlicht.

[45] Grohmann, Manfred; Schein, Markus; Tessmann, Oliver; Zimmermann, Gregor (2007): inExterior-Structures. Books On Demand, Norderstett, o.V.

[46] Becker, M.; Büchel, A.; Haberfellner, R.; Nagel, P.; von Massow, H. (1994): Systems Engineering – Methode und Praxis. Verlag Industrielle Organisation, Zürich.

[47] Rittel, Horst W. J (1970): Der Planungsprozess als iterativer Vorgang von Varietätserzeugung und Varietätseinschränkung. In: Reuter, Wolf.D. (Hrsg.) (1992): Horst W. J. Rittel: Planen Entwerfen Design. Ausgewählte Schriften zu Theorie und Methodik. Kohlhammer. Stuttgart, S. 85ff.

[48] Asimov, M. (1962): Introduction to Design, Prentice-Hall, Englewood Cliffs, New Jersey.

[49] Gero, J. S. (1999) Constructive memory in design thinking, In: Goldschmidt, G. and Porter, W. (Hrsg.)(1999): Design Thinking. Research Symposium: Design Representation, MIT, Cambridge. S. I.29-35.

[50] Lawson, Bryan (2006): How Designers Think – The design process demystified. Architectural Press, London. S. 48f.

[51] Zimmermann, Gregor (2007): Membran Beton Gitterschalen Tragwerke. Books On Demand, Norderstett.

[52] ebenda, S. 149.

Kapitel 6

[01] Glaeser, Georg (2004):Die Theorie der Freiformflächen. In: R. Burgard (Hrsg.)(2004):Kunststoffe und freie Formen. Springer-Verlag, Wien, New York. S. 45-52

[02] Glaeser, Georg (2005): Geometrie und ihre Anwendungen in Kunst, Natur und Technik. Spektrum-Verlag. S. 231f.

[03] Aranda, Benjamin; Lasch, Chris (2006): Pamphlet Architecture 27 – Tooling.Princeton Architectural Press, New York. S. 74-91.

[04] ebenda, S. 10-21.

[05] Sasaki, M: (2004): Shape Design of Free Curved Surface Shells. In: a+u – Architecture and Urbanism. Nr. 404. S.36.

[06] Schlaich, Jörg; Schober, Hans (2002): Filigrane Kuppeln. Beispiele, Tendenzen und Entwicklungen. In: tec21, Heft 12/2002, S.23.

[07] Radin,Charles (1994): The Pinwheel Tilings of the Plane. In: The Annals of Mathematics, second Series, Volume 139, Issue 3, S. 661-702.

[08] Schlaich, Jörg; Schober, Hans (2002): Filigrane Kuppeln. Beispiele, Tendenzen und Entwicklungen. In: tec21, Heft 12/2002, S.26

[09] Kilian, Axel (ohne Angabe): 6.836 Embodied Intelligence - Using Genetic Algorithms To Generate Developable Strips From Free formed Surfaces. http://destech.mit.edu/akilian/ (Stand: April 2006)

[10] Sasaki, M: (2005): Shape Design of Free Curved Surface Shells. In: a+u – Architecture and Urbanism. Nr. 404. S.37.

[11] Cui, C.; Ohmori, H.; Sasaki, M. (2003): Computational Morphogenesis of 3D-Structures by Extended Eso Methods. In: IASS (Journal of the International Association for Shell and Spatial Structures. Vol. 44. S.51-61.

[12] Kilian, A. (2006): Design Exploration through Bidirectional Modeling of Constraints. MIT, Cambridge. S. 82-89.

[13] Dritsas, Stylanios (2003): Design Scripting Library r10 http://www.dritsas.net/ (Stand: April 2007)

[14] Sedgewick, Robert (2002): Algorithmen. Pearson Education Deutland, München.

[15] Zimmermann, Gregor (2007): Membran Beton Gitterschalen Tragwerke. Books On Demand, Norderstett.

Kapitel 8

[01] Anmerkung Autor

[02] Anmerkung Autor

[03] Glaeser (2005): Geometrie und ihre Anwendungen in Kunst, Natur und Technik. Spektrum-Verlag. S. 232.

[04] Dawkins, Richard (1976)(1989): The Selfish Gene. Oxford University Press, Oxford.

[05] Rechenberg, Ingo (1994): Evolutionsstrategie 94. Frommann-Holzboog. Stuttgart. S. 222.

[06] Willems, Daan (2006): Kunstmatig Evolutionair Ontwerpen. Masterarbeit im Fachbereich Architektur der TU Eindhoven. Nicht veröffentlicht.

[07] Bentley, Peter (1999): Evolutionary Design by Computers. Morgan Kauffmann, San Franscisco. S. 31f.

[08] Anmerkung Autor

[09] Anmerkung Autor

[10] Lin S. C.; Goodmann, E.D.; Punch, W. F. (1994): Coarse Grain Parallel Genetic Algorithms: Categories and New Approaches. In: IEEE Conference on Parallel and Distributed Processing.

[11] Rechenberg, Ingo (1994): Evolutionsstrategie 94. Frommann-Holzboog. Stuttgart. S. 232.

Kapitel 9

[01] Schlaich, Jörg; Schober, Hans (2002): Filigrane Kuppeln. Beispiele, Tendenzen und Entwicklungen. In: tec21, Heft 12/2002, S.26.

Kapitel 10

[01] Maser, Siegfried (2001): Zur Planung gestalterischer Projekte. Beiträge zur Designtheorie. Band 2. Bergische Universität Wuppertal. S. 107.

[02] Reiser + Umemoto (2006): Atlas of Novel Tectonics. Princeton Architectural Press, New York.

Abbildungen

Bei den Abbildungen handelt es sich um Bildzitate nach §51 UrhG mit entsprechenden Quellenangaben zur Erläuterung des Inhalts – im Sinne der Vervielfältigung, Verbreitung und öffentlichen Wiedergabe, in einem durch den Zweck gebotenen Umfang in einem selbständigen wissenschaftlichen Werk. Im folgenden aufgeführt sind die Quellenangaben zu den verwendeten Abbildungen, soweit bekannt wurde der Urheber genannt. Bei allen nicht aufgeführten Abbildungen liegen die Rechte beim Autor dieser Arbeit.

Kapitel 1

Abb. 01	*http://wiki.arch.ethz.ch/twiki/bin/view/RZM/MedRaume*
Abb. 02	*Clemens Ortmeyer; http://phaeno.de/180.html*
Abb. 03 / links	*http://www.decopasion.com/dossier/la_iluminacion/lamparas_suspension_2/bunnylamp_mgx*
Abb. 03 / rechts	*http://www.unicahome.com/p25208/materialise/hidden-mgx-vase-by-dan-yeffet.html*
Abb. 04	*Hensel, M.; Menges, A. (2006): Morpho-Ecologies. AA Publications, London.*

Kapitel 2

Abb. 01 / oben	*http://algorithmicbotany.org/lstudio/flyer.pdf*
Abb. 01 / unten	*Martin Hansmayer; http://www.mh-portfolio.com/*

Kapitel 3

Abb. 01	*Nach : Lawson, Bryan (2006): How Designers Think – The design process demystified. Architectural Press, London. S. 106.*

Kapitel 4

Abb. 02	*Jana Beermann, David Seeland.*
Abb. 04	*Willems, Daan (2006): Kunstmatig Evolutionair Ontwerpen. Masterarbeit im Fachbereich Architektur der TU Eindhoven. Nicht veröffentlicht.*
Abb. 05	*Henrik Hornung, aus: Schein, Markus; Werner, Ole; Zimmermann, Gregor (2004): Sweet-Greens – A Computergenerated Landscape. Books on Demands, Norderstedt, ohne Verlag.*
Abb. 06 / oben	*KCAP*
Abb. 06 / unten	*ARUP / Fabian Scheurer*
Abb. 07 / o. links	*http://www.architonic.com/de/cat/gal/1000090*
Abb. 07 / o. rechts	*http://www.architonic.com/de/cat/gal/1000090*
Abb. 07 / u. links	*http://www.architonic.com/de/cat/gal/1000094*
Abb. 07 / u. rechts	*http://designmatcher.com/nl/images/objects/5049.jpg*
Abb. 08	*Werner, Ole (2003): Konstruktion von Entwurfsräumen. Diplomarbeit aus dem Studiengang Produktdesign. Kunsthochschule der Universität Kassel. Nicht veröffentlicht.*
Abb. 09	*http://www.omnispace.org/my_weblog/architecture/index.html*
Abb. 10	*Vogt und Weizenegger*
Abb. 11 / unten	*Timm Herok*
Abb. 12	*dEcoi*
Abb. 13 / links	*http://www.makita.de/*
Abb. 13 / rechts	*http://www.wp-bilderwelten.de/monatsbilder-06-htm/monatsbild-januar-06.htm*

Abb. 14	*Quelle unbekannt.*
Abb. 15	*www.FutureFactories.com*
Abb. 16	*www.biothing.org*

Kapitel 5

Abb. 01	*Nach: Bentley, Peter (1999): An Introduction to Evolutionary Design by Computers. In: Bentley, Peter (Hrsg.) (1999): Evolutionary Designs by Computer. Morgan Kauffmann, San Franscisco. S. 28.*
Abb. 02	*Sims, Karl (1994): Evolving Virtual Creatures. In: Sigraph94 - Proceedings. S. 21.*
Abb. 03	*Willems, Daan (2006): Kunstmatig Evolutionair Ontwerpen. Masterarbeit im Fachbereich Architektur der TU Eindhoven. Nicht veröffentlicht. S. 42.*
Abb. 04	*http://www.achimmenges.net*
Abb. 05 / oben	*Bentley, Peter (1999): From Coffee Tables to Hospitals: Generic Evolutionary Design. In: Bentley, Peter (Hrsg.) (1999): Evolutionary Designs by Computer. Morgan Kauffmann, San Franscisco. S. 41.*
Abb. 05 / unten	*ebenda: S. 414.*
Abb. 06	*Hornby, Gregor; Pollack, Jordan (2001): The Advantages of Generative Grammatical Encodings für Physical Design. Congress On Evolutionary Computation. S. 6. http://www.demo.cs.brandeis.edu/papers/long.html#hornby_cec01 (Stand: April 2006)*
Abb. 07	*Richard Dawkins. http://www.simonyi.ox.ac.uk/dawkins/WorldOfDawkins-archive/Dawkins/Work/Books/safari.gif*
Abb. 08	*www.FutureFactories.com*
Abb. 09	*Hung, Kwai Chan; Frazer, John H.; Xi, Tang Ming (2002): Interactive Evolutionary Design in a Hierachical Way. Design Technology Research Center, School of Design, Hong Kong Polytechnic University, ohne Verlag.*
Abb. 12	*Burry, Mark (2003): Between Intuition and Process: Parametric Design and Rapid Prototyping. In: Kolarevic, Branco (Hrsg.)(2003): Architecture in the Digital Age – Design and Manufacturing. Spon-Press, New York.*
Abb. 13	*http://www.grimshaw-architects.com*
Abb. 14	*Kolarevic, Branco (Hrsg.)(2003): Architecture in the Digital Age – Design and Manufacturing. Spon-Press, New York., S. 19.*
Abb. 15	*Burry, Mark (2003): Between Intuition and Process: Parametric Design and Rapid Prototyping. In: Kolarevic, Branco (Hrsg.)(2003): Architecture in the Digital Age – Design and Manufacturing. Spon-Press, New York.*
Abb. 16	*Kaas Oosterhuis. http://www.oosterhuis.nl/quickstart/index.php?id=117*
Abb. 17	*Bernhard Franken. http://www.franken-architekten.de/*
Abb. 18 / hinten	*Emmy Galle, Frank Bayer*
Abb. 19	*Gerhard Ennecker*
Abb. 20	*Nach: Becker, M.; Büchel, A.; Haberfellner, R.; Nagel, P.; von Massow, H. (1994): Systems Engineering – Methode und Praxis. Verlag Industrielle Organisation, Zürich.*
Abb. 21	*Nach: Rittel, Horst W. J (1970): Der Planungsprozess als iterativer Vorgang von Varietätserzeugung und Varietätseinschränkung. In: Reuter, Wolf.D. (Hrsg.) (1992): Horst W. J. Rittel: Planen Entwerfen Design. Ausgewählte Schriften zu Theorie und Methodik. Kohlhammer. Stuttgart.*
Abb. 22	*Nach: Asimov, M. (1962): Introduction to Design, Prentice-Hall, Englewood Cliffs, New Jersey.*
Abb. 23	*Nach: Lawson, Bryan (2006): How Designers Think – The design process demystified. Architectural Press, London. S. 49.*
Abb. 24 / links	*Gregor Zimmermann*
Abb. 24 / rechts	*http://www.deskproto.com/download/pr_dtower2004.htm*
Abb. 25	*Gregor Zimmermann*

Kapitel 6

Abb. 02 / oben	*http://www.vitra.de/products/home/living_room_stools_benches/ulmer_hocker/default.asp?lang=de_de*
Abb. 02 / unten	*http://www.greatbuildings.com/cgi-bin/gbi.cgi/Smith_House.html/cid_1126770230_9c.html*
Abb. 05 / oben	*http://www.greatbuildings.com/cgi-bin/gbi.cgi/Smith_House.html/cid_1126770230_9c.html*
Abb. 05 / unten	*Quelle nicht mehr verfügbar, ehemals: http://www.markanto.de/*
Abb. 06	*Benjamin Aranda, Chris Lash. http://www.terraswarm.com/projects/grotto.html*
Abb. 07	*Benjamin Aranda, Chris Lash. http://www.terraswarm.com/projects/vegas.html*
Abb. 08	*http://www.jakobmacfarlane.com/*
Abb. 09	*Spuybroek, Lars (2004): NOX - Bauten und Projekte. Deutsche Verlagsanstalt, München. S.144/145.*
Abb. 10	*Frei Otto; Quelle unbekannt.*
Abb. 11	*http://daddytypes.com/2004/05/30/eames_rocker_the_modern_solution.php*
Abb. 12 / links	*http://www.decopasion.com/dossier/la_iluminacion/lamparas_suspension_2/bunnylamp_mgx*
Abb. 12 / rechts	*http://www.unicahome.com/p25208/materialise/hidden-mgx-vase-by-dan-yeffet.html*
Abb. 13	*Bernhard Franken. http://www.franken-architekten.de/*
Abb. 14	*Jürgen Mayer H.. http://www.jmayerh.com/*
Abb. 15	*J. Scott Bovitz. http://bovitz.com/photo/traditional/jpgphotos/2003/walt_disney_concert_hall_fr.jpg*
Abb. 16	*William Nicholson. http://commons.wikimedia.org/wiki/Image:The_Sage_Gateshead.jpg*
Abb. 17	*http://en.wikipedia.org/wiki/Image:Eden_project.JPG*
Abb. 18	*Ultan Guilfoyle. http://www.moviesintofilm.com/images/sketches.htm*
Abb. 19	*http://www.mero-tsk.de/fileadmin/bilder/bausys/raumstabwerke/messe_mail3.jpg*
Abb. 20	*P. Bourke. http://mathworld.wolfram.com/AperiodicTiling.html*
Abb. 21	*http://deu.archinform.net/medien/00004929.htm*
Abb. 22	*http://www.newitalianblood.com/show.pl?id=687*
Abb. 23	*Kilian, Axel (2003): 6.836 Embodied Intelligence - Using Genetic Algorithms To Generate Developable Strips From Free formed Surfaces.*
Abb. 24	*Mirko Becker, Oliver Tessmann*
Abb. 25	*http://www.elcroquis.es/MagazineDetail.aspx?magazinesId=22&lang=en*
Abb. 26	*Sasaki, M: (2005): Shape Design of Free Curved Surface Shells. In: a+u –Architecture and Urbanism. Nr. 404. S.36.*

Kapitel 9

Abb. 05	*Christine Krüger*
Abb. 09	*Nach: Schlaich, Jörg; Schober, Hans (2002): Filigrane Kuppeln. Beispiele, Tendenzen und Entwicklungen. In: tec21, Heft 12/2002, S.26.*