Nach vor und zurück | Verschränkung digitaler und physischer Gestaltungsprozesse

Masterarbeit

Nach vor und zurück

Verschränkung digitaler und physischer Gestaltungsprozesse

ausgeführt zum Zwecke der Erlangung des
akademischen Grades eines Masters
unter der Leitung von

Christian Kern
Univ.Prof. Arch. Dipl.Ing.
E264/2
Institut für Kunst und Gestaltung

eingereicht an der Technischen Universität Wien
Fakultät für Architektur und Raumplanung
von

Kathrin Dörfler
0330982

Blindengasse 33/7/136
1080 Wien

in Zusammenarbeit mit

Romana Rust
Große Neugasse 11
1040 Wien

Wien, am 08.Mai 2012

Unterschrift

Ich habe zur Kenntnis genommen, dass ich zur Drucklegung meiner Arbeit unter der Bezeichnung

Masterarbeit

nur mit Bewilligung der Prüfungskommission berechtigt bin.

Ich erkläre weiters an Eides statt, dass ich meine Masterarbeit nach den anerkannten Grundsätzen für wissenschaftliche Abhandlungen selbständig ausgeführt habe und alle verwendeten Hilfsmittel, insbesondere die zugrunde gelegte Literatur genannt habe.

Wien, am 08. Mai

Unterschrift

Nach vor und zurück

Verschränkung digitaler und physischer Gestaltungsprozesse

Kathrin Dörfler und Romana Rust

Inhaltsverzeichnis

1 Abstract

1 Abstract

1. 1 Abstract

Im Rahmen der Masterarbeit werden Verfahren der interaktiven Bedienung von CNC-Maschinen untersucht, ausgehend von Algorithmen, die zwei- oder dreidimensionale Strukturen generieren.

In den herkömmlichen Methoden der CNC-gestützten Umsetzung von virtuellen Modellen in physische müssen alle gestalterischen Entscheidungen bereits getroffen sein, die Herstellungsprozesse sind linear und unflexibel, sie nützen das Potenzial der Maschinen für den/die Gestalter/in noch nicht aus.

Im Rahmen dieser Arbeit werden Systeme untersucht und geeignete Randbedingungen festgelegt, die menschliche Eingriffe intuitiver Art in den Fertigungsprozess ermöglichen, um Schnittstellen zwischen virtueller und physischer Wirklichkeit zu etablieren.

Ein Kuka Industrieroboter wird um offene Schnittstellen erweitert, die die online Steuerung des Roboters durch ein externes System ermöglicht. Dieses kann seinerseits mit verschiedensten Ein- und Ausgabegeräte sowie mit Sensoren verbunden werden. So wird das proprietäre Robotersystem für plattformunabhängige Erweiterungen geöffnet und die Grundlage geschaffen für interaktive Gestaltungsprozesse, die die Maschine nicht nur als Fertigungsautomat sondern als Gestaltungswerkzeug begreifen.

Neben den untersuchten theoretischen Grundlagen werden die diskutierten Konzepte in der Praxis mit systematischen Experimenten getestet.

2 Einführung

2 Einführung

2. 1 Integration physischer und digitaler Designmethoden

Entwerfen in der Architektur ist ein nichtlinearer Prozess der Wahrnehmung und der Verarbeitung. Der Prozess wird getragen vom Einsatz verschiedener Entwurfswerkzeuge digitaler und physischer Natur.

Als digitale Entwurfswerkzeuge haben sich dazu verschiedene Programme etabliert (Grasshopper, Generative Components, CATIA, etc.), die mehr als ein digitales Zeichenbrett sind, sondern Netzwerke und Strukturen wechselseitiger Abhängigkeiten abbilden können. Ihr Einsatz erfordert eine Rationalisierung und Quantifizierung von Designproblemen.

Mit der fortschreitenden Verwendung von digitalen Werkzeugen wurde der Abstand zu den physischen Entwurfswerkzeugen vergrößert, und können sich somit nicht mehr optimal ergänzen.

Dabei leistet die Auseinandersetzung mit Materialien und ihren Eigenschaften und Widerständen, ihrer Physis, wichtige Beiträge zum Entwurfsprozess. Sie könnte nicht vollständig rationalisierbare Gestaltungsprozesse ergänzen, dazu muss jedoch der Raum zwischen virtueller und physischer Realität überbrückt werden.

In Hinblick darauf sind Vermittler zwischen beiden Welten vonnöten, also Aktuatoren, vom virtuellen ins physische, und Sensoren vom physischen ins virtuelle.

Sind physische und virtuelle Realitäten ineinander verschränkt, werden physische Objekte zu Vertretern immaterieller Information und dienen gleichzeitig als Informationsquellen für ihr virtuelles Pendant.

Werden in diesem verschränkten System Regeln und Prozesse definiert und nicht Formen festgeschrieben, so kann durch Iteration Interaktion und Feedback ermöglicht werden.

Die beschriebenen Methoden, obwohl sie sich in Farben und Formen manifestieren, sind nicht an einen Maßstab gebunden,

auch nicht an Materialien, vielmehr an das Prinzip des Feedbacks, in regelmäßigen Abständen gemessene Ist-Zustände in die Berechnungen miteinfließen zu lassen.

Im Rahmen der Arbeit wird in Form von Experimenten herausgearbeitet, welche Qualitäten und Möglichkeiten dieser Ansatz bietet.

2. 2 Rationalisierung und Quantifizierung

In einer völlig objektivierten Situation sind Lösungen nur mehr vom Problem und nicht mehr von dem/der Gestalter/in abhängig, was voraussetzen würde, dass man ein Problem ganzheitlich erfassen kann.

In der Realität existieren solche Situationen jedoch nicht, wie Reinhard König ausführt: „Every rational approach - even in the sciences - is only rational to a limited degree. [...] Ultimately, the definition of a design rule or an entire system of rules is always based on heuristic knowledge, on a rule of thumb, and not on a purely analytical consideration of the problem at hand. The only reasonable approach is, therefore, to consciously apply a combination of intuitive and rational design strategies.“[1]

Daraus folgend kann man entnehmen, dass man in der Anwendung algorithmischer Methoden in Gestaltungsprozessen operationale Probleme automatisieren, und nicht vollständig operationale Probleme durch die Kombination menschlicher Intelligenz und Computer ermöglichen sollte.

2. 3 Interaktives Bedienen computergesteuerter Maschinen

„One might ask whether and why architects should use industrial robots or even computer programming, tools that can appear architecturally irrelevant. In our opinion it is crucial that architects, now and in the future, choose their means consciously

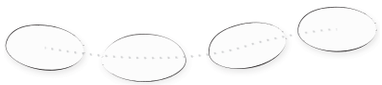
1 König, Reinhard (2011) Generative Planning Methods from a Structuralist Perspective. in: Structuralism Reloaded. Edition Axel Menges. S. 278

and master their tools. Accessing these generic tools enables architects to create their individual design instruments and thus generates diverse forms of expression.“[2]

2 Gramazio & Kohler, (2008) Digital Materialilty in Architecture. Lars Müller Publishers. Baden, Switzerland. S.9

Innerhalb unseres Verfahrens werden die CNC-Maschinen nicht nur als Fertigungsautomaten sondern als Gestaltungswerkzeuge begriffen, die von dem/der Gestalter/in während des Herstellungsprozesses interaktiv gesteuert werden können.

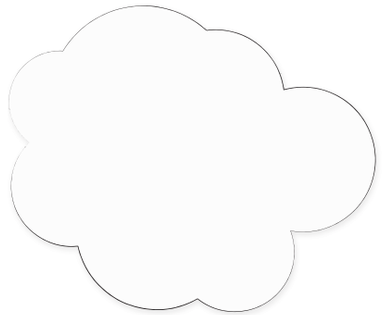
Das bedeutet, dass die Vorgänge, beginnend bei den Formfindungsprozessen bis hin zur Architekturproduktion und den Fertigungsprozessen, nicht nur digital gestaltet sind, und Informationen von einer Maschine auf die nächste übertragen werden, sondern dazu ist es notwendig, diese sogenannte „digitale Kette“ aufzulösen.



Digital Chain

Schon der etablierte „File to Factory“ Prozess veränderte die Rolle des architektonischen Planmaterials. Die Repräsentation oder der Plan wird von der Maschine und nicht mehr vom Mensch interpretiert, die digitalen Modelle werden direkt an den Produktionsprozess gebunden und die digitale Kette wird nicht unterbrochen.

Damit werden allerdings nach wie vor der Designprozess als auch der Produktionsprozess isoliert voneinander behandelt, und werden jeweils nacheinander bearbeitet. In der Umsetzung oder Übersetzung vom virtuellen Modell zu einem physischen Modell oder Prototyp müssen die Produktionspfade im Vorhinein erstellt werden, d.h alle Parameter müssen fixiert und alle gestalterischen Entscheidungen bereits getroffen sein.



Digital Cloud

Die Frage stellt sich, welche Möglichkeiten eröffnet werden, wenn man nicht in einzelnen Prozessschritten denkt, sondern der ganze Prozess vom Entwurf bis zur Produktion eine Einheit bildet.

Man entwirft dann nicht mehr die Form, die später wieder in Teile zerlegt und für die Produktion übersetzt werden muss, sondern entwirft die Prozesse, in denen die Produktion stattfinden kann.[3]

3 vgl. 2

2. 4 Synthesis of Form – Programm als DNA

Zieht man den Vergleich von einem Code zum Generieren architektonischer Struktur zum DNA-Code einer Pflanze, muss man anmerken, dass sich die Gestalt der Pflanze nicht nur von der DNA ableiten lässt. Die DNA bestimmt zwar den Prozess der Entstehung aber durch die Umwelt und deren Einflüsse verändert sich die Ausbildung der Geometrie der Pflanze. Die DNA ist lediglich eine Bauanleitung, die in ihrer Entfaltung jedoch noch viele Optionen und Varianten offen lässt.

Christopher Alexanders beschreibt bei seiner Theorie der Morphogenese in „Notes on the Synthesis of Form“[4], dass ein Programm als Genotyp verstanden werden kann, und die einzelnen realisierten archetypischen Lösungen davon als Phänotypen.

“Die Schönheit, die ich meine, wenn ich sage, dass ich die Welt wieder schön haben möchte, hat mit unendlicher Vielfalt zu tun. Die Natur der Dinge, die geschaffen werden, unterscheidet sich in Abhängigkeit von den verschiedenen Situationen und Orten. Sie hängt auch davon ab, welche Hände sie geformt haben. Dennoch kommt deren Kraft aus der Eleganz des generativen Codes, der sie entstehen lässt. Ich denke nicht, dass das ein Widerspruch ist.” [5]

Die Gestalt in ihrer finalen Form als Resultat eines algorithmischen Prozesses ist nur eine der vielen der Möglichkeiten, die der Code in sich birgen würde, aber ergibt sich aus genau den Umständen, die im Entstehungsprozess vorgefunden werden, somit auch gestalterischen Interventionen im Entstehungsprozess der Form, wo Richtungen abgeändert oder Korrekturen vorgenommen werden können.

2. 5 Ein angeregter Austausch

„Ein Haus ist ein Haus ist ein Haus. Aber das Medium, in dem sich Architektur ausdrückt, hat sich verändert, von der vorindustriellen Manufakturarbeit über die maschinelle Produktion des Industriezeitalters bis zu den Kommunikationsprozessen des heutigen Informationszeitalters. [...] Heute stellen Codes den Rahmen und den Boden bereit, auf dem die Kommunikation, das Planen, die Finanzierungen und das Konstruieren ablaufen [...] Mit Hilfe der Informationstechnologie lassen sich Netzwerke, Strukturen wechselseitiger Abhängigkeiten technisch abbilden und in Balance halten, Informationstechnik funktioniert nicht nach dem gängigen strikten Ursache-Wirkungsprozess, wie er aus der Mechanik bekannt ist, sondern nach den Regeln der Kommunikation, des wechselseitigen, angeregten Austauschs.“[6]

4 Alexander, Christopher. (1964) Notes on the Synthesis of Form. Cambridge, Massachusetts, London: Harvard University Press

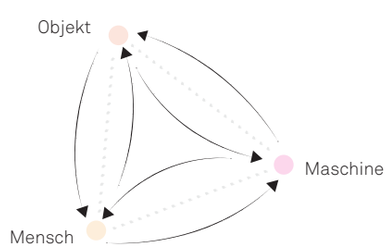
5 Herresthal, Kristina: „Von fließender Systematik und generativen Prozessen - Christopher Alexander im Gespräch mit Rem Koolhaas und Hans Ulrich Obrist“, in: ARCH+ Ausgabe Nr. 189, Zeitschrift für Architektur und Städtebau, Oktober 2008. S.23

6 Hovestadt, Ludger. (2010) Jenseits des Rasters – Architektur und Informationstechnologie. Birkhäuser Verlag AG Basel . Boston . Berlin S.16

2. 6 Methodik

Die Wechselbeziehungen zwischen Mensch, Objekt und Maschine formen einen dynamischen Prozess: Die Gestaltenden formulieren ein Programm, worin Algorithmen definiert werden, nach deren Regeln einzelne Elemente angeordnet und untereinander in Beziehung gesetzt werden, und so zwei- oder dreidimensionale Strukturen erzeugen.

Das Programm ist aus geeigneten Randbedingungen aufgebaut, lässt aber Parameter offen, was den Gestaltenden ermöglicht, in den laufenden Prozess einzugreifen und diese Parameter zu verändern. Die Strukturen werden auf Grund verschiedener Eingaben und in Abhängigkeit der von dem/der Gestalter/in definierten Parametern aufgebaut und umgebaut.



Wechselbeziehung zwischen Gestalter/in, Objekt und Maschine

3 Definitionen

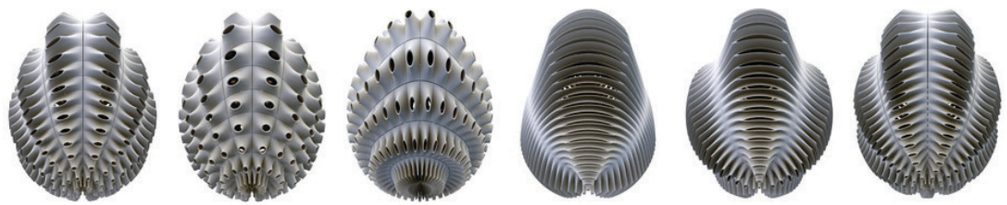
3 Definitionen

3.1 Parametrisches oder algorithmisches Entwerfen

Die folgenden Erklärungen sind der Versuch einer Differenzierung der beiden Begrifflichkeiten:

Parametrisches Entwerfen

Beim parametrischen Modellieren werden verschiedene Objekte, die man modelliert, miteinander in Beziehung gesetzt oder in Abhängigkeit gebracht. Zum Beispiel wird eine dreidimensionale Fläche von einer Leitkurve abhängig gemacht, durch die sie erzeugt wird. Wenn man diese Leitkurve verändert, dann ändert sich die Fläche anhand der neuen Kurve mit.

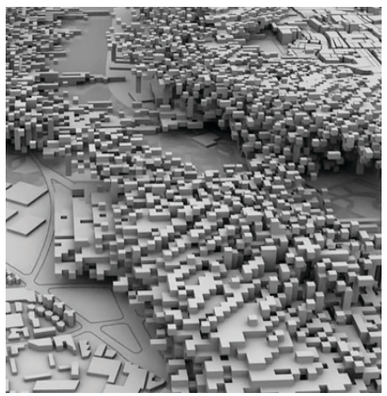


Cross Scalar Variation II
Rocker-Lange Architects. 2011

Natürlich passiert mit den Eingabeparametern (im Beispiel die Kurve) nichts ohne einen Algorithmus, der aus ihnen eine Ausgabe errechnet (im Beispiel die Fläche) Allerdings ist der Algorithmus sehr einfach, es gibt keine komplexen Beziehungen und daher ist das Ergebnis leicht aus den Parametern vorhersagbar. Das System ist schnell zu überblicken und die Designintelligenz der Gestaltenden steckt in der Wahl einer „guten" Kurve, d.h. in der Wahl guter Parameter.

Algorithmisches Entwerfen

Beim algorithmischen Entwerfen gibt es auch Eingabeparameter und einen Algorithmus, der aus ihnen eine Ausgabe berechnet. Nur sind hier die Parameter quasi fix, sie sind gemessene, wahre Größen (Stadtplan, Topografie, andere User-Eingaben, etc.). Die Designentscheidung bezieht sich auf die Gestaltung des Algorithmus, nach dem aus den Eingaben eine Ausgabe



Behavioral Urbanism
http://www.kokkugia.com/

erzeugt wird, allerdings ist das Ergebnis nicht mehr unbedingt vorhersehbar als auch nachvollziehbar, das System ist schwieriger zu durchblicken, da durch einfache Regeln komplexe Gebilde erzeugt werden können.

3. 2 Randbedingungen

In der Entwurfsphase ist es wichtig, die Randbedingungen (Constraints) klar zu definieren, denn sie sind es, durch die ein System entwickelt wird, oder durch die die Anforderungen an das System umschrieben werden.

Randbedingungen können geometrischer Natur sein, also die Form definieren, es können funktionale Bedingungen oder materialspezifische Eigenschaften sein, sie können durch die verwendeten Technologien bestimmt werden, oder verfügbare Ressourcen, etc.

Randbedingungen bedeuten nicht nur Einschränkungen, man kann sie auch als treibende Kraft der Kreativität sehen.[1] Sind keine oder nur wenige Randbedingungen vorhanden, dann gibt es zu viele Freiheiten und es ist schwer, eine Richtung zu finden.

Das Erkennen und Definieren der Randbedingungen ist ausschlaggebend für die weitere Lösungsfindung.

3. 3 Interaktives System

Ein interaktives System reagiert auf externe (oder interne) Vorfälle, indem es seine Benutzer und Anwender mit einer Ausgangsgröße versorgt. Die Ausgangsgröße kann dabei ein Feedback zum Vorfall sein, oder Rückschlüsse auf den Zustand des Systems geben.

1 vgl. Kilian, Axel (2006). Design Exploration through Bidirectional Modeling of Constraints. S.59

3. 4 Selbstorganisierende Systeme

Prozess der Kommunikation

Das selbstorganisierende System besteht aus einzelnen Elementen, Individuen, die über die Kommunikation untereinander gestaltende und beschränkende Einflüsse auf das Gesamtsystem ausüben.

Die sich permanent wechselnden Beziehungen der Elemente führen zu einer Komplexität des Gesamtsystems, dessen Verhalten sich dadurch schwer beschreiben oder vorhersehen lässt.

Das System bleibt trotz des klar definierten Aufbaus flexibel und beweglich und gleichzeitig stabil. Stabil in jener Hinsicht, dass es sich als eine funktionierende Einheit darstellt, und sich Fehlfunktionen von Individuen nicht fatal auf das Gesamtsystem auswirken. Der Vorteil der Flexibilität selbstorganisierender Systeme ist ihre Fähigkeit, sich an wechselnde Umgebungsbedingungen anpassen zu können, ohne dass sich die ihr zugrunde liegenden inneren Ordnungssysteme ändern müssen.

Wellen im Sand: ein Beispiel für ein natürliches selbstorganisiertes System
http://www.flickr.com/photos/nuytsia_pix/2612265575

3.5 Emergente Systeme

Die Gestaltung des Unvorhersehbaren

Emergenten Systemen wird zugesprochen, dass sich bestimmte Eigenschaften eines Ganzen nicht aus den Teilen erklären lassen.

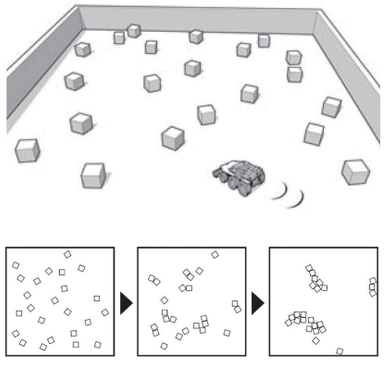
"Zu den spannendsten Phänomenen vernetzter, interaktiver Systeme gehört die Emergenz. [...] Das Ganze ist mehr als die Summe seiner Teile, weil die dynamische Interaktion der Teile ein Geschehen oder eine Struktur hervorbringt, die in den einfachen Regeln der Interaktionen nicht einmal zu erahnen sind. Emergente Systeme sind im wahrsten Sinne des Wortes erfinderisch. [...]" [2]

2 Hovestadt, Ludger. (2010) Jenseits des Rasters – Architektur und Informationstechnologie. Birkhäuser Verlag AG Basel . Boston . Berlin S. 81

Beispiele für Emergenz:

1. Das komplexe System eines Verkehrsflusses, worin sich Agenten (Autofahrer) unter Beachtung der Verkehrsregeln frei bewegen: Durch deren individuellen Entscheidungen, wie schnell zu fahren, welchen Weg zu nehmen und wo zu halten, ergeben sich Muster und Strukturen.

2. Die "Swiss Robots" von Rene te Boekhorst und Marinus Maris: Jeder der Swiss Robots ist mit jeweils zwei Motoren und zwei Infrarot-Sensoren ausgestattet, die den Abstand zu Objekten messen können. Im Experiment werden einige Roboter zusammen mit Styropor-Würfeln in der Arena verteilt, und es passiert, dass es aus der Sicht des Beobachters erscheint, als würden die Roboter aufräumen zu beginnen, sie häufen die Styropor-Würfel zu Clustern an. Aus der Sicht der Roboter jedoch folgt dieser nur einfachen Regeln, z.B. dass er ausweicht, wenn ihn ein Sensorsignal erreicht, und weiß nichts von den Anhäufungen, die durch seine Handlungen erzeugt werden. [3]



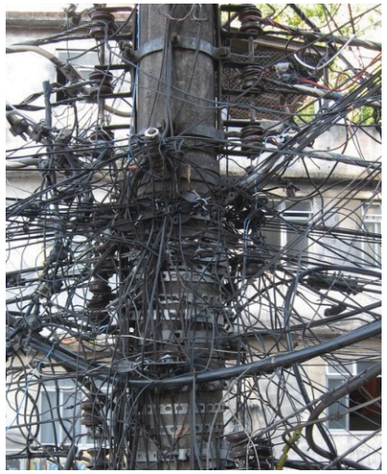
„Swiss Robots." Arena with randomly distributed Styrofoam cubes and sequence of images showing the clustering procedure.

3 vgl. Pfeifer, Rolf / Bongard, Josh. (2009) How the Body Shapes the Way We Think. A New View of Intelligence. London, Cambridge: The MIT Press, S. 72–74

„Top Down" vs „Bottom Up"

Eine architektonische Kompositionen, als ein Ganzes, besteht aus einzelnen Elementen, die miteinander in mehr oder weniger komplexen Beziehungen stehen, und nicht immer offensichtlich sind. Die Elemente alleine, isoliert für sich, haben keine Bedeutung, diese erlangen sie erst im Zusammenspiel als System. Man kann annehmen, dass die Gestalter, wenn sie ein Gebäude, eine Form, entwerfen, Elemente arrangieren, Systeme gestalten, bis sie ihren ästhetischen und funktionalen Kriterien entsprechen, und damit der Gestaltungsprozess, zumindest im Ansatz, ein emergentes System an sich darstellt.

Bei algorithmischen Formfindungsprozessen sind nicht die Eigenschaften der Elemente vorrangig, sondern ihre Beziehung zueinander, und wie sie miteinander kommunizieren. Diese Beziehungen müssen bis zu einem gewissen Grad abstrahiert werden, und es müssen Regeln auf lokaler Ebene gesucht werden, die zum erwünschten globalen Ziel führen können. Der folgende Auszug aus „How the Body Shapes the Way We Think" von Rolf Pfeifer und Josh Bongard fasst diese Prinzipien in vier Punkten zusammen:



Strommast in São Paulo
http://www.flickr.com/photos/71377087@N00/504409753/

Design principles for collective systems:

Level of abstraction: Proper level of abstraction must be chosen, and the implications must be clearly kept in mind

Design for emergence: Find local rules of interaction that lead to desired global behavior patterns

From agent to group: Agent design principles can often be applied to collective systems

Homogeneity - heterogeneity: A compromise has to be found between systems using only one type of module or robot, and employing several specialized types. [4]

4 Pfeifer, Rolf / Bongard, Josh. (2009) How the Body Shapes the Way We Think A New View of Intelligence. London, Cambridge: The MIT Press, S. 358

3.6 Kommunikation

Im 1948 veröffentlichten Artikel „A Mathematical Theory of Communication" von Claude E. Shannon werden drei Level von Kommunikationsproblemen beschrieben. Das technische Problem bezieht sich darauf, wie akkurat Symbole transportiert

werden können, das semantische Problem befasst sich damit, wie die übertragenen Symbole der gewünschten Bedeutung entsprechen und das Effizienzproblem behandelt die Interpretation der empfangenen Bedeutung. [5]

Shannon beschreibt, dass es notwendig ist, zusätzliche redundante (d. h. keine zusätzliche Information tragenden) Daten mitzusenden, um dem Empfänger eine Datenverifikation oder Datenkorrektur zu ermöglichen. Daten müssen vom Sender codiert werden, und vom Empfänger entcodiert, damit die Information korrekt interpretiert werden kann.

3.7 Daten und Information

Der Unterschied zwischen Daten und Information

Information ist eine nutzbare Antwort auf eine konkrete Fragestellung, sie wird meistens aus Daten extrahiert oder abgeleitet.[6]

Von Information wird gesprochen, wenn auf eine spezifische Frage eine Antwort gegeben wird, die das Verständnisniveau der Fragenden erhöht und sie befähigt, einem bestimmten Ziel näher zu kommen.

Nimmt man z.B. ein Buch über Astronomie als Datenmenge her, so ist der Informationsgehalt für jemanden, der sich in dieser Materie auskennt aufgrund seiner Erfahrung und seines Wissens größer wie für einen Nicht-Astrologen.

Angesichts der Datenmenge im Internet, benötigen wir eine Suchmaschine, wenn wir an eine bestimmte Information gelangen wollen. Google verwendet einen Suchalgorithmus mit unzähligen Filtern und Kriterien, nach denen die Webseiten bewertet und sortiert werden.

Globale vs. lokale Information

Betrachtet man eine agentenbasierendes System in der Simulation, gibt es immer zwei verschiedene Betrachtungsebenen:

5 vgl. Shannon, Claude E. (1948). A Mathematical Theory of Communication. Bell System Technical Journal Vol. 27: S. 379–423.

6 vgl. Zehnder, C.A. (1998). Informationssysteme und Datenbanken. Zürich: vdf-Hochschulverlag AG.

eine ortsbezogene/lokale, in der die einzelnen Individuen ihre Information aus ihrer unmittelbaren Umgebung beziehen, welche zur Berechnung ihres Verhaltens herangezogen wird, und eine allgemeine/globale Betrachtungsebene, in der die Programmierer globale Informationen verwenden können, zur Berechnung des Verhaltens der Agenten.

Sobald man jedoch anfängt, die Simulation in die reale Welt umzusetzen, also Roboter für die Agenten einzusetzen, gibt es diese globale Information für die Agenten nicht mehr. Es sei denn, man installiert Kameras, die die globale Position der Roboter tracken können, oder man stattet sie mit GPS aus. Dadurch verkompliziert sich jedoch das gesamte Setup und man findet sich bald in einem Kampf gegen Windmühlen wieder.

Deswegen werden im Bereich der „embedded robotics“ die Agenten mit Sensoren ausgestattet, damit sie ihre Informationen aus der Umgebung beziehen können, worauf dann die Berechnungen für ihr Verhalten basieren.

Die Umgebung als Information

Aus physischen Faktoren der Umgebung wird Information extrahiert, ein Beispiel hierfür ist der Roomba® Staubsauger von iRobot.

Das Ziel des Staubsauger-Roboters ist die maximale Erfassung jedes Raumes, also die Reinigung der größtmöglichen Fläche, der Roboter weiß jedoch nicht, wie der Raum beschaffen ist oder welche Hindernisse auf seinem Weg liegen.

Dieser bezieht die Information nun aus seiner Umgebung, wenn er sie braucht. In diesem Fall fährt der Roboter so lange, bis er an etwas stoßt, dann dreht er sich in einem bestimmten Winkel um und fährt weiter (vgl. Swiss Robots). Dieses Beispiel zeigt auf, dass die Information für eine komplizierte globale Konfiguration ad hoc aus der Umgebung abgerufen werden kann, um einfaches lokales Verhalten zu beeinflussen.

iRobot Roomba®
http://www.irobotroomba-info.com/

4 Zugänge

4 Zugänge

4. 1 Zugänge

Während im Kapitel „Definitionen“ relevante Begriffe für die Arbeit erklärt werden, werden hier Erweiterungen in verschiedene Themenbereiche geboten, die zur Richtung und Entwicklung der Arbeit beigetragen haben.

4. 2 Intuition

„Als ich dazu kam zu unterrichten, musste ich mir genau klar werden über das, was ich – meist unbewusst – tat.“ [1]

1 Zitat von Paul Klee 1921 beim Antritt seiner Lehrtätigkeit am Bauhaus, Weimar

Während des intuitiven Vorgehens beim Entwerfen von Architektur, beispielsweise beim Erstellen einer architektonische Skizze, werden Entscheidungen getroffen, die mit dem Verstand oft nicht erklärbar sind. Dieses „Bauchgefühl“ kann nicht wirklich beschrieben oder definiert werden, aber Designentscheidungen werden nicht willkürlich getroffen, sondern werden unbewusst von einem inneren Erfahrungsreichtum geleitet.

„ […] Dennoch gibt es eine Grundbedeutung: intuitiv denken heißt in der Dauer denken. Der Verstand geht gewöhnlich aus vom Unbewegten und rekonstruiert recht und schlecht die Bewegung durch nebeneinander gesetzte unbewegliche Punkte. Die Intuition geht von der Bewegung aus, setzt sie oder vielmehr erfasst sie als die Wirklichkeit selber und sieht in der Unbeweglichkeit nur eine Abstraktion, gleichsam eine Momentaufnahme unseres Geistes. […] Für die Intuition ist die Veränderung das Wesentliche: was das Ding angeht, wie es der Verstand auffasst, so ist es nur ein Querschnitt im Fluss des Werdens, den unser Geist als Ersatz für das Ganze genommen hat. Der Gedanke stellt sich gewöhnlich das Neue vor als eine neue Anordnung von vorher existierenden Bestandteilen, für ihn geht nichts verloren, entsteht aber auch nichts Neues. Intuition, mit einer Dauer verbunden, bedeutet inneres Wachstum, sie gewahrt in ihr eine ununterbrochene Kontinuität von unvorhersehbarer Neuheit, sie sieht, sie weiß, dass der Geist über sich selbst hinaus zu wachsen vermag,

dass die wahre Geistigkeit gerade darin besteht, und dass die vom Geist durchdrungene Wirklichkeit Schöpfung ist. [...]

Ob es sich um standesmäßiges Denken oder Intuition handelt, immer benutzt das Denken ohne Zweifel die Sprache, und wie jedes Denken wird sich die Intuition schließlich auch in Begriffen niederschlagen: Dauer, qualitative oder heterogene Mannigfaltigkeit, Unterbewusstsein, [...] Aber der Begriff, der verstandesmäßigen Ursprungs ist, ist ohne Weiteres klar, zum Mindesten für den Geist, der eine genügende Anstrengung machen könnte, während die Idee, die aus der Intuition hervorgeht, gewöhnlich im Anfang dunkel ist, welches auch unsere Anstrengung des Denkens sein möge. Es gibt nämlich zwei Arten von Klarheit.

Eine neue Idee kann klar sein, weil sie uns elementare Ideen, die wir schon besaßen, lediglich in einer neuen Anordnung darbietet. Unser Verstand, der dann in dem Neuen nur Altgewohntes findet, fühlt sich in bekannten Gefilden, er fühlt sich zu Hause, er „begreift". [...] Es gibt aber noch eine andere Klarheit, die erst allmählich durch fortschreitende Vertiefung in den Gegenstand gewonnen wird. Es ist diejenige, welche der radikal neuen und unreduzierbaren Idee innewohnt, der mehr oder weniger eine Intuition zugrunde liegt. Da wir sie nicht mit vorher existierenden Elementen aufbauen können, da sie keine Elemente hat, und da andererseits das mühelose Begreifen darin besteht, Neues aus Altem zusammenzusetzen, geht unsere erste Regung dahin, sie für unverständlich zu erklären."[2]

Henri Bergson beschreibt, dass Intuition nur „während" einer Tätigkeit, sei diese eine handwerkliche oder geistige, passieren kann, und somit in der Zeit eingebettet ist. Bei allen kreativen Tätigkeiten entsteht bei der Arbeit mit dem Werkzeug, z.B. mit dem Pinsel oder Meißel, mit der Zeit eine Eigendynamik, bei welcher nicht mehr nur der Kopf den nächsten Pinselstrich oder die nächste Schnitzkerbe plant, sondern sozusagen das Werkzeug, als Erweiterung der Hand, weiß, was es tut.

Begreifen wir die CNC-Maschinen auch als Gestaltungswerkzeuge, sollte deren Gebrauch für den/die Gestalter/in auch intuitiv zugänglich gemacht werden. Dabei werden sie aber keinesfalls nur als die Verlängerung des menschlichen Armes

2 Bergson, Henri. (1993) Denken und schöpferisches Werden. Hamburg: eva-Taschenbuch S.46-48 (Originalausgabe: Bergson, Henri. (1946) La pensée et le mouvant. Paris: Presses Universitaires de France)

verstanden, oder als Ersatz der menschlichen Hand, sondern es bieten sich viele Möglichkeiten der Gestaltung eines komplexeren Werkzeuges, da in ihm digitale Prozesse eingeschrieben werden können. Somit soll ein Zusammenspiel entstehen, zwischen den Eigenschaften, die digitalen Systemen zugrunde liegen, und den menschlichen Fähigkeiten.

Es soll ein halboffenes System geschaffen werden, bei dem der/die Benutzer/in das Werkzeug und die Kombination aus Eingaben und deren Übersetzung in eine Gestalt als Gesamtheit erfährt und darauf reagieren lernt.

4. 3 Der Zufall

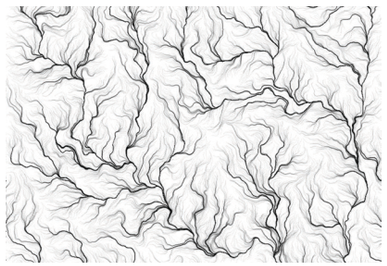
Das Stellen der richtigen Fragen

„Die meisten Leute, die glauben, ich sei am Zufall interessiert, begreifen nicht, dass ich Zufall als eine Methode benutze, man denkt im Allgemeinen, ich benutze den Zufall als eine Möglichkeit, mich einer Entscheidung zu entziehen, aber meine Entscheidungen bestehen darin, welche Fragen überhaupt gestellt werden."[3]

Wie John Cage es in diesem Zitat formuliert, geht es ihm bei der Verwendung des Zufalls nicht darum, Entscheidungen auszuweichen, sondern die Entscheidungen passieren vorher, beim Stellen der richtigen Fragen. Wenn er dann den Zufall als Methode benützt, dann nur an von ihm klar definerten Stellen. Das bedeutet nicht, dass seine Kompositionen dadurch einer Beliebigkeit unterworfen sind.

Im Designprozess verhält es sich ähnlich. Man muss die richtigen Fragen stellen, die Randbedingungen und ihre Beziehungen zueinander definieren, und Parameter, die variabel bleiben sollen, sinnvoll einsetzen. Bei der Planung unseres Systems geht es darum, dem Benutzer die Möglichkeit zu bieten, in den Gestaltungsprozess mit CNC-Maschinen interaktiv einzugreifen. Dabei gilt es vorab folgende Fragen zu klären: Von welcher Art ist die Benutzereingabe? Wie wird sie übersetzt? Welche Parameter sind während der Laufzeit veränderbar?

3 Zitat von John Cage in Ö1 Hörbilder Spezial vom 1.11.2011, "32 Betrachtungen über Grete Sultan", Jean-Claude Kuner



Perlin Noise
http://flashyprogramming.wordpress.com/2009/12/26/visualizing-perlin-noise/

4. 4 Architektonische Ordnung

Strukturalismus

Ein Bauwerk, das Ordnungsprinzipien folgend, aus einzelnen Teilen besteht, und dessen einzelnen Teile das Ganze bestimmen, aber wiederum sich logisch vom Ganzen ableiten lassen, bildet eine Einheit. Diese Einheit wird als Struktur bezeichnet. [4] Der Strukturalismus beschreibt einen logischen Vorrang des Ganzen gegenüber den Teilen und versucht einen internen Zusammenhang von Phänomenen als Struktur zu fassen.[5]

Wenn der Entwurf auf einer logischen Strategie beruht, und das Gebäude auch wiederholt von den einzelnen Komponenten aus überprüft wird, und es damit zu einer Korrektur des Ganzen (des Themas) kommt, kann man von einer Arbeitsweise sprechen, welche durch Rückkoppelung ein ordnendes Prinzip erreicht, „[...] und somit eine Struktur erstellt, die programmiert ist, um alle gegebenen Situationen aufnehmen zu können."[6]

Ein bekanntes Beispiel für den Strukturalismus in der Architektur ist das Waisenhaus in Amsterdam von Aldo van Eyck, 1955-60, dessen Komplex von Straßen und Plätzen und eigenständigen Baukörpern in seinen Ausformulierungen den Kosmos einer Stadt widerspiegelt: "Eine kleine Welt in der großen, und die große Welt in der kleinen, ein Haus als eine Stadt und eine Stadt als ein Haus, ein Heim für Kinder zu schaffen, das war mein Ziel."

„Neostrukturalismus"

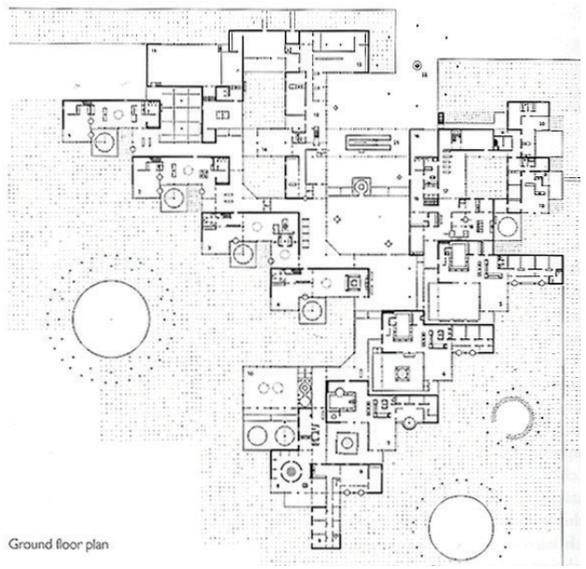
"[...] eine der Grundannahmen im Strukturalismus lautet, dass Objekte nicht essentialistisch zu verstehen sind, sondern erst in ihrer Einordnung in Strukturen überhaupt bestehen. Mit der Digitalisierung der Architektur, die dadurch relational und parametrisch erfassbar wird, liegen die technischen Voraussetzungen für eine strukturale Lesart der Architektur vor, nur diesmal nicht mehr sprachanalytisch, sondern geometrisch begründet. [...]"[7]

4 vgl. Hertzberger, Hermann. (1995) Vom Bauen. Vorlesungen über Architektur. München: Aries Verlag, S. 122

5 Hans-Dieter Gondek, Strukturalismus, Artikel in: Hans Jörg Sandkühler (Hrsg.), Enzyklopädie Philosophie, Bd. 2, Hamburg 1999, S. 1542

6 Hertzberger, Hermann. (1995) Vom Bauen. Vorlesungen über Architektur. München: Aries Verlag, S. 122

7 Kuhnert, Nikolaus und Nho, Anh-Linh: „Entwurfsmuster", in: ARCH+ Ausgabe Nr. 189, Zeitschrift für Architektur und Städtebau, Oktober 2008. S.8

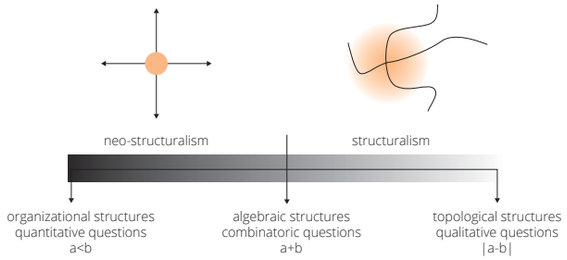


Amsterdam Waisenhaus, Amsterdam, Holland, 1955 bis 1960. Strauven, F., „Aldo van Eyck, The Shape of Relativity", Architectura & Natura 1998

Den neostrukturalistischen Strömungen wird vorgeworfen, aufgrund ihrer Reduktion auf geometrische Ordnungsprinzipien dem Strukturalismusgedanken der 60-er Jahre nicht gerecht zu werden, der inneren Beziehungen folgend Freiheiten in der architektonischen Entfaltung lässt, während sich der Neostrukturalismus einem konkreten Design-Vokabular unterwirft.

Toni Kotnik sieht den großen Unterschied zwischen dem Strukturalismus der 60er Jahre und den sogenannten neostrukturalistischen Strömungen darin, dass das Arbeiten am Computer die Notwendigkeit birgt, in kausalen Zusammenhängen zu denken, also eine genaue Spezifikation der Designgedanken und eine Quantifizierung dieser fordert, während der Strukturalismus sich durch ein topologisches und qualitatives Denken auszeichnet, welches seine Spuren in einer Architektur hinterlässt, die in ihrer physischen Präsenz vage und offen bleibt. [8]

8 vgl. Kotnik, Toni. (2009) Alogorithmic Design. Structuralism Reloaded? in: Structuralism Reloaded. Edition Axel Menges .S. 327-335



„Strukturen"
Kotnik, Toni. (2009) Alogorithmic Design. Structuralism Reloaded? in: Structuralism Reloaded. Edition Axel Menges. S. 333

4. 5 Mensch und Maschine

Operationale und nicht operationale Probleme

Der Entwurfsprozess lässt sich nicht völlig rationalisieren und automatisieren, es gibt Probleme, die nicht völlig operationalisierbar sind, und besser intuitiv lösbar sind. Damit sind die einzigen sinnvollen Modelle automatisierer Methoden diese, die menschliche Eingriffe ermöglichen, und somit zu einem Miteinander von Mensch und Maschine animieren.

„Ein flammender Handwerker ist in seiner Erregung bereit, die Kontrolle über seine Arbeit verlieren, und stolpert so über glückliche Zufälle, die zu großen Erfindungen führen können. Maschinen hingegen, bei Verlust ihrer Kontrolle, sind einfach nur defekt." [9]

Zum Vergleich dazu ein Auszug aus einer Publikation für automatisierte Musik-Komposition[10], in dem dargestellt wird, wie flexibel und offen automatisierte Prozesse aufgebaut sein müssen, damit dass Ergebnis nicht bedeutungslos wird:

„Imitating and not creating boring and predictable results

Many AMC applications seem to be only good at boring, imitations of existing compositions. Many simply provide variations on existing works, rather than generating new and creative works. This problem can be attributed to the over reliance on rule-based strategies and too conventional implementation techniques.

9 Senett, Richard (2008): Handwerk. Berlin. Berlin Verlag, S. 135f.

10 Senaratna, N. I. (2006). Automatic Music Composition Using a Tree of Interacting Emergent Systems. Degree of Bachelor of Computer Science (Special) of the University of Colombo, S.17

Boring and predictable results

Another bad result of over reliance on rule-based strategies and too conventional implementation techniques is that the resultant output is highly predictable. Unpredictability is an essential part of creativity.

Lack of control over the nature of the output

At the other extreme of things, techniques using „pure" emergent techniques (for example pure fractal based melodic composition or purely cellular automata based music composition), the resultant musical output tends to be ad hoc and meaningless. The user (or initiator) of the process has little control results. In practical situations this is a serious problem; what is the use of an AMC system if it cannot produce what the user wants?"

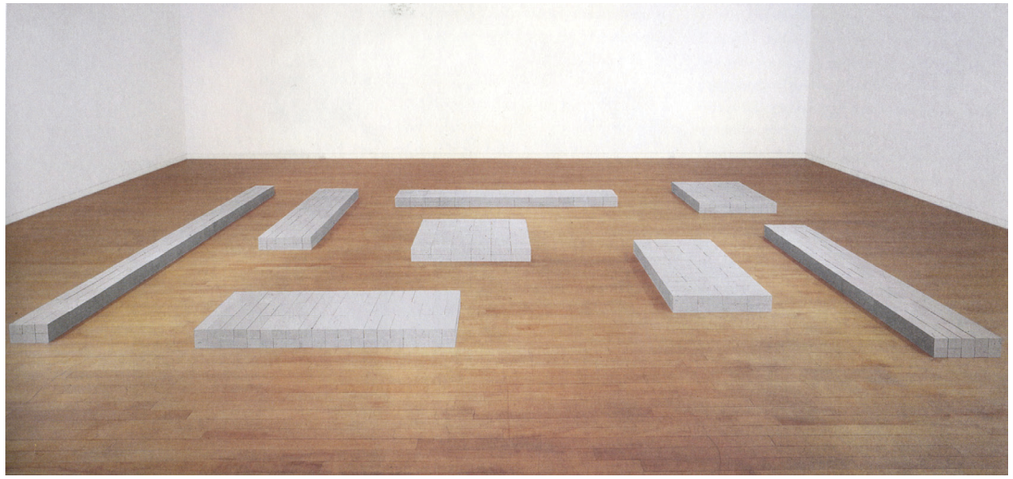
Nam June Paik, „Robot K-456"
Chance in a lifetime, von: John G. Hanhardt on Nam June Paik ArtForum, April, 2006 by John G. Hanhardt

4. 6 Entstehung der Form

Ordnung im Chaos - Minimal Art

„I have found a set of solutions to a set of problems in sculpture, and I work within those parameters, but it is limits that give us possibilities. Without limits nothing really good can be accomplished. I feel I've been liberated by them." Carl Andre [11]

11 Rider, Alistair. (2011) Carl Andre. Things In Their Elements. London: PHAIDON PRESS



Sand-Lime Instar. Gagosian Gallery New York 1995. Sand lime brick, eight 120-unit rectangluar solids, 2 units high. 448.9 x 726.4cm

In der minimalen Skulptur wird versucht, nichts mehr als eine Untersuchung der Materialeigenschaften und -formen einzubinden. Es wird mit Materialien aus der Industrie gearbeitet, die so neutral wie möglich in ihrer Erscheinungsform sind und die keinerlei Assoziationen hervorrufen sollen.

Carl Andre's Arrangements seiner designierten Bauelemente[12] bauen auf orthogonalen Rastern auf, und wie es oft bei mathematische Modellen ist, sind die physischen Manifestationen von Konzepten und Theorien oft ästhetische Objekte an sich. Trotzdem betont Carl Andre, dass sich „unter den Platten keine Ideen verstecken, das es trotzdem bloß Platten wären."

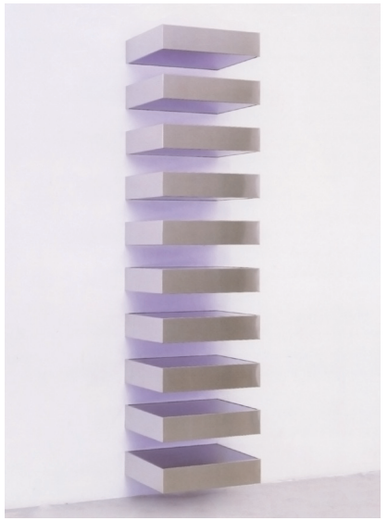
12 vgl. http://www.martinries.com/article1991CA.htm

Ein wichtiger Aspekt der Minimal Art ist, dass gängige Vorstellungen von Komposition ersetzt werden durch strukturierte Sequenzen, die viel systematischer und routinierter sind, die

die Methode der Anordnung oder Konstruktion in den Vordergrund rücken. Einzelteile werden nach Regeln angeordnet, das „Wie" des Anordnens wird automatisiert durch das Akzeptieren der Methode, und der seriellen Wiederholung.

Während die traditionelleren Methoden der abstrakten Kunst vom europäischen Kubismus das Augenmerk auf die Balance der Gestalten, auf eine ausgewogene Beziehung zwischen der Einzelteile und dem Ganzen erben, gibt es im Minimalismus nach den Worten von Donald Judd nur „ein Ding nach dem anderen", also nur die Beziehung von einem Teil zum nächsten. Und das so offensichtlich, dass sich die Aufmerksamkeit des Betrachters wegbewegt von den Fragen der Komposition und der internen Relationen zu einer direkteren Erfahrung der Skulptur im Raum als Funktion von Raum, Licht und dem Sehfeld des Betrachters.

„Die Skulpturen werden, nachdem sie keine in sich geschlossenen Formen sind, oft auch spezifisch auf die verschiedenen Räume abstimmt, damit sie mit der sie umgebenden Architektur im Einklang stehen, manchmal soweit, dass die Maße von den Proportionen der Räume abgeleitet werden." [13]



Ohne Titel, Donald Judd (1992), Edelstahl Plexiglas, 10teilg, je 15,2 x 68,5 x 61 cm

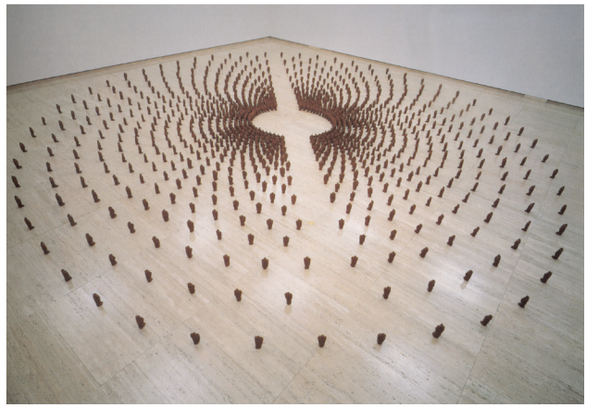
13 Rider, Alistair. (2011) Carl Andre. Things In Their Elements. London: PHAIDON PRESS

Uncarved Blocks. Carl Andre (2005)

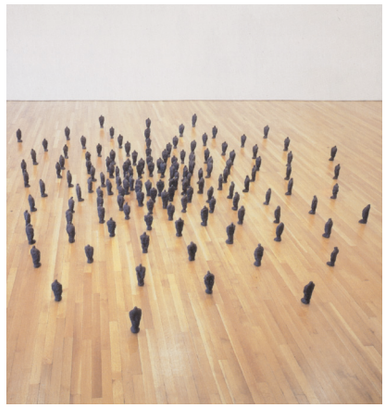
4. 7 Visible Entropy

Antony Gormley

„How can you make the spaces that people displace into a collective energy field - in other words take the idea of spatial extension from the idea of a singularity, producing an expanded field to an immersive field of individual packets of energy?“[14]



Field II (1989). Antony Gormley



Field I (1989). Antony Gormley
Montreal Museum of Fine Arts. (1993) Buch der Ausstellung FIELD von Antony Gormley. München/Stuttgart: Oktagon Verlag S.83

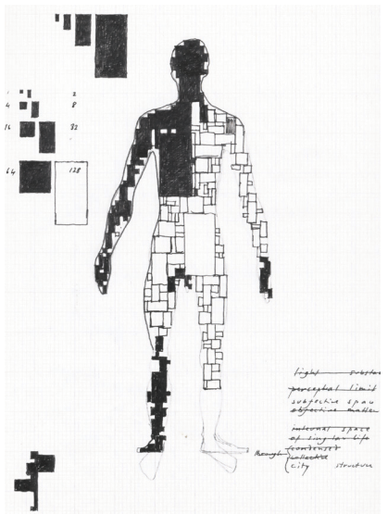
„Made of cast iron from variable blocks, which recall architectural elements each work shows a different body position perhaps caused by a moment of spasm, whereby the human figure has lost its centre of gravity. They appear animated by an invisible impulse coming from somewhere deep inside their core.“[15]

14 Antony Gormley über „Domain Fields“ (2003)

15 Antony Gormley über „Making Space, Taking Place“ (2006)
http://www.antonygormley.com/

Domain Field. Antony Gormley.
BALTIC, The Centre for Contemporary Art, 2003
Katalog der Ausstellung von Antony Gormley im BALTIC, The Centre for Contemporary Art, Gateshead. (2003) DOMAIN FIELD ANTONY GORMLEY. Gateshead.

Making Space, Taking Place. Antony Gormley.
Arte all'Arte 9 - la forma delle nuvole. (2006) FAI SPAZIO PRENDI POSTO ANTONY GORMLEY MAKING SPACE TAKING PLACE. Associazione Arte Continua



Study for Making Space, Taking Place. Antony Gormley

5 Umsetzung

5 Umsetzung

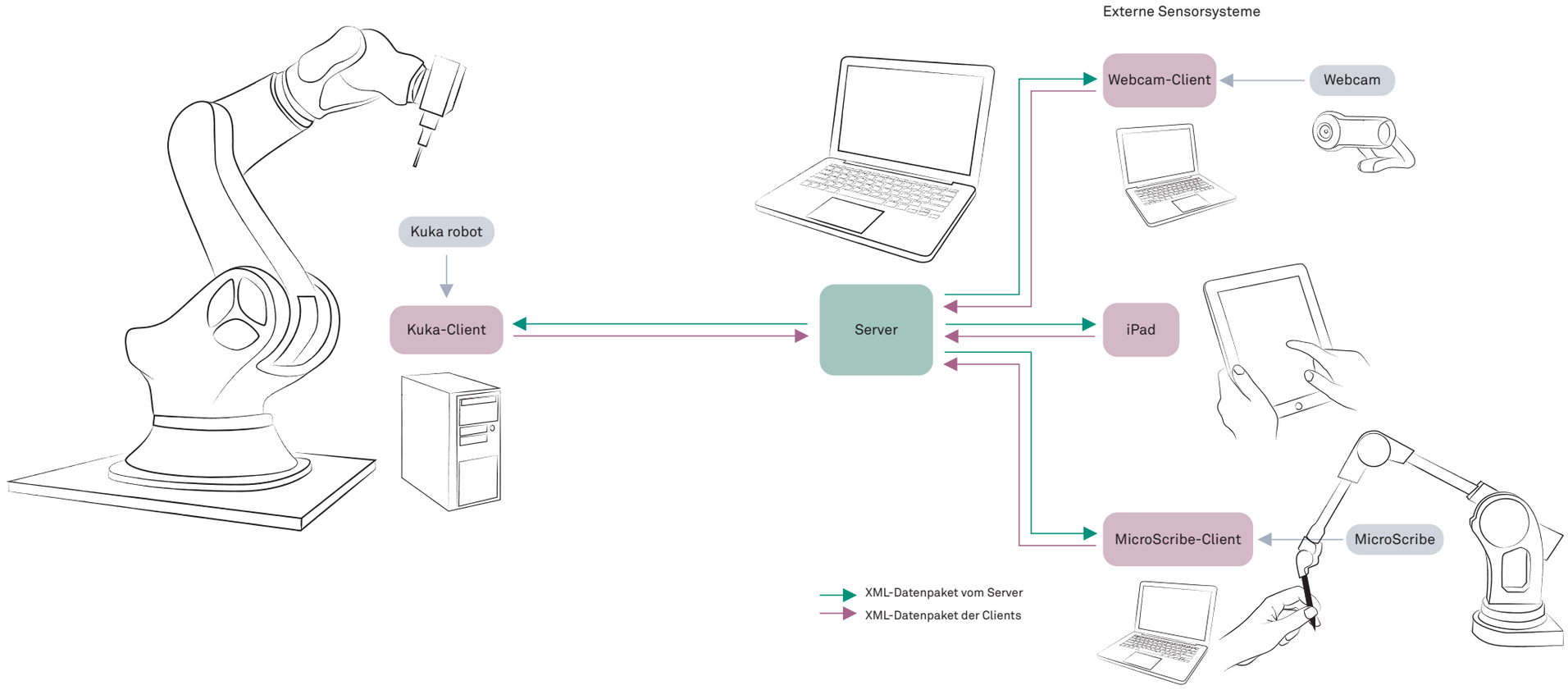
5. 1 Umsetzung

Die folgenden zwei Kapitel „Umsetzung“ und „Fallstudien“ behandeln die Exerimente, mit denen Methoden der interaktiven Bedienung des 6-achsigen Industrieroboters in Gestaltungsprozessen untersucht werden. Die Umsetzung behandelt den technischen Hintergrund für die in den Fallstudien näher erläuterten ausgeführten Experimente, und sie sind inhaltlich teilweise ineinander verschränkt.

Ein Kuka Industrieroboter wird um offene Schnittstellen erweitert, die die online Steuerung des Roboters durch ein externes System ermöglicht. Dieses kann seinerseits mit verschiedensten Ein- und Ausgabegeräte sowie mit Sensoren verbunden werden. So wird das proprietäre Robotersystem für plattformunabhängige Erweiterungen geöffnet und die Grundlage geschaffen für interaktive Gestaltungsprozesse, die die Maschine nicht nur als Fertigungsautomat sondern als Gestaltungswerkzeug begreifen.

In den herkömmlichen Methoden der CNC-gestützten Umsetzung von virtuellen Modellen in physische müssen alle gestalterischen Entscheidungen bereits getroffen sein, die Herstellungsprozesse sind linear und unflexibel, sie nützen das Potenzial der Maschinen für den/die Gestalter/in noch nicht aus. Im Rahmen dieser Arbeit werden Systeme untersucht und geeignete Randbedingungen festgelegt, die menschliche Eingriffe intuitiver Art in den Fertigungsprozess ermöglichen, um Schnittstellen zwischen virtueller und physischer Wirklichkeit zu etablieren. In Hinblick darauf sind Vermittler zwischen beiden Welten vonnöten, also der Roboter als Aktuator vom virtuellen ins physische, und Sensoren vom physischen ins virtuelle.

Sind physische und virtuelle Realitäten ineinander verschränkt, erlaubt das eine Erweiterung der digitalen Entwurfsmethoden in den physischen Raum, wo die Materialien der physischen Welt mit Information, die virtuell vorhanden sind, überlagert werden können, und Daten, die über Sensoren aus der physischen Welt extrahiert werden, als Informationsquelle für das virtuelle Modell dienen. Werden in diesem verschränkten System Regeln und Prozesse definiert und nicht Formen festgeschrieben, so kann durch Iteration Interaktion und Feedback ermöglicht werden.



5. 2 Aufbau und Methode

Beschreibung der interaktiven Steuerung und das ihr zugrunde liegende System

Um eine interaktive Steuerung des Roboters (oder im weiteren auch anderer CNC-Maschinen) zu ermöglichen, wurde eine Kommunikation über die Netzwerk-Schnittstelle gewählt.

Von der KRL-Steuerung des KUKA-Industrieroboters wird das Ethernet KRL-XML-Modul verwendet, das bereits bestimmte Methoden für die Netzwerkkommunikation innerhalb eines laufenden KRL-Programmes bereitstellt.

Es gäbe auch die Möglichkeit einer Verbindung über die serielle Schnittstelle zum Roboter, aber in Hinblick auf eine zukünftige Erweiterung des Systems wurde die Netzwerkschnittstelle als die sinnvollere befunden, da es dafür von KUKA steuerungsabhängig Erweiterungsmodule gibt, wie z.B das RSI-XML (Robot Sensor Interface) Modul oder das FRI (Fast Research Interface) für den KUKA DLR Lightweight Robot.

Der Aufbau

Das Setup besteht aus einem sechsachsigen KUKA-Industrieroboter, einem Computer, und verschiedenen externen Sensorsystemen, die je nach Verfahren hinzugefügt werden können und auswechselbar sind, wie z.B. eine Kamera, ein iPad oder der fünfachsige Microscribe Koordinatenmessarm.

Die Applikation am Computer beinhaltet einen Server, der Roboter und die anderen externen Sensorsysteme werden als Clients dieses Servers behandelt (Diese Struktur wurde anhand der Funktionaliät des KRLXML Moduls entwickelt). Der Server regelt die Kommunikation zwischen den Clients bzw. den Datenaustausch untereinander.

Die Daten werden in Form von XML-Paketen von den Clients gesendet und empfangen. Der Roboter kann beispielsweise XML[1]-Pakete empfangen, die die für ihn anzufahrenden Positionen beinhalten[2] (sowie weitere für das Programm spezielle Variablen), und XML-Pakete an den Server zurück senden.

Die Applikation übernimmt weiters die Auswertung der Daten der Clients, die Berechnungen innerhalb eines gewählten Algorithmus (meistens abhängig von den Eingangsdaten der Clients) und die zwei- oder dreidimensionale Darstellung der jeweiligen Verfahren.

5. 3 Die Applikation

Die Applikation, regelt, wie schon erwähnt, die Kommunikation zwischen den Clients bzw. den Datenaustausch untereinander

1 Extensible Markup Language, kurz XML, ist eine Auszeichnungssprache zur Darstellung hierarchisch strukturierter Daten in Form von Textdaten. XML wird u. a. für den Plattform- und implementationsunabhängigen Austausch von Daten zwischen Computersystemen eingesetzt, insbesondere über das Internet. (http://de.wikipedia.org)

2 Eine exemplarische XML-Nachricht für den Roboter-Client:

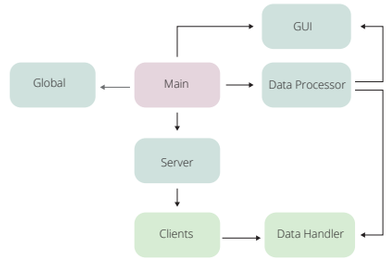
```
<ExternalData>
    <Position>
        <XPos>34.0</XPos>
        <YPos>33.5</YPos>
        <ZPos>2.7</ZPos>
        <APos>0.0</APos>
        <BPos>90.0</BPos>
        <CPos>0.0</CPos>
        <E1Pos>0.0</E1Pos>
    </Position>
  <Boolean>
    <VacuumState>0</VacuumState>
    <SafeLevelBefore>0</SafeLevelBefore>
    <SafeLevelAfter>0</SafeLevelAfter>
    <Motor>1</Motor>
    <CState>0</CState>
  </Boolean>
</ExternalData>
```


und die Berechnung des Algorithmus. Sie ist dabei in mehrere Module unterteilt, die wichtigsten sind hier aufgelistet:

Das Main Modul: Hier werden alle Module zusammengeführt und globale Parameter definiert. Von hier aus wird die Applikation gestartet.

Das GUI: (Graphical User Interface): Über das GUI kann man den gewünschten Algorithmus auswählen und diesen starten, pausieren oder beenden. Ausserdem werden hier die empfangenen sowie gesendeten Daten von den Clients und andere, vom Programm definierte Nachrichten ausgegeben. Bei Bedarf werden hier auch die Berechnungen des Algorithmus in grafischer Form dargestellt.

Das Server-Client Modul: Die Zuordnung der einzelnen Clients erfolgt über definierte IP-Adressen, damit der Server weiß, ob er es z.B. mit einem iPad oder einer Kamera zu tun hat. Dann wird an einem bestimmten Port auf Verbindungen der Clients gewartet, und sobald eine Verbindung aufgebaut ist, wird in einer Schleife das Empfangen und Senden der Datenpakete für diesen Client behandelt.



Struktur der Applikation

Das Data Handler Modul: In diesem Modul werden die Eingangs- und Ausgangsdaten von und für die verschiedenen Clients encodiert und decodiert. D.h. hier werden Daten in definierter Formatierung in XML-Pakete verpackt oder die Werte aus XML-Paketen ausgelesen und zurückgegeben.

Das Data Processor Modul: In diesem Modul wird die ausgewählte Funktion bzw. der Algorithmus ausgeführt. Die Daten werden hier in Abhängigkeit des Algorithmus und der Inputs berechnet, im Modul Data Handler formatiert, und dann dem Server zum Senden übergeben.

Beschreibung des Programmablaufs:

Beim Start der Applikation wird automatisch der Server gestartet, der an einem bestimmten Port auf die Verbindung der Clients wartet. Über das GUI kann man das gewünschte Verfahren (Algorithmus) auswählen, damit wird auch festgelegt, welche Verbindungen (Clients) vom Server erwartet werden, um den Algorithmus ausführen zu können.

Damit vom KUKA-Roboter zum Server eine Kommunikation hergestellt werden kann, startet man am KUKA-Rechner ein KRL (KUKA Robot Language) – Programm. Dieses Programm erstellt eine Verbindung zwischen dem KUKA-Steuerungssystem und dem Server und wartet dann fortwährend an einem definierten Port auf XML-Nachrichten, deren Struktur beliebig vordefiniert werden kann, z.B. kann die XML-Nachricht die Position (X, Y, Z, A, B, C) oder weitere, für den Algorithmus speziellen Variablen (z.B. Vakuum On/Off) beinhalten.

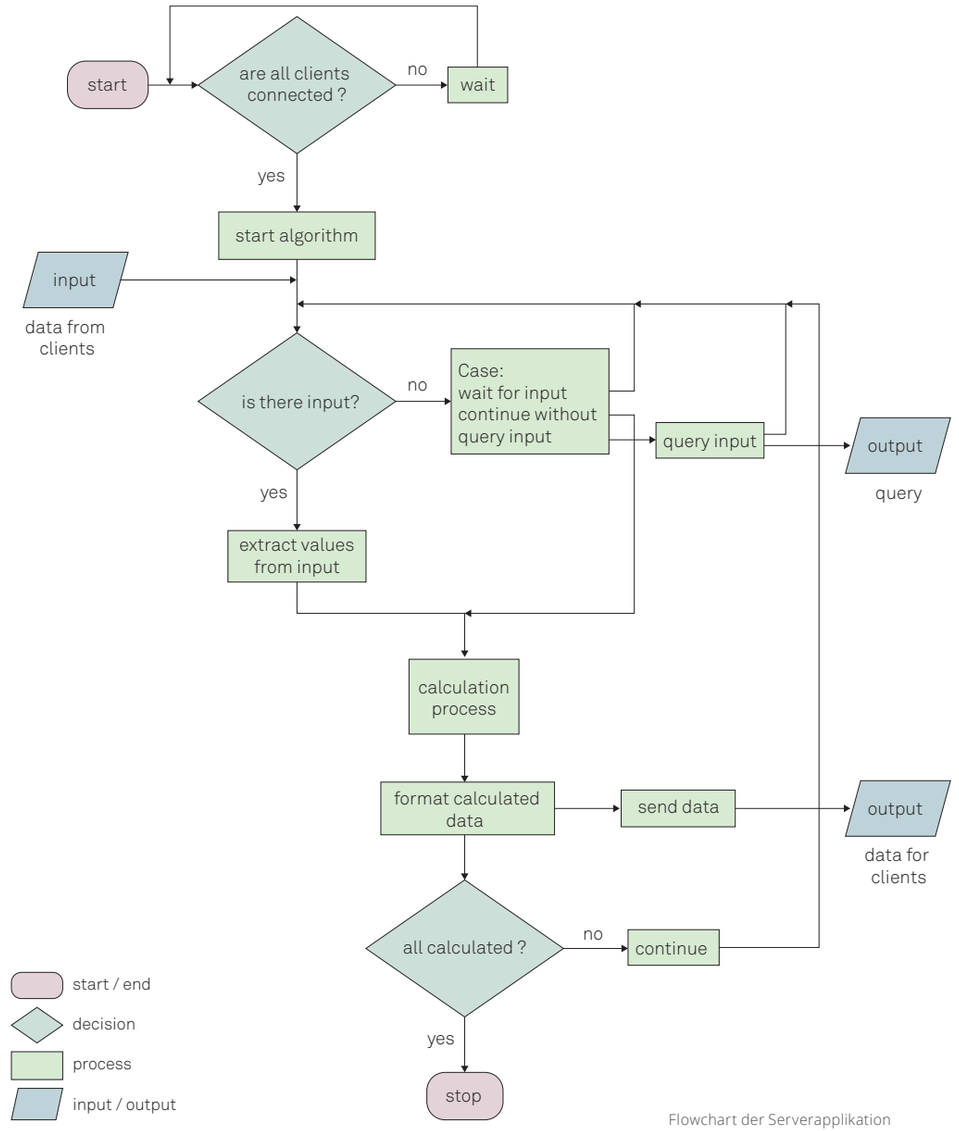
Außer dem iPad, welches sich direkt zum Server verbinden kann (ein Server-Client Modul wurde in den jeweiligen Apps am iPad integriert), werden die jeweiligen Sensorsysteme wie Kamera oder Microscribe an einen externen Rechner angeschlossen, über den anschließend die Verbindung (durch ein Python-Programm) zum Server erstellt wird, um Daten zu senden und zu empfangen.

Sind alle für den Algorithmus notwendigen Clients verbunden, kann das Programm (der Algorithmus) gestartet werden. Wenn das Programm Eingabedaten erfordert, werden diese über den Server von den Clients abgefragt. Die Clients senden ihre Daten verpackt als XML-Pakete an den Server, der diese an den Data-Handler weitergibt, wo sie ausgewertet werden, und so in das laufende Programm einfließen können. Abhängig von Eingabeparametern der Clients werden dann Positionen für den Roboter berechnet, die wiederum als XML-Pakete an den Roboter-Client gesendet werden.

Im laufenden KRL-Programm am Roboter-Client werden die XML-Pakete empfangen, ausgelesen und abhängig von den ausgelesenen Werten verschiedene Roboter-Befehle ausgeführt. Sobald der Roboter eine Anweisung erledigt hat, wird über das KUKA-Steuerungssystem eine XML-Nachricht an den Server mit der exakten Position des Roboters zurückgesendet. Dies ermöglicht eine Kontrolle der „Ist-Situation", bzw. ob und wie viele Befehle ausgeführt worden sind.

Das Programm läuft nun so lange, bis der Algorithmus zu einem Ende kommt oder man den Prozess unterbricht. Innerhalb dieses Zeitraums besteht nun die Möglichkeit, zu interagieren, d.h. die vorher variabel definierten Parameter zu verändern, und somit den Prozess mit neuen Eingaben zu verändern.

Was in dem gewählten Aufbau oder System steckt und welche Möglichkeiten es schafft, diesen Fragen haben wir uns in verschiedenen Experimenten und Verfahren anzunähern versucht (siehe Kapitel Fallstudien). Dabei sind wir nicht nur auf die speziellen Eigenschaften des Systems gestoßen, sondern haben die Applikation ständig erweitert und angepasst.



Flowchart der Serverapplikation

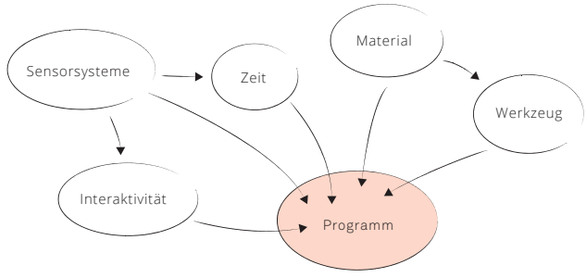
5. 4 Das Entwerfen der Algorithmen

Einfließende Parameter

Beim Entwerfen der Algorithmen zum Zeichnen auf einem Blatt Papier oder zum Stapeln von Bauklötzen oder zum Abwickeln eines Fadens gibt es viele Parameter zu berücksichtigen.

Die Algorithmen müssen auf einem Modell der Wirklichkeit beruhen, d.h vom Material mitgebrachte Randbedingungen (Materialeigenschaften wie Gewicht, Größe, Form, Stabilität) müssen im Algorithmus integriert sein. Auch die Wahl der Werkzeuge und/oder der verschiedenen Devices oder Sensorsysteme und der ihnen zugrunde liegenden Eigenschaften sind Teil des Programms (z.B. dass ein Pinsel in Farbe getaucht werden muss, anders als beim Filzstift).

Zum Erstellen von Formen und Strukturen im virtuellen Raum gibt es einen wichtigen Unterschied, nämlich dass etwas „aufgebaut“ (evtl. auch abgebaut) wird. Dass Vorgänge hintereinander passieren müssen, einer bestimmten Reihenfolge entsprechen müssen, und nicht anders.



Einfließende Parameter in den Algorithmus

Damit Interaktion stattfinden kann, ist es wichtig, dass im Algorithmus sinnvolle Parameter definiert werden, die variabel sind, und während der Laufzeit des Programms verändert werden können, oder das Programm überhaupt von Eingaben von Seiten der Benutzer abhängt, und somit für das Fortsetzen des Programmes eine Interaktion erforderlich ist.

Somit entsteht ein halboffenes System, das klaren Designentscheidungen unterliegt, also aus definierten Randbedingungen besteht, aber in dem Bereiche noch offen sind, die erst im Bauen oder im Herstellen zu Entscheidungen führen.

Wie bereits im Kapitel Einführung erwähnt, gibt es Bezüge zu Christopher Alexanders Theorie der Morphogenese[3], die er in „Notes on the Synthesis of Form“[4] beschreibt, dass ein Programm als Genotyp verstanden werden kann, und die einzelnen realisierten Lösungen davon als Phänotypen. Bzw. die DNA bestimmt zwar den Entstehungsprozess und ist die Grundlage jeder Entfaltung, aber die tatsächliche Form hängt von den jeweiligen Faktoren aus der Umwelt ab (z.B Licht, Nahrung usw.)

3 vgl. Herresthal, Kristina: „Von fließender Systematik und generativen Prozessen - Christopher Alexander im Gespräch mit Rem Koolhaas und Hans Ulrich Obrist“, in: ARCH+ Ausgabe Nr. 189, Zeitschrift für Architektur und Städtebau, Oktober 2008. S.23, S.25.

4 Alexander, Christopher. (1964) Notes on the Synthesis of Form. Cambridge, Massachusetts, London: Harvard University Press

5. 5 Wahl der Programmiersprache

Very pythonic!

Als Programmiersprache wählten wir die zu den höheren Programmiersprachen gehörende, und für alle gängigen Betriebssysteme (Linux/Unix, Mac, Windows, u.a.) frei erhältliche Programmiersprache Python. Die Python Implementierung besitzt eine Open Source Lizenz, die sie frei verwend- und verteilbar macht (auch für den kommerziellen Gebrauch).

Sie zeichnet sich neben ihrer Offenheit und ihrer leichten Lesbarkeit (das ohnehin übliche Einrücken definiert gleichzeitig auch die Blockstruktur) auch dadurch aus, dass man Programme anderer Sprachen (wie z.B. die maschinennähere Sprache C) als Modul einbetten kann, um beispielsweise zeitkritische Teile im Programm mit der Programmiersprache C zu ersetzen.

Aufgrund der Tatsache, dass Python ständig weiterentwickelt wird und als moderne Sprache gilt, standen viele notwendige Bibliotheken/Libraries für unser Programm in Python zur Verfügung.[5]

Zusätzlich wird Python auch oft als Scriptsprache in 3D-Modeling Programmen (z.B. Rhino, Blender, Cinema 4D, Maya, City Engine) zur Verfügung gestellt.

Python Logo

5 http://www.python.org

Libraries:

Element Tree XML API (http://docs.python.org/library/xml.etree.elementtree.html)

Numpy (http://numpy.scipy.org/)

OpenCV (http://opencv.willowgarage.com/wiki/PythonInterface)

VTK (http://www.vtk.org/)

QT (http://www.riverbankcomputing.co.uk/software/pyqt/intro)

5. 6 Roboter

Programmiersystem

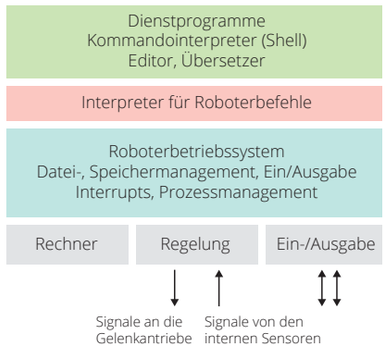
Die Programmierumgebung des Roboters umfasst die gesamte Hard- und Software zur Programmerstellung und -ausführung. Die Robotersteuerung besteht aus Hard- und Software, die die Entgegennahme von Roboterbefehlen und -programmen, Abarbeitung von Bewegungskommandos und Weiterleitung der Kommandos an Gelenkregler übernehmen. Die Software im Robotersteuerrechner besteht aus dem Roboterbetriebssystem, dem Interpreter für Roboterbefehle und verschiedener Dienstprogramme.

Das Roboterbetriebssystem organisiert den Ablauf der Programme, den Zugriff auf Speicher und Dateien, die Prozesskoordination, über die Ein-/Ausgabeschnittstelle den Anschluss von Peripheriegeräten und Sensoren und den Anschluss von Leitrechnern.

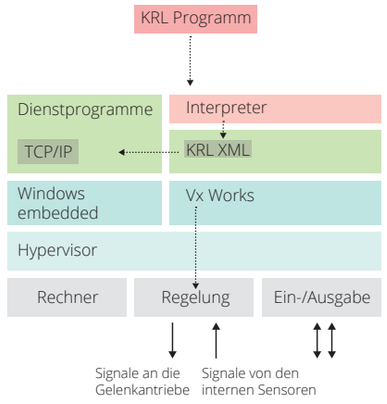
Der Interpreter für die Roboterbefehle ist verantwortlich für die Ausführung einzelner roboterorientierter Anweisungen. Er zerlegt Bewegungen in eine Folge von Zwischenpositionen (Bahninterpolation[6]) und leitet Bewegungsinkremente an die Regelung weiter.

Für unsere Anwendung haben wir ein KRL-Programm (KUKA Robot Language) erstellt, welches vom Roboter-Interpreter ausgewertet wird. Der Interpreter greift für die Methoden vom Modul KRL XML auf den Treiber vom KRL XML Modul (befindet sich in VxWorks) zu, der über das TCP/IP Protokoll (befindet sich Windows embedded) die XML Pakete empfängt und sendet.

Das heißt, die Bahninterpolation wird vom KUKA Steuerungssystem übernommen, was über die Netzwerkschnittstelle empfangen wird, sind die einzelnen Roboterpositionen und für das Programm spezifische Variablen.



Programmiersystem
vgl. Prof. Dr. Thomas Ihme, Lehrveranstaltung Autonome Mobile Roboter (AMR), WS 2010, Fakultät für Informatik, Hochschule Mannheim



KUKA Programmiersystem mit Ethernet.KRL XML

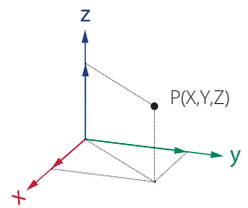
6 Die Bahninterpolation generiert das Verfahrprofil für die Bahn, berechnet im IPO- Takt die Bahnstützpunkte und leitet daraus über die Kinematiktransformation die Achssollwerte zu den IPO-Taktzeitpunkten ab.

Koordinatensysteme

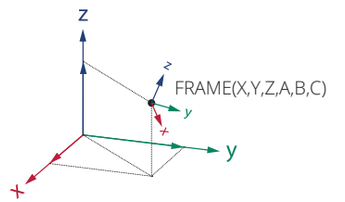
Um einen Punkt im Raum eindeutig zu identifizieren, können verschiedene Koordinatensysteme (z.B. Zylinderkoordinatensystem, Polarkoordinatensystem, kartesisches Koordinatensystem) verwendet werden.

Um für Roboter die Position des Endeffektors (Roboter-Werkzeugspitze) im Raum eindeutig zu bestimmen, braucht es zusätzlich zur Angabe des Punktes auch noch Angaben zur Orientierung.

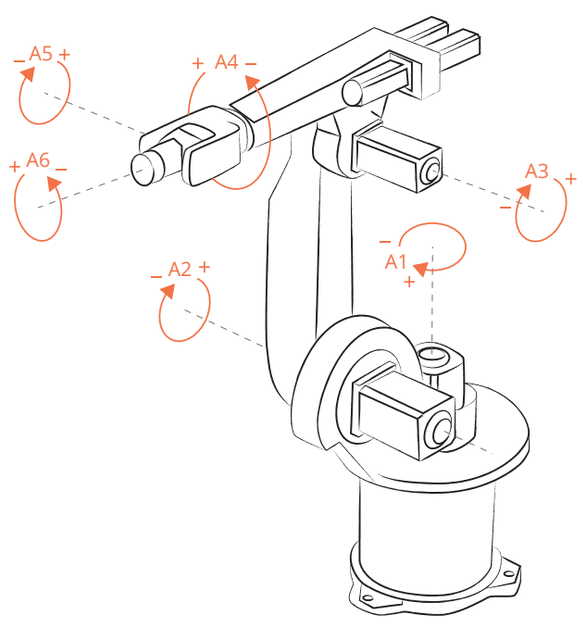
Dazu hat man die Möglichkeit, diese Position eindeutig über das achs-spezifische Koordinatensystem anzugeben (Winkelangaben für alle Achsen des Roboters), oder über Frames, die ein Koordinatensystem in Bezug auf ein anderes vollständig beschreiben, da sie zusätzlich zu den Positionen (X,Y,Z) noch die Eulerwinkel (A,B,C) als Information zur Orientierung des Endeffektors beinhalten.



Kartesisches Koordinatensystem



FRAME im kartesischen Koordinatensystem



6 Achsen - achs-spezifisches Koordinatensystem

Geometrische Datentypen in KRL

Von KUKA sind folgende geometrische Datentypen definiert:

AXIS	REAL	A1,A2,A3,A4,A5,A6
E6AXIS	REAL	A1,A2,A3,A4,A5,A6,E1,E2,E3,E4,E5,E6

Achs-spezifisches Koordinatensystem

POS	REAL	X,Y,Z,A,B,C, INT S,T
E6POS	REAL	X,Y,Z,A,B,C,E1,E2,E3,E4,E5,E6, INT S,T
FRAME	REAL	X,Y,Z,A,B,C

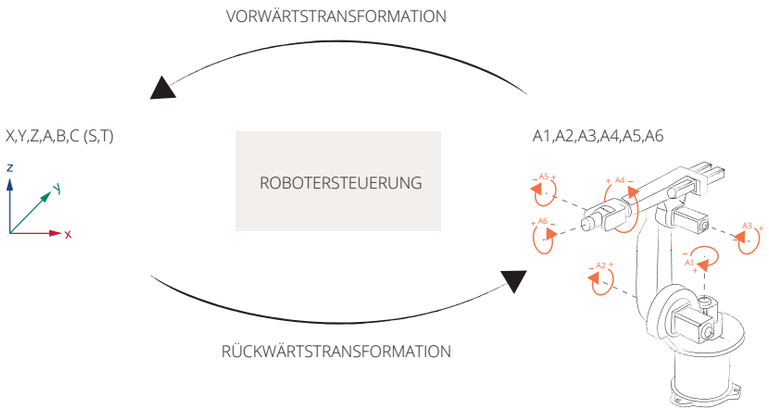
Kartesisches Koordinatensystem

Die Komponenten des Datentyps AXIS sind Winkelangaben für die einzelnen Achsen, die des Datentyps FRAME sind 3 Werte für die Position (X,Y,Z) und 3 Werte für die Orientierung (A,B,C) Eulerwinkel im kartesischen Koordinatensystem.

Kinematik

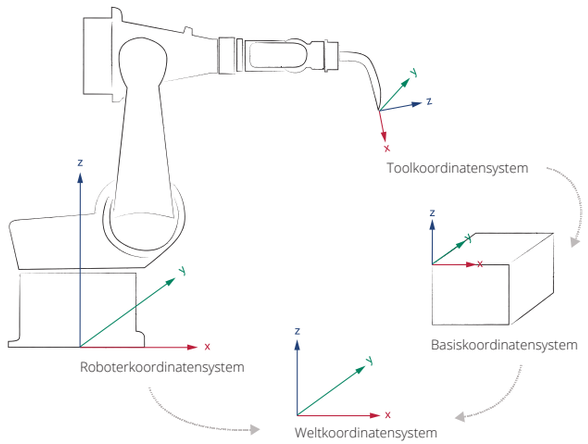
Die Angaben für die Endeffektorstellung des Roboters im kartesischen Koordinatensystem vom Datentyp FRAME werden von der Robotersteuerung automatisch in achs-spezifische Werte transformiert, bevor eine Bewegung ausgeführt wird (Rückwärtskinematik).

Die Transformation muss in Echtzeit durchgeführt werden, die Ergebnisse sind in der Regel nicht eindeutig. Um die Eindeutigkeit sicherzustellen, muss der FRAME um Angaben zur Drehrichtung der Achsen ergänzt werden, dazu verwendet man den Datentyp POS, der um die zwei Komponenten S und T erweitert ist.



Vorwärts- und Rückwärtskinematik

Von KUKA vordefinierte FRAMES:



Kartesische Koordinatensysteme (FRAMES) für Roboter

Weltkoordinatensystem: Befindet sich außerhalb des Roboters und ist die Referenz für das gesamte Robotersystem.

Roboterkoordinatensystem: Befindet sich am Fuß des Roboters und bildet das Referenzsystem für den mechanischen Aufbau des Roboters. Es ist abhängig vom Weltkoordinatensystem.

Toolkoordinatensystem: Hat seinen Ursprung an der Spitze des Werzeuges. (Tool Center Point: TCP). Die Orientierung kann so gewählt werden, dass die X-Achse identisch mit der Richtung des Werkzeuges ist und nach unten zeigt. Wenn das Werzeug bewegt wird, bewegt sich das Toolkoordinatensystem mit. Es ist abhängig vom Basiskoordinatensystem.

Basiskoordinatensystem: Das Basiskoordinatensystem beschreibt die Position des Werkstücks in Relation zum Weltkoordinatensystem. Der Roboter wird immer im Basiskoordinatensystem programmiert, damit ist es möglich, dasselbe Programm in verschieden eingemessenen Basen zu verwenden. Bei der Bahninterpolation werden die Positionen für das Toolkoordinatensystem in Bezug zum Basiskoordinatensystem berechnet.

FRAME Linkage

In KRL gibt es unter anderem den geometrischen Operator „:“, mit dem man die Datentypen FRAMES oder POS miteinander verketten kann, die Verkettung von zwei FRAMES ist eine Transformation der Koordinatensysteme (siehe Koordinatentransformation S. 59).

Arbeitsraum:

Der Arbeitsraum besteht aus denjenigen Punkten im 3-D Raum, die von der Roboterhand angefahren werden können.

Die Grundform des Arbeitsraums ist der Raum, der sich ergeben würde, wenn man die gegenseitige Behinderung der Armsegmente des Roboters und die Begrenzung der Gelenkwinkel nicht berücksichtigt.

5. 7 Koordinatentransformation

Für die Abbildung von Punkten in den verschiedenen kartesischen Koordinatensystemen (Roboterbasis, Roboterwerkzeugspitze, iPad, Kamera, Microscribe) ist es notwendig, eine Koordinatentransformation durchzuführen.

Die Koordinatentransformation bezeichnet die Übertragung von den ursprünglichen Koordinaten $(x_1, ..., x_n)$ zu den neuen Koordinaten $(x'_1, ..., x'_n)$.

Im Rahmen dieses Projektes werden affine Transformationen benötigt. [7]

Bei der affinen Transformation errechnet sich der neue Koordinatenvektor $\vec{x}' = (x'_1, ..., x'_n)$ durch die Multiplikation des ursprünglichen Koordinatenvektors $\vec{x} = (x_1, ..., x_n)$ mit der Matrix A (der Abbildungsmatrix) und der Addition mit dem Verschiebungsvektor $\vec{b}$:

$$\vec{x}' = A \cdot \vec{x} + \vec{b}$$

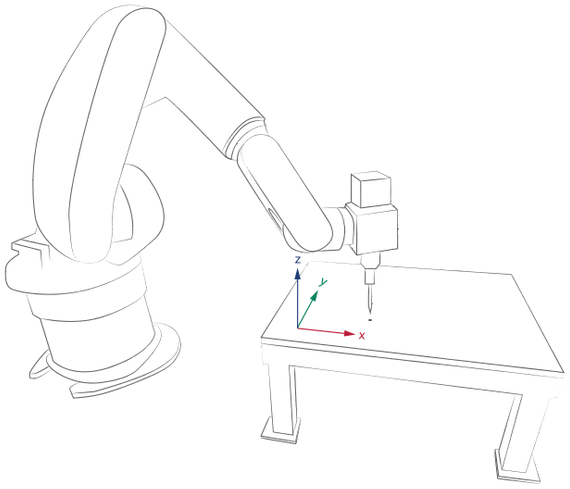
Für den dreidimentionalen Raum ist A eine 3×3 Matrix, die eine Kombination aus Rotation und Skalierung beschreiben kann und der 3×1 Vektor $\vec{b}$ stellt eine Translation (Verschiebung) dar.

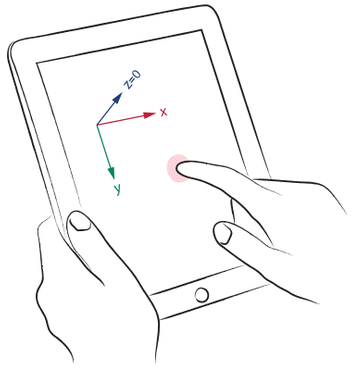
Seien nun die Koeffizienten a_{ij} von der Matrix A gegeben und die b_i die Koeffizienten vom Vektor $\vec{b}$, dann lässt sich die Transformation darstellen mit:

$$\begin{pmatrix} x' \\ y' \\ z' \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix} \times \begin{pmatrix} x \\ y \\ z \end{pmatrix} + \begin{pmatrix} b_1 \\ b_2 \\ b_3 \end{pmatrix}$$

Da die Aufteilung in Multiplikation und Addition bei der Berechnung oft hinderlich ist, erweitert man die 3×3 Matrix um eine Spalte und eine Reihe, um eine 4×4 Matrix zu erhalten, und dadurch die Verschiebung auch in der Matrix unterzubringen. (Homogene Transformation) Dazu muss man die kartesischen Koordinaten zu homogenen Koordinaten transformieren, d.h. man erweitert sie um eine Dimension.

7 Eine affine Transformation bezeichnet im Prinzip die Abbildung zwischen zwei linearen Koordinatensystemen (durch einen Koordinatenursprung und gleichmäßig unterteilte Koordinatenachsen gegeben) bei der Kollinearität, Parallelität und Teilverhältnisse bewahrt bleiben.





Koordinatensystem Ipad-Roboter

$$(x, y, z)^T \rightarrow (x, y, z, 1)^T$$

Somit erhält man die Transformationsmatrix mit:

$$D = \begin{pmatrix} R & T \\ P & S \end{pmatrix} \qquad \begin{array}{lll} R & - & 3\times3 \text{ Rotation} \\ T & - & 3\times1 \text{ Translation} \\ P & - & 1\times3 \text{ Perspektivtransf. (keine)} \\ S & - & 1\times1 \text{ Skalierungsfaktor (=1)} \end{array}$$

Um nun einen Punkt (x, y, z) vom ursprünglichen Koordinatensystem im neuen Koordinatensystem abzubilden, muss man ihn nach der Dimensionserweiterung nur mehr mit der Matrix multiplizieren.

$$\begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ 0 & 0 & 0 & 1 \end{pmatrix} \times \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

R T

P S

5. 8 Eulerwinkel und Yaw-Pitch-Roll

Zur Festlegung der Roboterachsen benötigt man die Drehwinkel (Eulerwinkel) α , β , γ . Die Winkel stellen jeweils eine

Rotation um bestimmte Achsen (in unserem Fall: $z - y' - x''$) dar und definieren so die Orientierung des Roboterwerkzeug-Koordinatensystemin Bezug zum Roboterbasis-Koordinatensystem.

$R_z(\alpha)$: Drehung α um die $z\text{-}Achse$ (yaw)

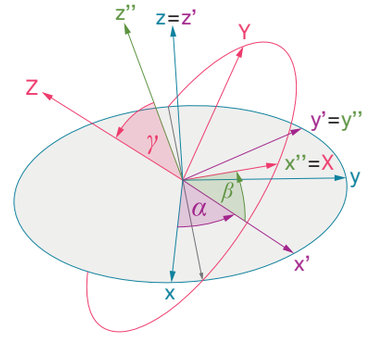
$R_{y'}(\beta)$: Drehung β um die verdrehte $y\text{-}Achse$ y' (pitch)

$R_{x''}(\gamma)$: Drehung γ um die verdrehte $x\text{-}Achse$ x'' (roll)

Da sich die Rotationsmatrizen verketten lassen, kann man sie miteinander multiplizieren: und zwar, wenn man immer um die gedrehte Achse rotiert: von links nach rechts und wenn man um feste Achsen rotiert: von rechts nach links, in der Reihenfolge der Ausführung der Rotationen.

$$R_s(\alpha, \beta, \gamma) = R_z(\alpha) \cdot R_{y'}(\beta) \cdot R_{x''}(\gamma)$$

$$= \begin{pmatrix} \cos\alpha\cos\beta & \cos\alpha\sin\beta\sin\gamma - \sin\alpha\cos\gamma & \cos\alpha\sin\beta\cos\gamma + \sin\alpha\sin\gamma \\ \sin\alpha\cos\beta & \sin\alpha\sin\beta\sin\gamma + \cos\alpha\cos\gamma & \sin\alpha\sin\beta\cos\gamma - \cos\alpha\sin\gamma \\ -\sin\beta & \cos\beta\sin\gamma & \cos\beta\cos\gamma \end{pmatrix}$$



Eulerwinkel α, β, γ

Wenn man die Rotation des einen Koordinatensystems zum anderen kennt, dann ist es möglich, die Rotationsmatrix R zu berechnen durch die Aneinanderreihung beliebiger Rotationen. Von dieser Matrix R ausgehend, kann man die Eulerwinkel α , β , γ bestimmen, indem man die Koeffizienten von R zum Matrixprodukt $R_s(\alpha, \beta, \gamma)$ vergleicht.

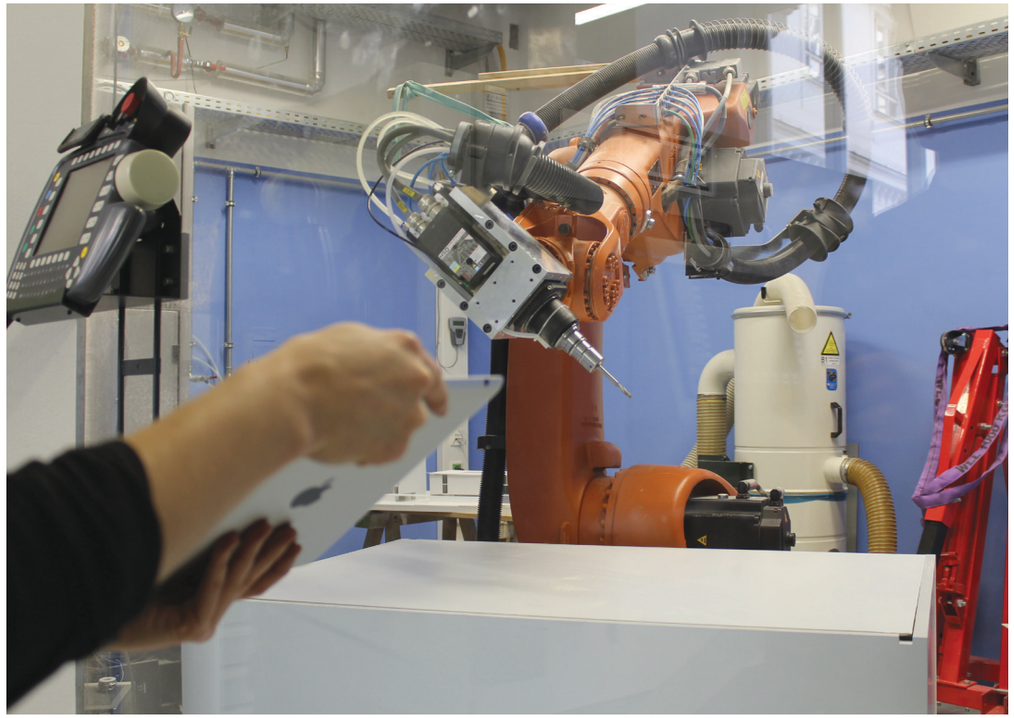
$$R = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix} \qquad \begin{array}{l} a_{31} = -\sin(\beta) \\ \Rightarrow \beta = -\sin^{-1}(a_{31}) \end{array}$$

Allerdings muss man mit der Interpretation dieser Gleichung vorsichtig sein, da $\sin(\beta) = \sin(\pi - \beta)$. (Der Einfachheit wegen wird der zweite Fall $\sin(\pi - \beta)$ hier nicht behandelt.)

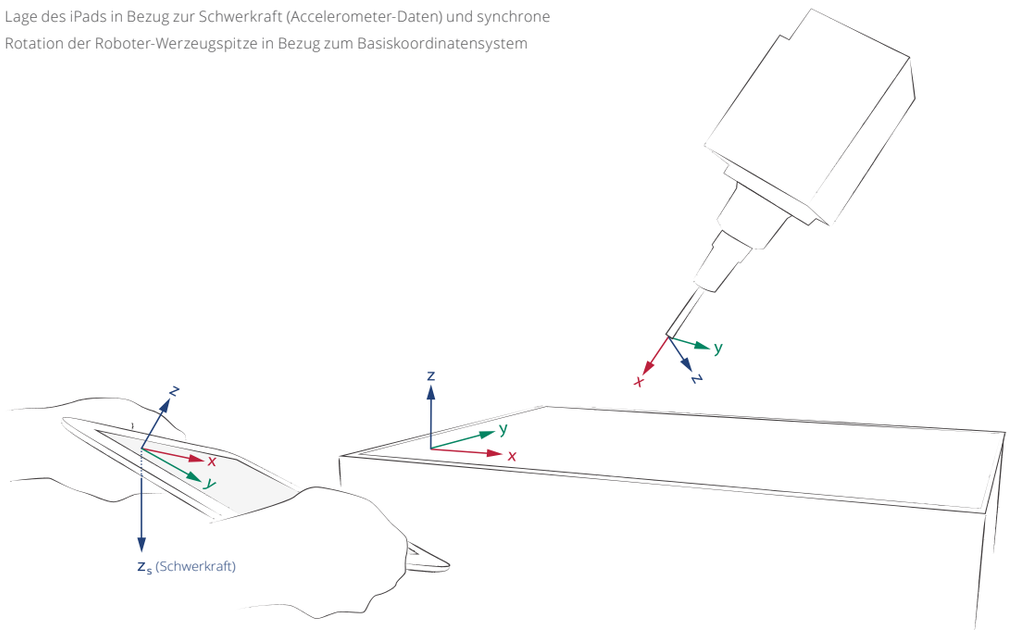
Weiters kann man nun durch Umformung α und γ bestimmen:

$$\frac{a_{32}}{a_{33}} = \tan(\gamma) \qquad \Rightarrow \gamma = \text{atan2}(a_{32}, a_{33})$$

$$\frac{a_{21}}{a_{11}} = \tan(\alpha) \qquad \Rightarrow \alpha = \text{atan2}(a_{21}, a_{11})$$

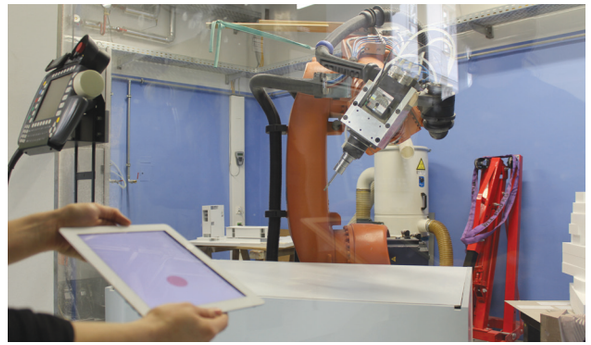


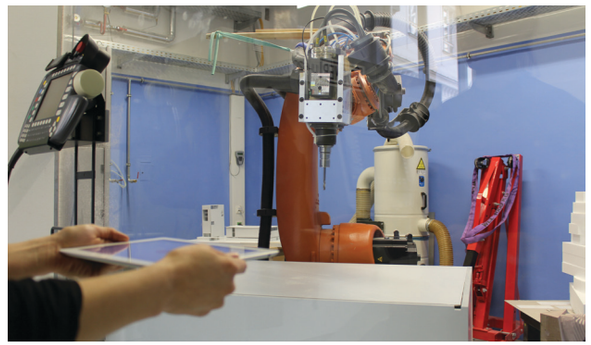
Lage des iPads in Bezug zur Schwerkraft (Accelerometer-Daten) und synchrone Rotation der Roboter-Werzeugspitze in Bezug zum Basiskoordinatensystem

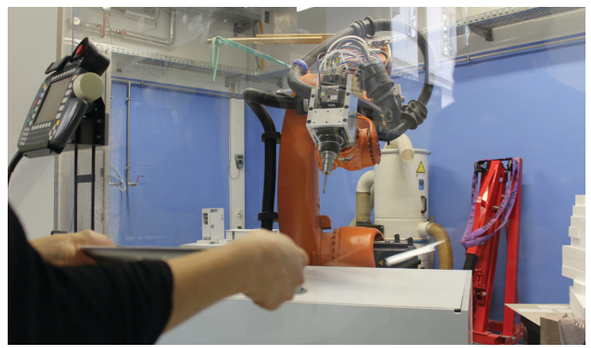


5. 9 Synchronbewegung iPad und Roboter - Werkzeugspitze

Umrechnung der Accelerometer-Daten des iPads in ABC-Eulerwinkel für den Roboter: Das Rotieren des iPads veranlasst die Rotation des Koordinatensystems der Roboter-Toolspitze dahingehend, dass die Toolspitze immer senkrecht zum iPad-Display steht.







http://vimeo.com/37554666

5. 10 Zeit und Interaktivität

In welchem Zeitrahmen ist Interaktion für den User erfahrbar?

Der Fertigungsprozess vom Roboter braucht eine gewisse Zeit. Betrachtet man beispielsweise als Aufgabe das Setzen eines Bausteines an eine bestimmte Position, dann sind das für den Roboter vier Vorgänge:

1. Fahre an die Position an der der Baustein liegt
2. Hebe den Baustein auf
3. Fahre zur Stelle an die der Baustein platziert werden soll
4. Setze den Baustein ab

Im virtuellen Raum ist man sich der Zeitlichkeit dieser Vorgänge nicht bewusst: Ein Würfel wird an einer bestimmten Position erzeugt und das Verschieben desselben erfolgt in Millisekunden. Bei parametrisch konstruierten Modellen bewirkt die Veränderung von Parametern die sofortige Veränderung des ganzen Modells.

Will man nun in den Herstellungsprozess interaktiv eingreifen, kommt man nicht umher, den Faktor Zeit als wesentliches Element zu berücksichtigen. Eine bestimmte Aufgabe hat im Allgemeinen mehrere Vorgänge zur Folge, und es dauert daher länger, bis ein Ergebnis einer bestimmten User-Eingabe erkennbar ist. Dieser zeitliche Abstand, der von einem Input des Benutzers bis zur Ausführung einer Aufgabe durch die Maschine besteht, und dem daraus folgenden Erkennen oder Nicht-Erkennen einer interaktiven Handlung, ist für die Arbeit wesentlich.

5. 11 Übersetzung von Eingabeparametern

Wie User-Eingaben in den Algorithmus einfließen

Nicht nur der zeitliche Abstand von einer Eingabe bis zur Ausführung ist entscheidend, wie eine Interaktion wahrgenommen wird, sondern auch, wie sie umgesetzt wird, ob sie

direkt oder indirekt übertragen wird. Bei einer indirekten Übertragung ist wesentlich, mit welchem Abstraktionsgrad der Eingabeparameter zur Ausgabe geführt wird. Ist der Abstraktionsgrad zu hoch, kann man nicht mehr erkennen, wie die Eingabe überhaupt einen Einfluss auf den Prozess ausübt. Der Abstraktionsgrad muss also so gewählt werden, dass der Benutzer in relativ kurzer Zeit erlernen kann, wie sich seine Eingabe auf das System auswirkt.

Es ist wichtig, die Balance zu finden zwischen dem, was durch fixe Parameter im Algorithmus vorgegeben wird, und dem, was man an Entscheidungsmöglichkeiten für den User bereit stellt. Das Feedback durch den User sollte auch bei komplexen Algorithmen zu möglichst individuellen, voneinander unterscheidbaren Lösungen führen.

5. 12 Feedback in Prozessen

Überprüfung lokaler Gegebenheiten

Ein Feedback in Prozessen ist die erkennbare Rückmeldung eines vollzogenen Ereignisses zurück in das Ursprungssystem, wobei die Eingangsgröße in direkter oder indirekter Form die Ausgangsgröße beeinflusst. Man unterscheidet zwischen positivem und negativem Feedback. Beim positiven Feedback verstärkt die Eingangsgröße den vom System geführten Prozess und beim negativen Feedback kommt es zu dessen Abschwächung bzw. Begrenzung.

Beispiel negatives Feedback: Ein Thermostat vergleicht die Ist-Temperatur eines Thermometers mit der eingestellten Soll-Temperatur. Wenn es einen Unterschied zwischen diesen beiden Werten gibt, reguliert das Thermostat die Heizung fortwährend, bis der Ist-Zustand dem Soll-Zustand entspricht (kybernetische Regelung, negatives Feedback).

Das Feedback kommt in unseren Versuchen z.B. von der Kamera oder vom User, von dem ein Ist-Zustand wahrgenommen bzw. gemessen wird. Fließen die gemessenen und erkannten Werte des Ist-Zustandes in den Algorithmus als Eingangsgrö-

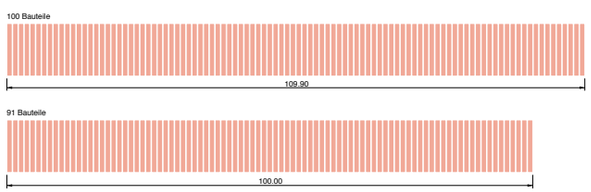
ßen ein, entstehen Feedbackschleifen. Von wem das Feedback gegeben wird, macht für den Prozess keinen Unterschied.

Die Messung des Ist-Zustandes erfolgt immer nach Beenden eines Iterationsschrittes im Algorithmus. Das Verarbeiten von Benutzereingaben oder das Auswerten eines Kamerabildes kann also nur nach der Vollendung des aktuellen Iterationsschrittes erfolgen, d.h wenn ein Iterationsschritt noch nicht beendet ist, (z.B der Roboter noch in Bewegung ist) wird die Verarbeitung der Eingabe oder die Auswertung des Ist-Zustandes verzögert.

Der Vorteil, dass man den aktuellen Zustand als Eingangsgröße hernimmt und nicht eine vom Programm kalkulierte Ausgangsgröße, räumt dem System die Eigenschaft ein, Fehler zu erlauben und ermöglicht das Verwenden von unregelmäßigen Bauteilen oder speziellen Materialien, deren Verhalten man nicht oder nur sehr aufwendig vorherbestimmen kann.

Bei den Faden-Experimenten (siehe Kapitel Fallstudien S. 110) wird ein Faden von einer Spule abgewickelt und fällt von einer gewissen Höhe auf eine Platte. Ausgegangen wird davon, durch unterschiedliche Mengen an fallengelassenem Faden an bestimmten Stellen unterschiedliche Farbintensitäten zu erreichen, aber wo und wie dieser Faden genau auf die Platte fällt, ist nur bedingt vorherbestimmbar. Deshalb ist der Algorithmus so gestaltet, dass man einer bestimmten Farbintensität an einer Stelle nicht vorher eine gewisse Menge an Faden zuweist, sondern an der Stelle solange Faden fallen lässt und die Zustände (Farbintensität) in gewissen Zeitabständen misst, bis der gewünschte Zustand erreicht ist.

Ein anderes Beispiel, bezogen auf die Experimente mit den Holzwürfeln (siehe Kapitel Fallstudien S. 100), wäre, dass man nicht bestimmt, 100 Bausteine aneinander zu reihen, sondern im Algorithmus definiert, so lange Bausteine aneinander zu reihen, bis eine gewisse Marke überschritten wird (lokal vs global):



Das bedeutet, man muss die Breite der Bausteine nicht vorher wissen, und muss nicht im Voraus berechnen, wieviele Bausteine für das Überbrücken einer gewünschten Distanz gebraucht werden, sondern es benötigt nur der wiederholten lokalen Überprüfung des Ist-Zustandes, um diesen mit dem Soll-Wert zu vergleichen, daraus folgend kann auf Zustände „reagiert“ werden.

5. 13 Mathematische Darstellung des Verfahrens

Feedback

Man kann die Experimente abstrakt beschreiben, indem man die unterschiedlichen Systeme im mathematischen Raum abbildet.

Das System besteht aus verschiedenen Zuständen und aus den Übergängen von einem Zustand in den nächsten. Diese Iterationsvorschriften werden in Form von Funktionen dargestellt, die Ergebnisse der einzelnen Funktionen sind die Zustände.

Dazu orientiert man sich am Konzept des Phasenraums aus dem Feld der Theoretischen Informatik:

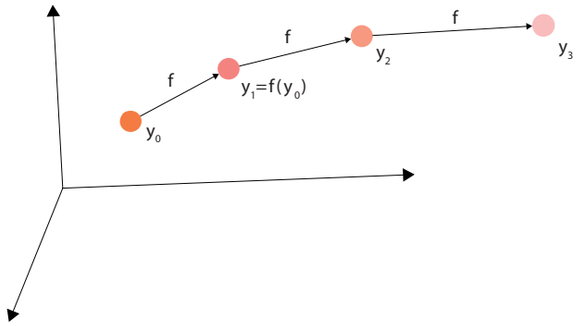
Der Phasenraum (Zustandsraum) beschreibt die Menge aller möglichen Zustände, $M = \{y_0, y_1, ..., y_n\}$, die ein dynamisches System einnehmen kann. Der Phasenraum bildet einen mathematischen Raum, der von sämtlichen veränderlichen Variablen des Systems aufgespannt wird. Ein Zustand y_n wird beschrieben durch die Kombination der Werte sämtlicher Variablen des Systems nach der n -ten Iteration.

Unser Verfahren besteht im Allgemeinen aus n Iterationsschritten, $n \in \mathbb{N}_0$. Der Zustand des Verfahrens nach n Iterationsschritten sei y_n. f sei die Funktion, der gewählte Algorithmus, der bei den Experimenten jeweils von verschiedenen Parametern abhängt. Im einfachsten Fall ist f nur von n abhängig und der Zustand y_n berechent sich folgendermaßen:

$$y_n = f(n)$$

Verwendet das Verfahren ein Feedback, dann berechnet sich der nächste Zustand y_{n+1} über den Algorithmus f, der nun direkt abhängig ist vom vorhergehenden Zustand y_n.

$$y_{n+1} = f(y_n)$$

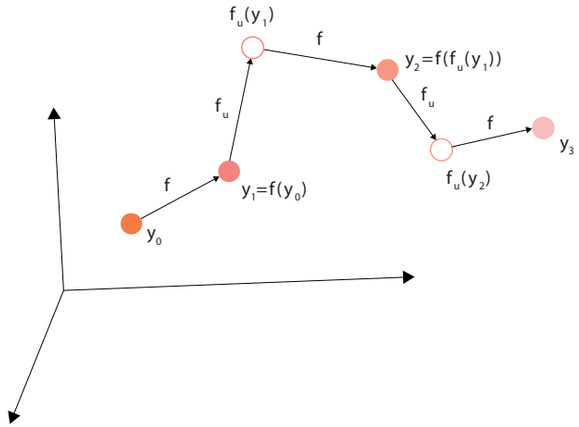


y_0 - y_n ... Zustände
f ... Funktion (beschreibt den Übergang von einem Zustand in den nächsten), der nächste Zustand ist immer abhängig vom aktuellen.

f_u bezeichnen wir als den User-Input, der übers iPad, Microscribe o.ä. oder auch direkt am, vom Verfahren erzeugten, Modell, gegeben werden kann. Er ist abhängig vom aktuellen Zustand y_n. Folglich lässt sich der User-Input darstellen mit $f_u(y_n)$. Hierbei muss erwähnt werden, dass der User-Input von vielen spezifischen Faktoren abhängig ist, wie Erfahrung, Tagesverfassung, usw. Das heißt, das f_u eigentlich nur für eine Person zu einem bestimmten Zeitpunkt oder in einem bestimmten Zeitraum gültig ist.

Verändert der Benutzer den vorherigen Zustand y_n, bevor er als Parameter in den Algorithmus f einfließt, dann ergibt sich:

$$y_{n+1} = f(f_u(y_n))$$



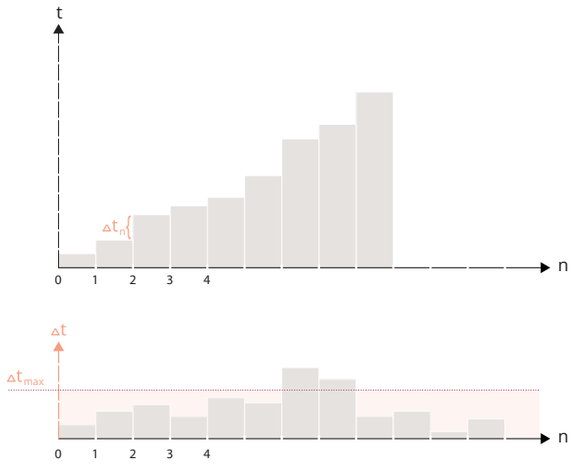
y_0 - y_n ... Zustände
f ... Funktion (beschreibt den Übergang von einem Zustand in den nächsten), der nächste Zustand ist immer abhängig vom aktuellen, und beinhaltet den User-Input, wenn es einen gibt.
f_u ... User-Funktion (beschreibt den User-Input), der User-Input ist abhängig vom aktuellen Zustand.

Diese Abbildung über das Verfahren mit direktem Feedback lässt sich ohne Ende komplexer gestalten. Der Algorithmus f könnte beispielsweise nicht nur den vom User veränderten letzten Zustand $f_u(y_n)$ als Parameter brauchen, sondern könnte außerdem noch abhängig sein von den Iterationsschritten n, und einem zweiten User-Input f_{u2}, der wiederum abhängig ist von y_n, folgendermaßen, dass gilt:

$$y_{n+1} = f(n, f_u(y_n), f_{u2}(y_n))$$

Zeit

Nach n Iterationsschritten unseres Verfahrens ist t_n Zeit vergangen. Das Δt, also die Zeit zwischen den Iterationsschritten, t_n - t_{n+1}, variiert ständig weil die auszuführenden Schritte und Weglängen für den Roboter auch ständig variieren.

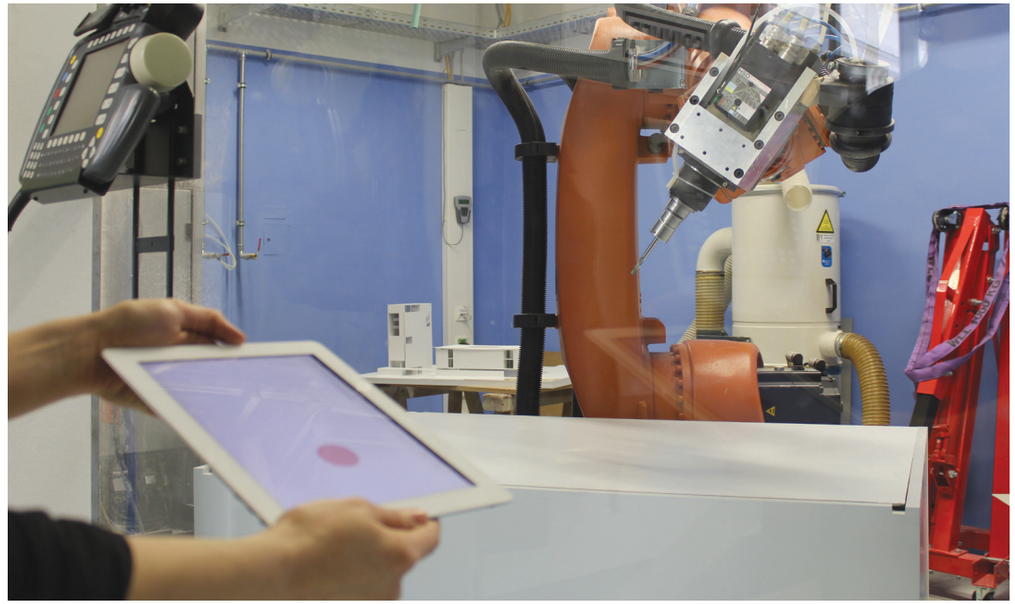


t ... Zeit
n ... Iterationsschritte
Δt ... Zeit zwischen den Iterationsschritten
Δt_{max} ... maximale Zeit zwischen den Iterationschritten

Wir bezeichnen Δt_{max} mit der maximalen Zeit, zwischen zwei Iterationsschritten, d.h zwischen einer möglichen Aktion, die ein Benutzer ausführt und einer erkennbaren Verarbeitung der Eingabe. Alles was über Δt_{max} liegt, ist für den Benutzer nicht mehr als Interaktion erfahrbar, Δt_{max} ist abhängig von den unterschiedlichen Verfahren (wie schnell der Benutzer sich in der Vorstellung ein Modell vom System bilden kann.)

Ein Beispiel

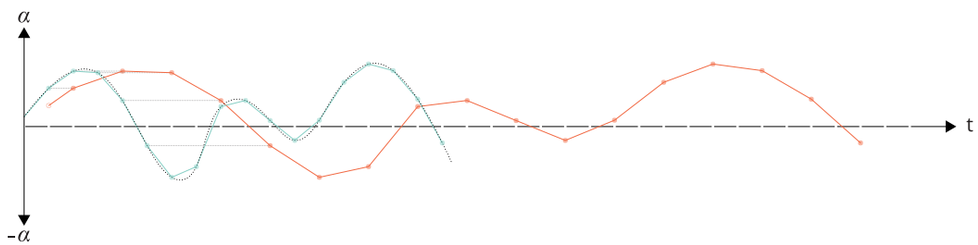
Visualisierung der Live-Steuerung des Roboters über die Auswertung der iPad Accelerometer-Daten und Vergleich vier unterschiedlicher Arten der Filterung und Berechnung des Weges für die Roboter-Werkzeugspitze, wobei Fall 1 und 2 die stattgefundenen Experimenten beschreiben, und Fall 3 und 4 mögliche Optimierungen beschreiben.



α ... Verdrehung des Koordinatensystems zum Basiskoordinatensystem (im Diagramm zweidimensional dargestellt)
t Zeit
rekonstruierte Trajektorie[8] vom iPad
Trajektorie von der Kuka-Werkzeugspitze
gemessene iPad-Acceleromter Werte
angefahrene Kuka-Positionen abhängig von der Rotation des iPads
Trajektorie vom iPad

8 Eine Trajektorie im Phasenraum bezeichnet die Menge aller Punkte, die von einem bestimmten Anfangspunkt aus die zeitliche Entwicklung des Systems bestimmen

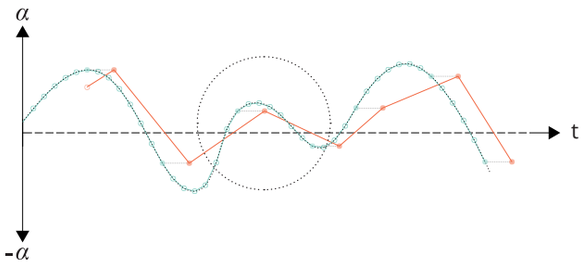
Fall 1:

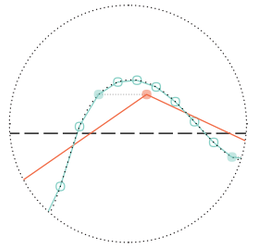


Die Accelerometer-Werte des iPads werden in regelmäßigen Zeitabständen (z.B. 0,2 sek) gemessen, in Kuka Positionen (E6Pos) umgerechnet und an den Kuka-Roboter geschickt. Dadurch, dass die Bewegungen des Roboters nicht so schnell

ausgeführt werden können wie das iPad bewegt wird und es Verzögerungen in der Übertragung gibt, kommt es zu einer zeitlichen Skalierung der Roboter-Trajektorie, aber alle Messpunkte der Bewegunges des iPads werden ausgeführt.

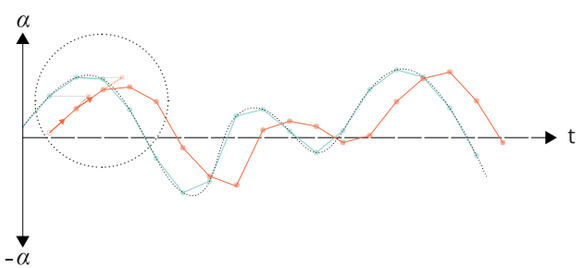
Fall 2:

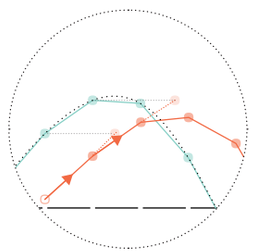




Zum Zeitpunkt, an dem der Kuka-Roboter einen Befehl fertig ausgeführt hat, werden die aktuellen iPad-Accelerometer-Daten abgefragt und die daraus errechnete Kuka-Position angefahren. Auch hier kommt es zu einer Verzögerung, abhängig von der Länge des Weges, den der Roboter hinter sich zu bringen hat, aber die Verzögerung summiert sich nicht mit der Anzahl der gemessenen Daten, weil das Messen und Schicken nicht in regelmäßigen Abständen passiert, sondern immer nur, wenn der Roboter aktuelle Messungen anfordert. Die daraus resultierende Trajektorie des Kuka-Roboters ist wesentlich ungenauer als im Fall 1, weil Teile der Bewegungen des iPads nicht erfasst werden, ist aber in der Benützung interaktiver erfahrbar als die genauere aber zeitlich mehr verzögerte Bewegung im Fall 1.

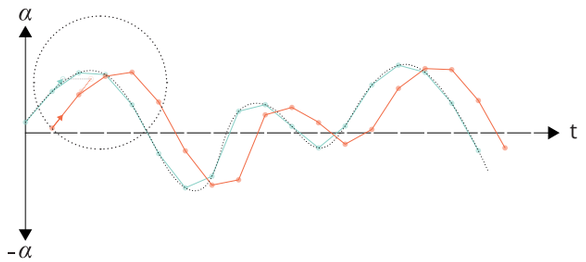
Fall 3:

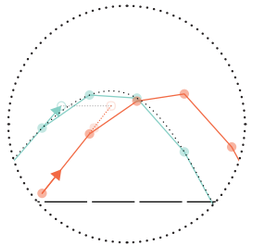




Die Accelerometer-Werte des iPads werden in regelmäßigen Zeitabständen gemessen, in Kuka Positionen (E6Pos) umgerechnet und an den Kuka-Roboter geschickt. Dadurch, dass die Bewegungen des Roboters nicht in derselben Geschwindigkeit ausgeführt werden können, wie das iPad bewegt wird, bewegt der Roboter das Werkzeug nur ein Stück in die gewünschte Richtung, d.h. der Roboter erreicht die gewünschte Position nicht unbedingt vollständig, befindet sich aber am Weg dorthin. Die Bewegung wird unterbrochen, sobald die nächste Messung erfolgt, usw. Die daraus resultierende Trajektorie des Kuka-Roboters ist folglich nur eine Annäherung an den Weg des iPads, aber die „Gleichzeitigkeit" der Prozesse bleibt erhalten.

Fall 4:





Die Accelerometer-Werte des iPads werden in regelmäßigen Zeitabständen gemessen, es werden aber nicht die direkten Messwerte an den Kuka-Roboter geschickt, sondern aus den Werten werden Richtungsvektoren ermittelt, aus denen die eigentlichen Zielpunkte errechnet werden (vgl. als Bespiel: Wenn ein Ball geworfen wird, wird derjenige, der den Ball fangen soll, nicht an die Stelle laufen, wo er geworfen wurde, sondern wo er voraussichtlich landen wird.) Gleich wie im Fall 3 bewegt der Roboter das Werkzeug nur ein Stück in die gewünschte Richtung, bis die Bewegung vom nächsten Befehl unterbrochen wird, die daraus resultierende Trajektorie des Kuka-Roboters ist aber folgend eine noch genauere Annäherung an den Weg des iPads.

5. 14 Werkzeuge und Durchlichttisch

Für die verschiedenen Experimente, die im Kapitel Fallstudien beschrieben werden, werden spezifische Werkzeuge für den Roboter benötigt, von denen hier einige davon beschrieben werden:

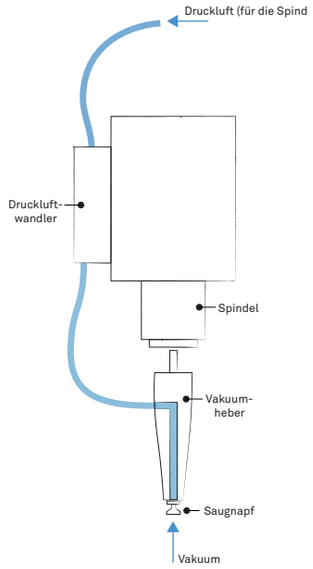
Der Vakuumheber

Der Vakuumheber wurde entwickelt, um kleine Bauelemente mit dem Roboter aufnehmen und ablegen zu können.

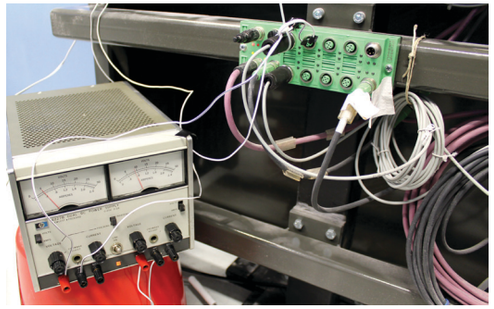
Dabei wird über ein Gerät die Druckluft, die vom Roboter kommt, in Vakuum umgewandelt und in das Werkzeug geleitet, welches wie ein Fräser in der Spindel eingespannt werden kann.

Die verschiedenen Saugnäpfe an der Spitze erlauben eine gewisse Toleranz (bis zu 2mm) beim Positionieren, für das Ansaugen der Bauteile.

Das Ein- und Ausschalten des Vakuums erfolgt über das Setzen einer Variable im KRL-Programm.



Verschiedene Saugnäpfe

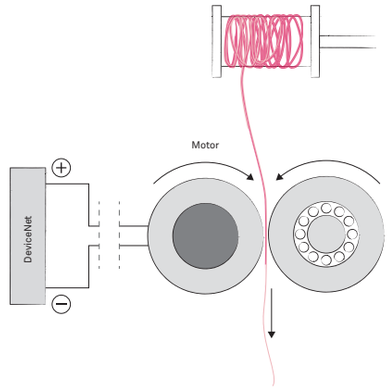


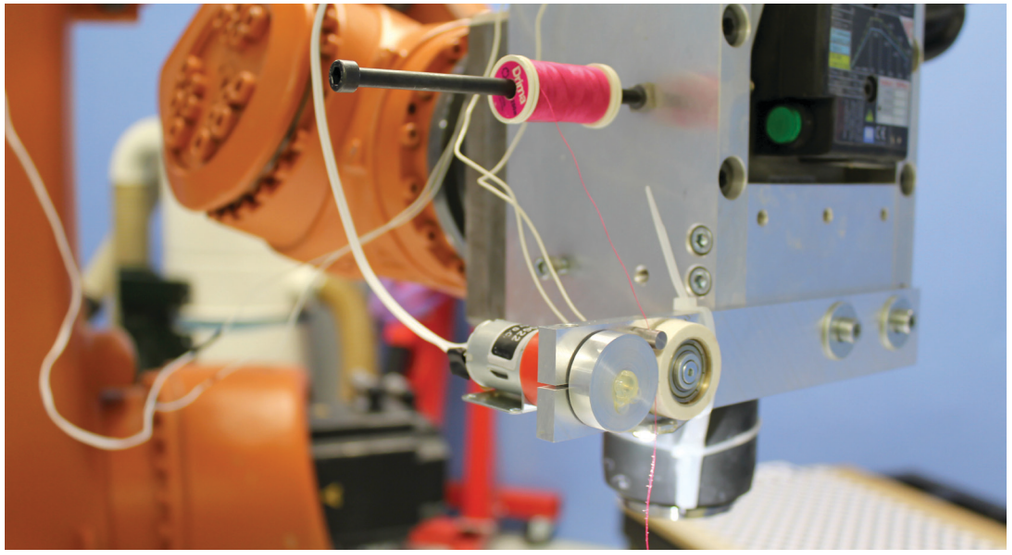
DeviceNet Bus am Roboter

Die Vorrichtung zum Abwickeln des Fadens

Um einen Faden von einer Spule abzuwickeln, wurde eine Vorrichtung gebaut, bei der der Faden über zwei sich drehende Rollen nach unten gezogen wird. Eine der beiden Rollen wird über einen Motor angetrieben, die andere dreht sich mit, und transportiert so den Faden.

Das Ein- und Ausschalten des Motors erfolgt über einen DeviceNet Bus, der am Roboter angeschlossen ist, und kann wiederum über eine Variable im KRL-Programm gesteuert werden.

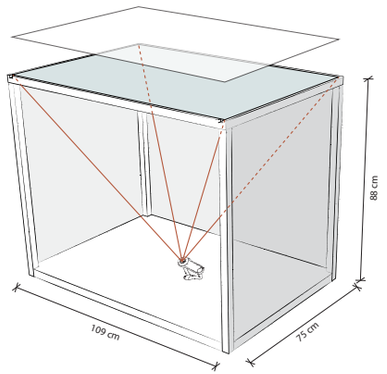




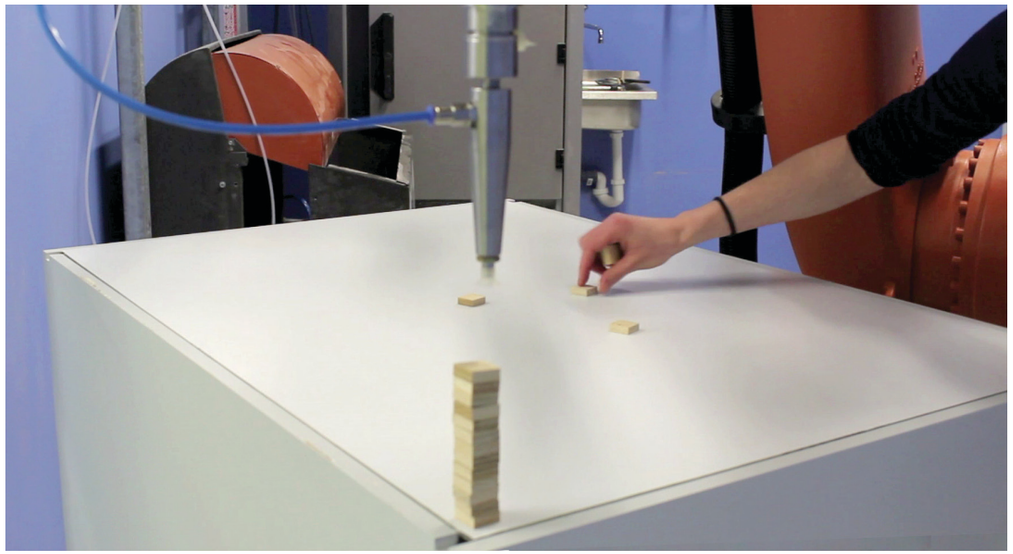
Durchlichttisch

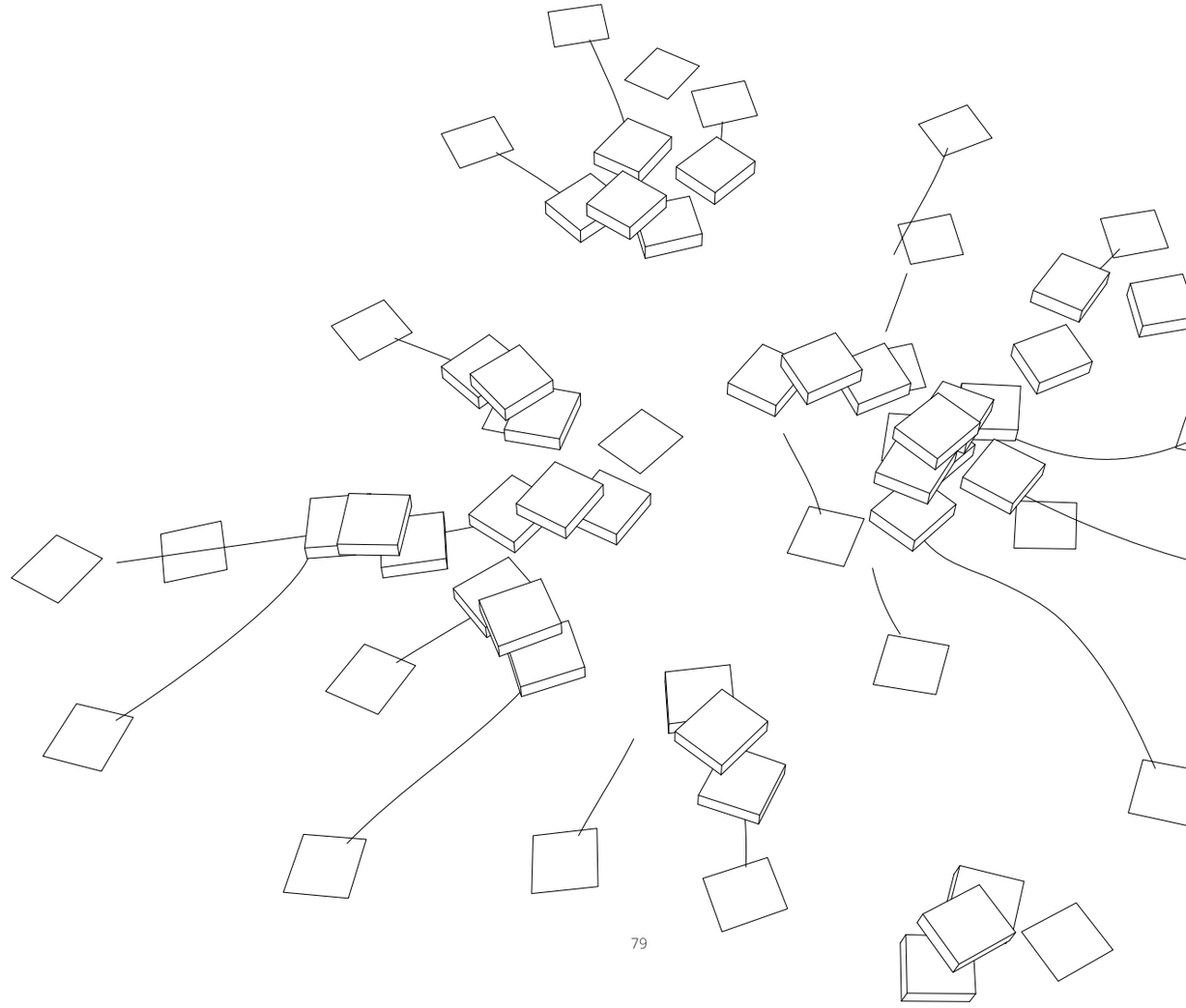
Da das Tracking von oben über einer Platte nicht möglich ist, weil der Roboter sich ständig darüber bewegt, wird von unten durch eine Glasplatte getrackt.

Damit die Glasplatte und die Kamera in fixer Position zueinander gehalten werden, gibt es einen Rahmen mit Platten an den Seiten und am Boden zur Aussteifung und zur Verdunkelung. Dadurch werden gleichbleibende Lichtbedingungen (Licht nur von oben) für die Kamera geschaffen, die an der Bodenplatte fixiert ist. Außerdem ist die Glasplatte mit einer transluzenten Folie abgedeckt, damit nur mehr der Schatten von Objekten, die direkt auf der Glasplatte liegen, durchdringen kann und so als grafischer Filter dient.



Kamerabild





6 Fallstudien

6. 1 Experimente und Fallstudien

Darstellung

In diesem Kapitel werden die Experimente und Fallstudien vorgestellt, welche verschiedene Versuche sind, Eigenschaften des Systems zu untersuchen. Erkenntnisse aus den einzelnen Experimenten flossen in die Weiterentwicklung und Adaption der Software mit ein, deshalb werden die einzelnen Stadien und Entwicklungsschritte mit den dazugehörigen Experimenten beschrieben.

Jedes Experiment wird auf die zuvor behandelten Themen Interaktivität, Feedback, Zeit und Materialität untersucht und die Schlussfolgerungen dargestellt.

Die Fallstudien wurden dokumentarisch in kurzen Videos festgehalten. Man findet die Links zu den Videos in der jeweiligen Seitenkolumne.

6. 2 Kommunikationstest iPhone - Roboter

Das erste Programm dient zur Überprüfung der funktionierenden Verbindung zwischen Roboter, Server und iPhone.

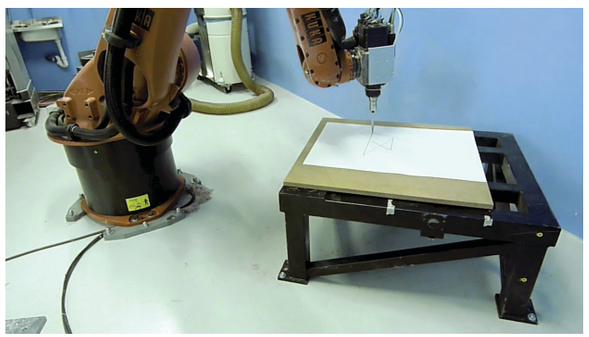
Werkzeug am Roboter: Filzstift

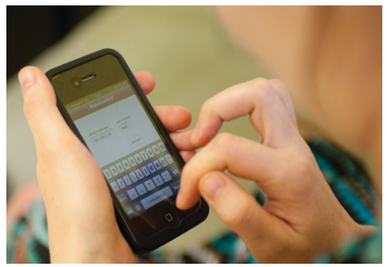
Externes Sensorsystem: iPhone

Abbildung: Die Eingabe über Display des iPhones (960x640px, 8.89cm Bilddiagonale) wird abgebildet auf den Bereich eines Blatt Papiers (80x50 cm).

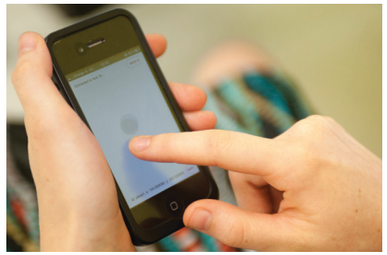
Ablauf: 

Am Display eines iPhones kann ein Kreis verschoben werden. Die Koordinaten des Kreismittelpunktes werden gesendet, sobald man den Button „send“ aktiviert. Die gesendeten zweidimensionalen Koordinaten werden am Computer, auf dem die Server-Applikation läuft, empfangen, mit einer bestimmten z-Koordinate erweitert, und auf die eingemessene Basis des Roboters abgebildet. Die übersetzten Koordinaten werden in eine XML Nachricht verpackt und an den Roboter gesendet. Dieser bewegt die Werkzeugspitze an die gesendeten Koordinaten und hinterlässt mit dem Filzstift eine Linie am Blatt Papier. Der Roboter kann immer erst eine neue Koordinate empfangen, wenn die vorherige Bewegeung fertig ausgeführt wurde. Der Ablauf kann so lange wiederholt werden, bis man sich entscheidet das Programm abzubrechen.

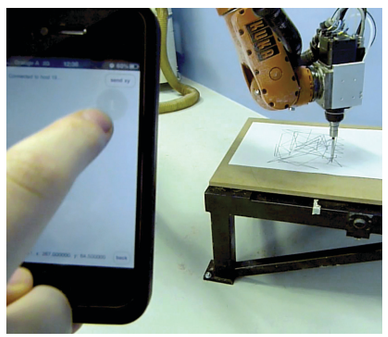




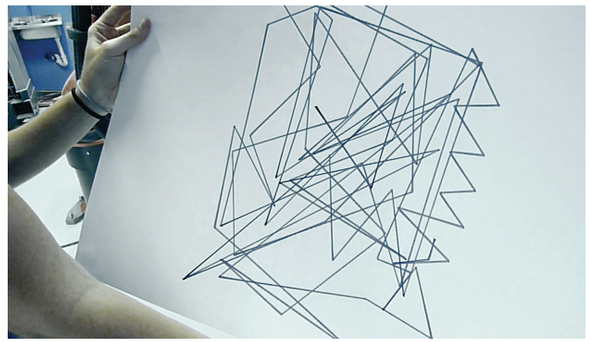
Applikation am iPhone: Verbindungsaufbau mit dem Server

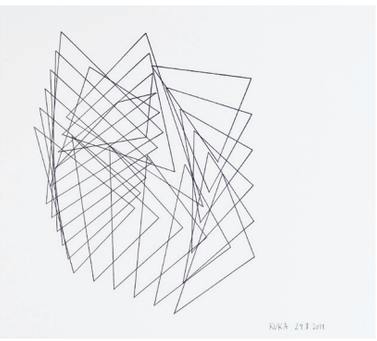


Senden von x/y Koordinaten des iPhones an den Server



Vom Server werden die empfangenen x/y Koordinaten des iPhones an den Roboter gesendet

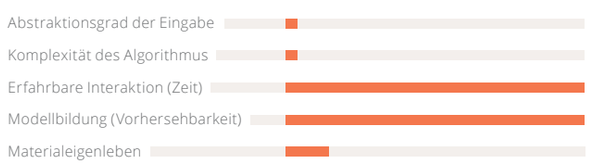




Beobachtungen:

Der Abstraktionsgrad der Transformation der Eingabe ist minimal. Die gesendeten Koordinaten des iPhones werden direkt auf den Roboter übertragen. Die Zeichnung ist eine direkte Abbildung des Verschiebens des Kreises am Display des iPhones in skalierter Form.

http://vimeo.com/37556938



Mathematische Darstelllung:

$$f_u(s_n) = P(x; y)$$

$$\text{mit } x, y \in \mathbb{N}_0,\ 0 \leq x \leq 640,\ 0 \leq y \leq 960$$

$s_0 = 0$ Zustand 0, ein leeres Blatt Papier

$s_{n+1} = f(f_u(s_n))$, s ... Zustand, n ... Iterationsschritte

Der nächste Zustand errechnet sich über eine Funktion, die den User-Input als einzigen Parameter nimmt.

6. 3 Punktefelder

Dieses Programm dient zur Überprüfung der Größe der XML-Datenpakete, die über das KRL-Programm am Roboter empfangen werden können.

Werkzeug: Filzstift

Externes Sensorsystem: iPhone

Abbildung: Die Eingabe über Display des iPhones (960x640px, 8.89cm Bilddiagonale) wird abgebildet auf den Bereich eines Blatt Papiers (80x50 cm).

Material: Papier und Pinselmarker: weiche Spitze des Pinselmarkers erlaubt ein verschieden starkes Eindrücken auf dem Papier.

Der Algorithmus

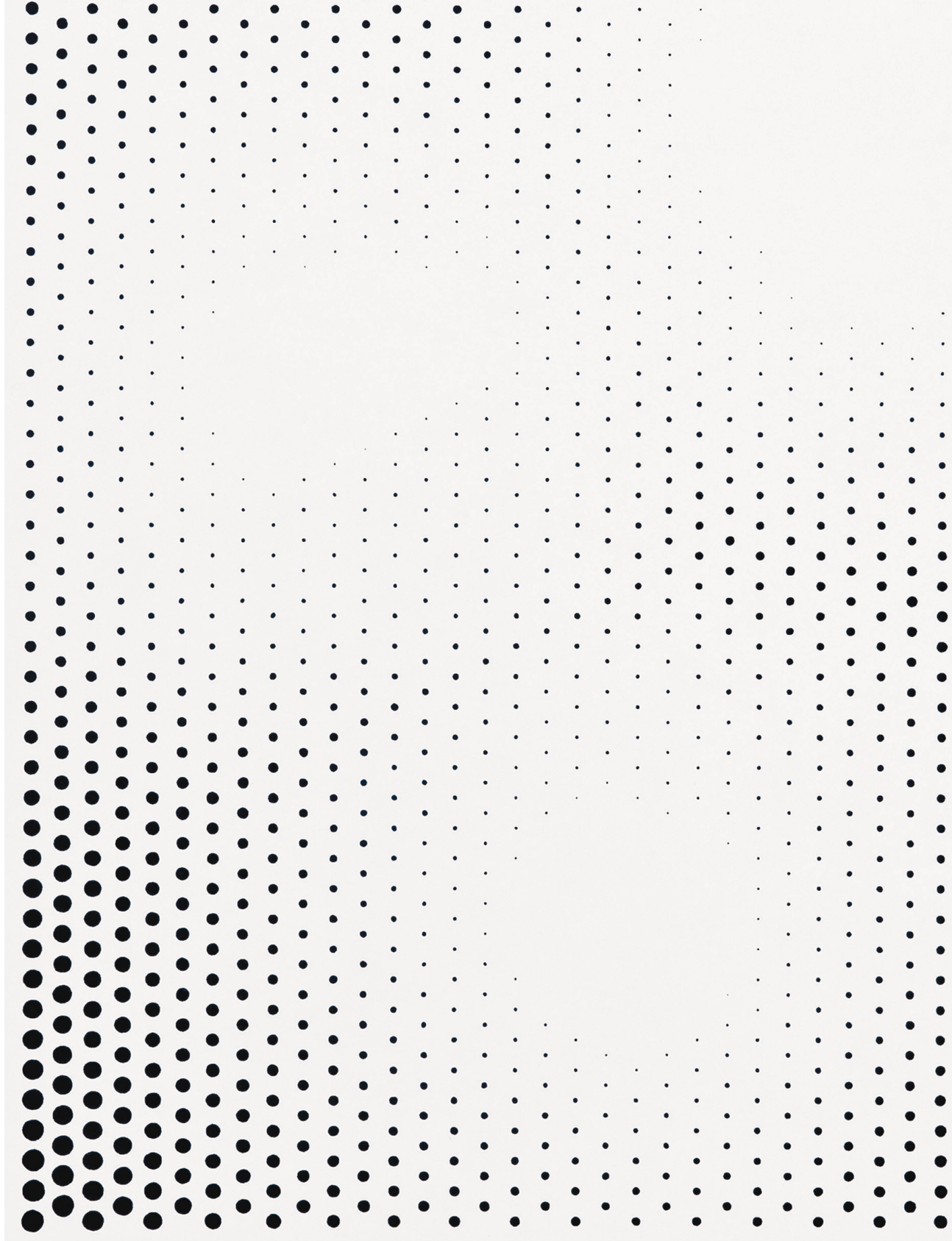
In der xy-Ebene wird ein virtuelles Punkteraster erzeugt. Die xy-Koordinate ist festgelegt, die z-Koordinate ist von den Abständen zu drei Attraktoren abhängig, je näher der Abstand zu einem Attraktor ist, desto höher ist der z-Wert. Die Positionen der drei Attraktoren werden über die Eingabe am iPhone definiert.

Definierte Parameter: Dichte und Größe des Punkterasters, Wertebereich innerhalb der der z-Wert variieren kann.

Variable Parameter: die Position der Attraktoren

Ablauf: 

Am Display eines iPhones können drei Kreise verschoben werden. Die xy-Koordinaten der Kreismittelpunkte können über einen Button an den Server geschickt werden. Die empfangenen Positionen werden auf dem Punkteraster abgebildet und definieren die z-Koordinate der einzelnen Punkte (wie tief der am Roboter eingespannte Pinselmarker in das Papier eingedrückt wird).



Der Algorithmus berechnet die Koordinaten aller Punkte im Raster für den Roboter, und fügt jeder Koordinate die Anfahrts und Rückfahrtsposition zum Absetzen des Filzstiftes hinzu. Jeder zu zeichnende Punkt benötigt also 3 Roboterpositionen, was bei einem 48x32 großen Punkteraster zu 4608 Befehlen führt.

Die Befehle können nicht alle auf einmal vom Roboter empfangen werden, denn die Größe seines Buffers ist begrenzt. Deshalb müssen die Positionen in Paketen zu je sechs oder weniger gesendet werden, und erst wenn der Roboter die Positionen eines Pakets fertig angefahren hat, können vom Server aus neue Positionen gesendet werden.

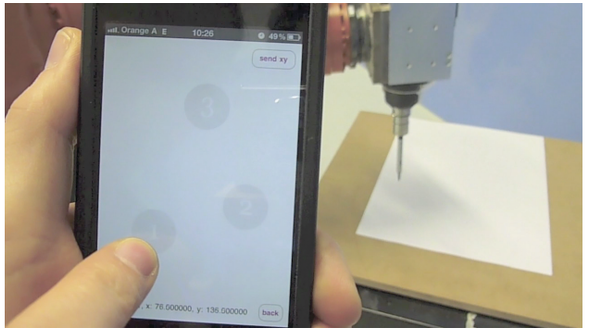
Der Roboter beginnt nach Erhalt des ersten Pakets das Punkteraster zu zeichnen, und die Punkte weisen abhängig von der z-Koordinate verschiedene Größen auf.

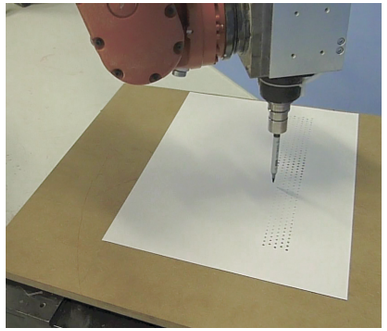
Das Verfahren endet, wenn alle Punkte gezeichnet, d.h. alle Befehle ausgeführt worden sind.

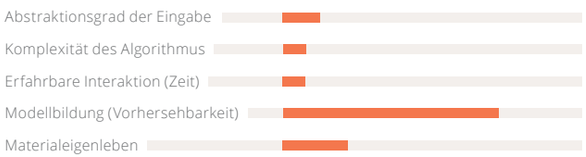
Beobachtungen:

Das Verfahren ist linear, es gibt einen User-Input, aber nur zu Beginn des Programms, sodass man eigentlich nicht von einem interaktiven Prozess sprechen kann. Alle Positionen für das vollständige Bild werden zum Zeitpunkt berechnet, sobald der User-Input erfolgt, und können während des Zeichnens auch nicht mehr verändert werden.

http://vimeo.com/37557147







Mathematische Darstelllung:

$$f_u = P(x;y)$$
$$\text{mit } x,y \in \mathbb{N}_0,\ 0 \leq x \leq 640,\ 0 \leq y \leq 960$$

$n = 1$, n ... Iterationsschritte
$s_n = f(f_u)$, s ... Zustand

Dieser Algorithmus besteht nur aus einem Iterationsschritt, der von der User-Eingabe abhängig ist. Das Verfahren beinhaltet kein Feedback.

6. 4 Dendritenwachstum

Im Unterschied zum vorherigen Experiment wird der Prozess nun dahingehend erweitert, dass der Server während der Laufzeit des Programms weitere Eingaben verarbeiten kann, welche den Programmablauf und die Ausführung durch den Roboter verändern können.

Statt des iPhones wird nun ein iPad verwendet, welches sich durch ein größeres Display unterscheidet. Über das Display des iPads erfolgen die User-Eingaben, parallel dazu werden die Ergebnisse der Berechnungen des Algorithmus am Display visualisiert, während der Roboter die Punkte zeichnet.

http://vimeo.com/39110298
Kuka-iPad: Dendritic Growth Algorithm 1

http://vimeo.com/39120131
Kuka-iPad: Dendritic Growth Algorithm 2

http://vimeo.com/40413004
iPad Visualization of Dendritic Growth Algorithm

Dendritenwachstum Algorithmus:

Dendriten nennt man gewächsartige Strukturen, die sich beispielsweise wie die Adern in einem Blatt einer Pflanze verzweigen. Diese Adern werden in Form von aufeinander folgenden Punkten gezeichnet, die als „Nodes“ bezeichnet werden.

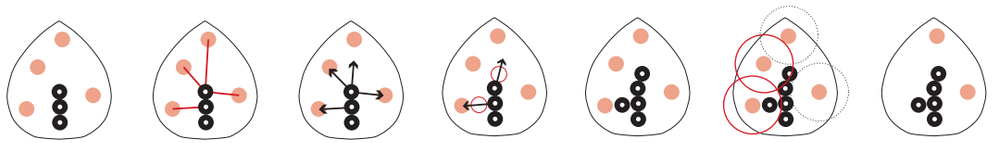
Zu Beginn wird die erste Node, der Anfang der Ader, definiert. Um diese Node werden zufällige Punkte (Sources) in einem gewissen Radius r verteilt, die zur Berechnung der nächsten Node dienen. Es gelten folgende Regeln[1]:

1. Lösche alle Sources deren Abstand zu den Nodes kleiner als der definierte Mindestabstand kd ist.

2. Berechne den Abstand von den Sources zu allen Nodes und weise diesen die Nodes zu, die ihnen am nächsten sind.

3. Für jede Node, der Sources zugewiesen worden sind, wird ein resultierender Richtungsvektor aus den Abstandsvektoren zu den Sources berechnet.

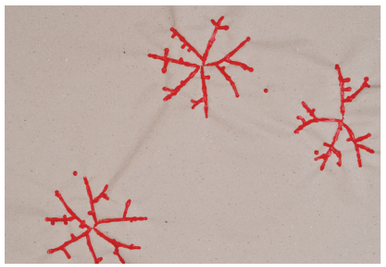
4. Neue Nodes werden erzeugt, in der Richtung des zuvor errechneten Richtungsvektors mit dem Abstand d zur vorherigen Node. Die Stärke jeder neu erzeugten Node wird um den Fakor e reduziert.



vgl. Illustration in http://algorithmicbotany.org/papers/venation.sig2005.html, S.5

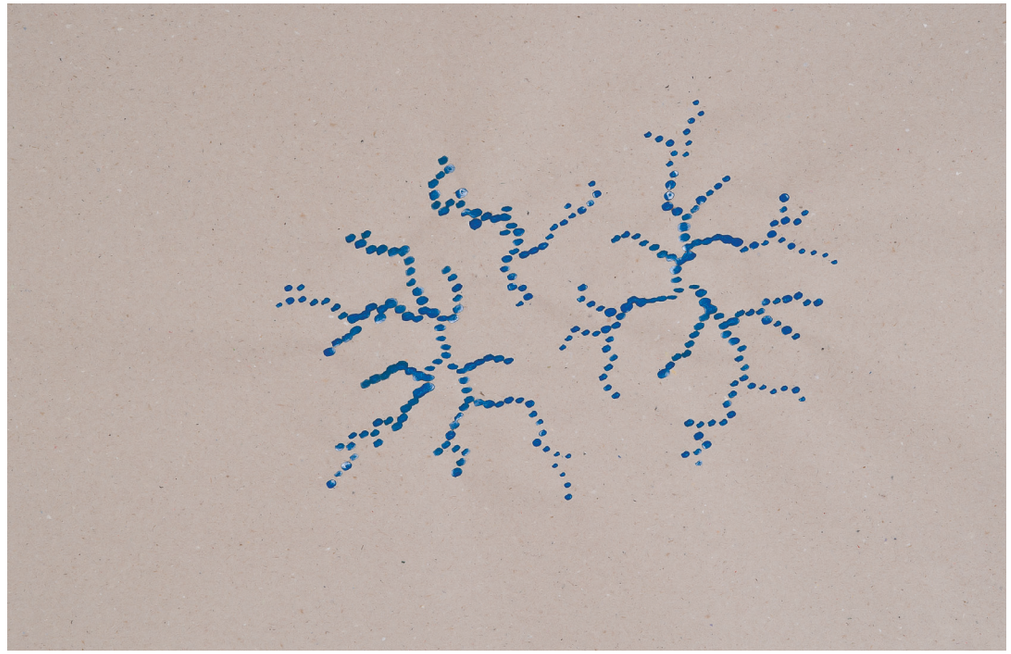
Die Dichte der Nodes, die Art oder die Menge der Verzweigungen können über verschiedene Parameter entweder vorher oder auch während der Laufzeit des Programmes gesteuert werden: der Radius r in welchem um die Node Sources erzeugt werden, der Mindestabstand kd zwischen Source und Node, der Abstand d zwischen zwei Nodes, und der stärkereduzierende Faktor e.

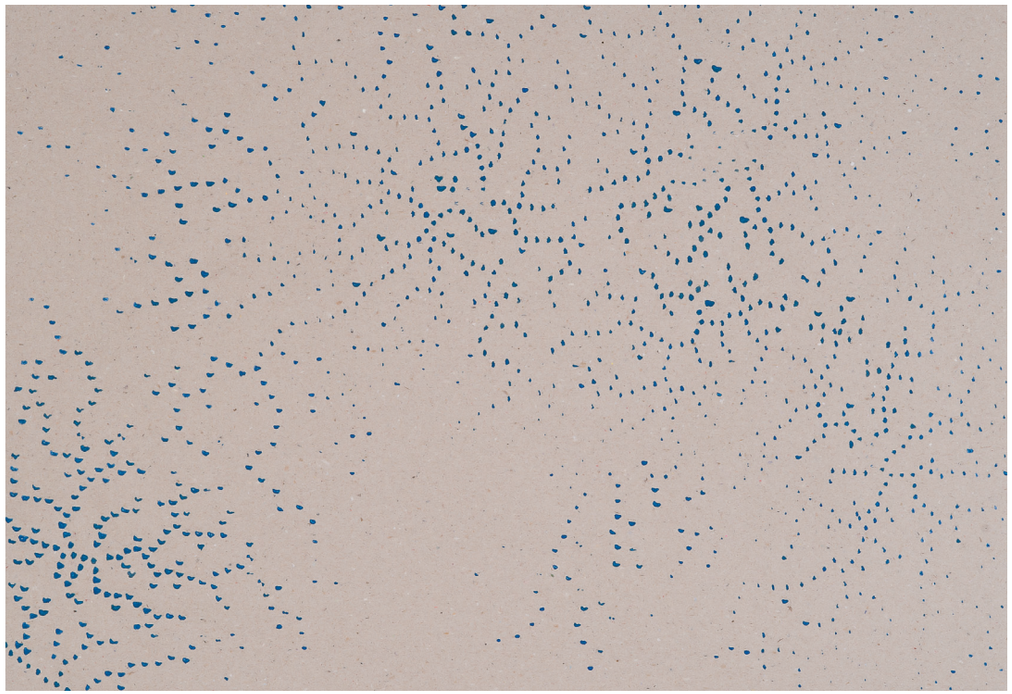
Die Parameter, die während der Laufzeit vom Benutzer in den Algorithmus einfließen, sind die Koordinaten der jeweiligen Anfangspunkte der Blattadern.



Sehr eng gesetzte Nodes, die Punkte verfließen zu einer Linie

1 Runions, Adam / Fuhrer, Martin / Lane Brendan / Federl, Pavol / Rolland-Lagan, Anne-Gaëlle/ Prusinkiewicz, Przemyslaw (2005): Modeling and visualization of leaf venation patterns, http://algorithmicbotany.org/papers/venation.sig2005.pdf, in http://algorithmicbotany.org, September 2011





Werkzeug: Pinsel

Externes Sensorsystem: iPad

Abbildung: Display des iPads (2048x1536 px, 24.64 cm Bilddiagonale) wird abgebildet auf einen Bereich eines Blatt Papiers (70x40 cm)

Material: Papier, Aquarellfarbe, Pinsel

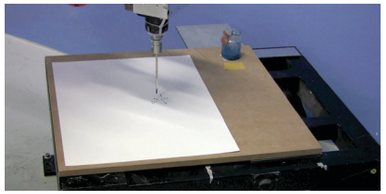
Ablauf:

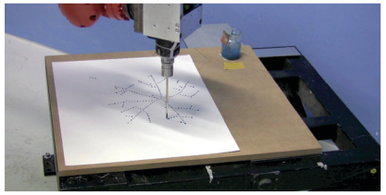
Das Programm ist so aufgebaut, das vom Benutzer immer die Anfangspunkte gesetzt werden können, von denen aus die Struktur zu wachsen beginnt.

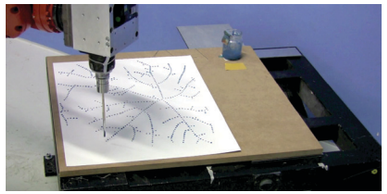
Am Display des iPads kann ein Kreis verschoben werden, und über den Button „Send“ können die Koordinaten des Kreismittelpunktes an den Server geschickt werden. Die vom Server empfangenen zweidimensionalen Koordinaten fließen in den Algorithmus als Anfangspunkte der Dendriten-Adern ein, d.h. an jeder neu gesendeten Koordinate beginnt ein neues Gewächs, solange sich an dieser Position noch nichts befindet.

Sobald der erste Anfangspunkt gesendet wird, beginnt der Roboter die Struktur zu zeichnen. Sein Werkzeug ist ein Pinsel, der immer wiederholt in ein Farbgefäß eingetaucht werden muss, um wieder weiter Farbe auftragen zu können.

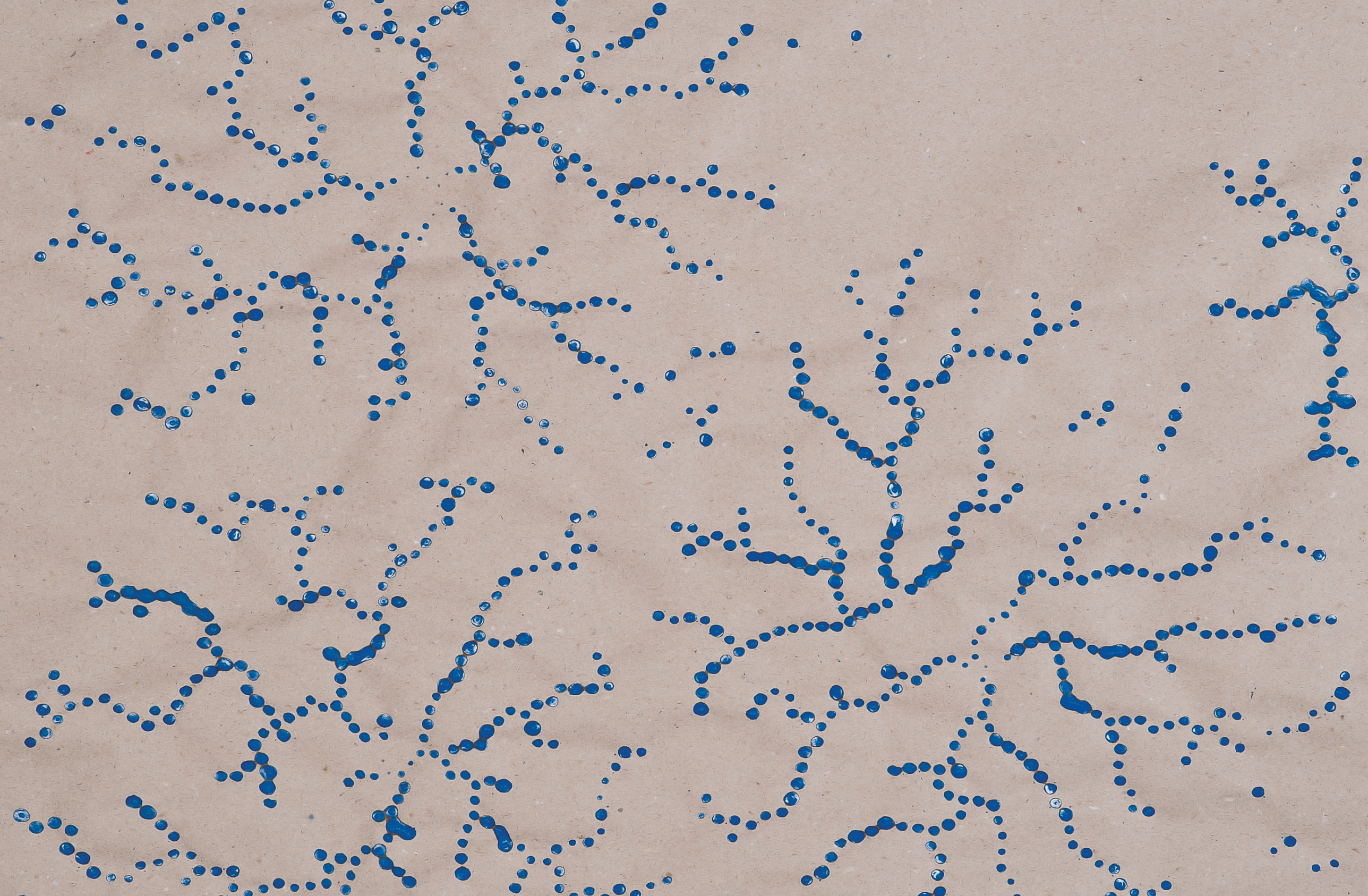
Das Verfahren endet, wenn vom Algorithmus keine neuen Nodes berechnet werden können, oder der User das Programm abbricht.







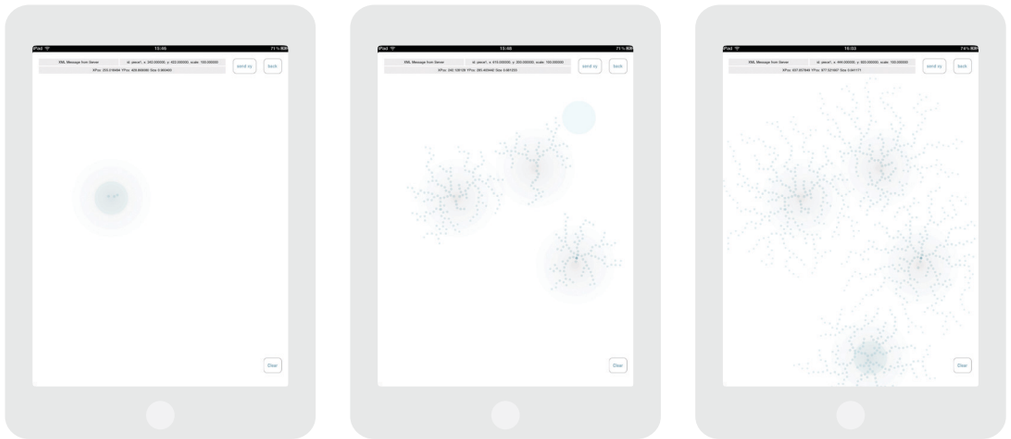
Zeichnen der Punkte der Dendriten-Struktur vom Roboter



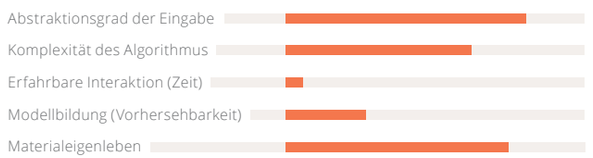
Beobachtungen:

Was man im Algorithmus vordefiniert, wirkt sich sehr stark auf das aus, was gezeichnet wird (Dichte, Verteilung usw.). Als User hat man keine wirklichen Interaktionsmöglichkeiten, außer dem Setzen der Anfangspunkte, von wo aus die Verzweigungen gezeichnet werden. Dazu gibt es eine große zeitliche Verzögerung zwischen dem Senden der Koordinate vom iPad, und dem Zeichnen der Punkte um die gesendete Koordinate vom Roboter. D.h. es ist schwierig für den User, das, was gezeichnet wird, mit dem zu verbinden, was aus seinen Inputs hervorgeht.

Die Visualisierung des Algorithmus am iPad kann viel schneller passieren als das Zeichnen der einzelnen Punkte vom Roboter, und erleichtert dem User ein Erfahren seiner Interaktion, aber sie unterscheidet sich sehr vom gezeichneten Ergebnis am Blatt Papier, durch die Unregelmäßigkeiten, die entstehen, durch die tropfende Farbe oder der, über die Zeit sich verändernden Pinselspitze.



Visualisierung der zu zeichnenden Punkte am iPad



Mathematische Darstelllung:

$$f_u(s_n) = P(x; y)$$

$$\text{mit } x, y \in \mathbb{N}_0,\ 0 \leq x \leq 1536,\ 0 \leq y \leq 2048$$

$s_0 = 0$ Zustand 0, ein leeres Blatt Papier

$s_{n+1} = f(s_n)$ bzw. $s_{n+1} = f(s_n, f_u(s_n))$

s ... Zustand, n ... Iterationsschritte

Der nächste Zustand errechnet sich über eine Funktion, die immer vom vorherigen Zustand abhängt, und die über den User-Input verändert werden kann.

6. 5 Kopierfräsen

Mit einem Microscribe 5-Achs Koordinatenmessarm und einer 3-Achs Fräse.

In diesem Versuch wurde nicht der Roboter, sondern die 3-Achs-Fräse mit der Steuerung LinuxCNC verwendet. Auch hier werden über eine Netzwerkverbindung Daten an die Steuerung LinuxCNC geschickt, nur in diesem Fall in Form von G-Code.

Das Empfangen der Daten erfolgt hier deutlich schneller als am Roboter, das ermöglicht ein häufigeres und schnelleres Senden.

3D-Daten, die über den Microscribe Koordinatenmessarm erfasst werden, werden während der Betätigung eines Fußschalters ungefiltert über die Serverapplikation an die Fräse geschickt, d.h. alle Bewegungen, die man in dieser Zeit aufnimmt, werden von der Fräse im eingemessenen Bereich ausgeführt. Am Messarm wird ein Nullpunkt definiert, der dann dem Nullpunkt auf der Fräse entspricht, die erfassten Koordinaten selbst werden nicht transformiert.

Werkzeug: 3mm Schaftfräser

Exteres Sensorsystem: Microscribe Koordinatenmessarm

Abbildung: Die Eingabe über den Microscribe Koordinatenmessarm der x, y und z-Koordinaten wird 1:1 abgebildet auf den eingemessenen Bereich der BZT 3-Achs Fräse.

Material: Polyurethanschaum

Ablauf: 

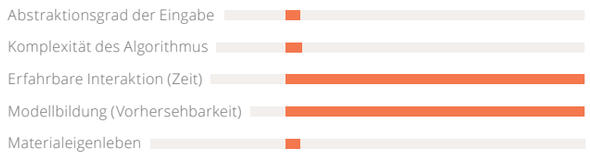
Ein Kaffeelöffel wird mit dem Microscribe Koordinatenmessarm gescannt, und parallel dazu in Echtzeit aus einem Polyurethanblock ausgefräst. Die Fräsbahnen entsprechen genau den Bewegungen, die man mit dem Messarm ausführt.

Scannen und gleichzeitiges Fräsen eines Kaffeelöffels

http://vimeo.com/40415668

Beobachtungen:

Interessanter könnte der Versuch gestaltet werden, wenn man die erfassten Koordinaten einer Transformation unterzieht, z.B das Objekt skaliert, gespiegelt oder verbogen fräst.



Mathematische Darstelllung:

$$f_u(s_n) = P(x, y, z)$$

mit $x, y, z \in \mathbb{R}$, Arbeitsbereich des Koordinatenmessarms

$s_{n+1} = f(f_u(s_n))$, s ... Zustand, n ... Iterationsschritte

Original und Kopie, Fräsbahnen entsprechen der Bewegung der Spitze des Koordinatenmessarms über den Löffel

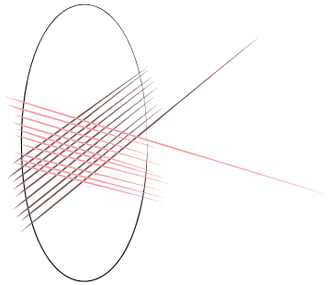
6. 6 Umwickeln von dreidimensionalen Körpern

(nur in der Simulation)

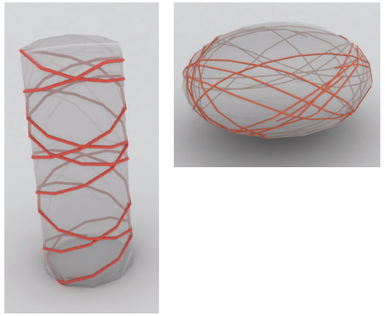
Die vorangegangenen Studien zeigten auf, dass die Interaktion im System, also die Eingabe eines/r Benutzers/in und die darauf folgende Reaktion erfahrbarer für den/diejenige sein muss. Folglich wurde nach einem Aufbau gesucht, in der eine Eingabe über das iPad sich direkt und sofort auf die Bewegungen des Roboters auswirkt.

Die Messdaten des Accelerometers des iPads sollten verwendet werden (siehe Kapitel Umsetzung 5.9), um den Steigungswinkel eines Fadens zu verändern, der vom Roboter geführt einen dreidimensionalen Körper umwickelt. Dabei würde die Neigung der Roboterspitze der Neigung des iPads entsprechen und wäre deshalb sofort ersichtlich.

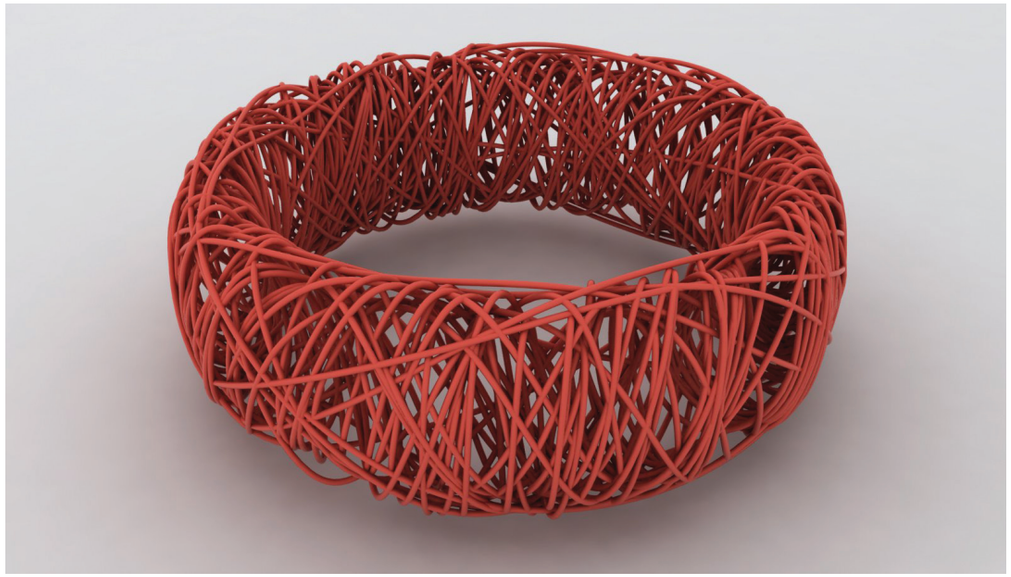
In der Simulation wurde ausgetestet, wie die Beeinflussung des Steigungswinkels sich auf die Windung einer Polylinie um verschiedene 3D-Körper auswirken kann.



Unterschiedlicher Steigungswinkel - unterschiedliches Wickelmuster

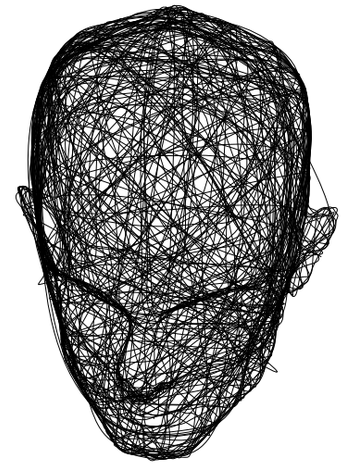


Umwicklung eines Zylinders und eines Ellipsolides



Die Resultate dieser Tests führten leider nicht zum gewünschten Ergebnis: Ein Verändern des Parameters war am umwickelten Körper nicht ablesbar.

Trotzdem war dieser Versuch für die Arbeit wichtig, da man erkennen kann, dass die Randbedingungen so stark sind, dass eine Interaktion in diesem Rahmen nicht sinngemäß stattfinden kann. Bei diesem Beispiel würde man eine Veränderung der Stellung der Roboterspitze abhängig von der Orientierung des iPads sofort bemerken und als Eingabe erkennen, jedoch würden diese das Ergebnis des umwickelten Körpers kaum verändern, da der Weg des Fadens nur von der Geometrie des zu umwickelnden Körpers abhängt.



Umwickeln verschiedener dreidimensionaler Geometrien

6. 7 Manipulation am Objekt

In den folgenden Versuchen geht es darum, den User-Input nicht mehr über ein externes Device zu empfangen, sondern um die Möglichkeit Strukturen, die gebaut werden, direkt zu manipulieren. Dazu wird eine Umgebung benötigt, in der man Parameter aus der realen Situation extrahieren kann.

In den folgenden Experimenten wird versucht, über eine Kamera (siehe Kapitel 4.13) Parameter des aktuellen Zustandes (z.B Helligkeit) zu messen, und diese zu verwenden, um den Programmablauf zu beeinflussen.

6. 8 Türme bauen

Werkzeug: Vakuumheber

Externes Sensorsystem: Kamera

Abbildung: Kamerakoordinaten werden auf die eingemessene Basis des Roboters abgebildet

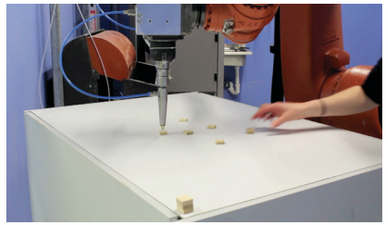
Material: Holzwürfel 25x25x8mm

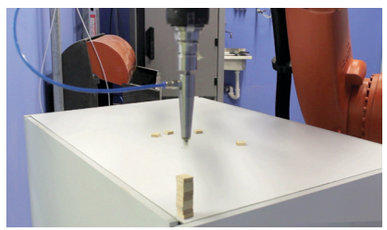
Ablauf:

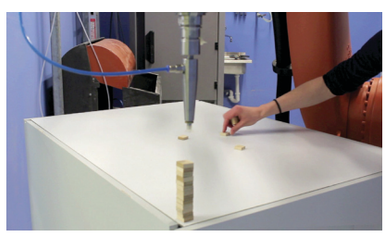
Im ersten Experiment werden auf einer Platte, die von unten getrackt wird, Bausteine zufällig angeordnet und über die Kamera werden Anzahl, Position und Orientierung der Bausteine ermittelt. Dann werden einzelne Bausteine ausgewählt, vom Roboter abgeholt und zu einem Turm aufgestapelt. Es können, während das Programm läuft, immer wieder Bausteine dazugelegt werden.

Nach jedem Ablegen eines Bausteines wird über den Server ein neues Kamerabild angefordert, und die vorhandene Situation neu ausgewertet, damit neu dazugelegte Bausteine in

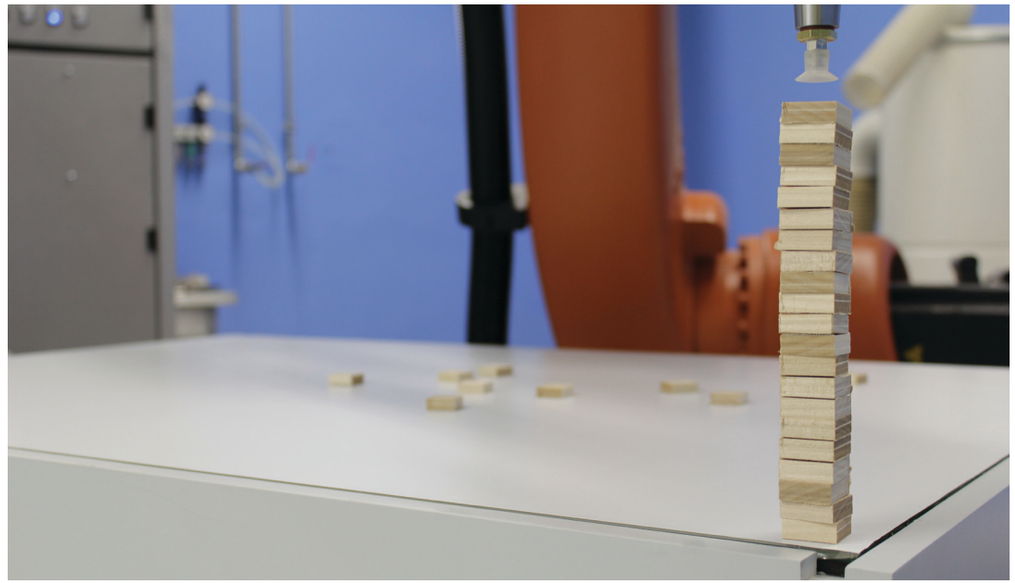
http://vimeo.com/37549013
Optical Tracking - Picking up a building block







Turm stapeln mit dem Roboter

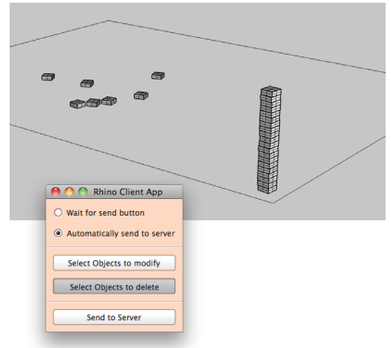


den Algorithmus einfließen können. D.h. ein Iterationsschritt ist darauf beschränkt, über die Kamera einen Baustein aus allen vorhandenen auszuwählen (nach dem Zufallsprinzip oder anderen Kriterien wie z.B Farbe), abzuholen und abzulegen, und beginnt dann wieder von vorne.

Beobachtungen:

Von der Kamera können nur Veränderungen in 2D erfasst werden, d.h man kann Bausteine am Tisch wegnehmen oder dazulegen, aber nicht vom Turm. Aus diesem Grund werden die Bausteine, die in eine Struktur eingebaut werden (in diesem Fall ein Turm), im Programm virtuell mitgespeichert, und müssen, wenn man einen Baustein vom real gebauten Turm wegnimmt, vom virtuellen Speicher auch wieder entfernt werden.

Dieses Vorgehen erlaubt Fehler, da es während des Programmablaufs immer eine Überprüfung der Realsituation gibt.



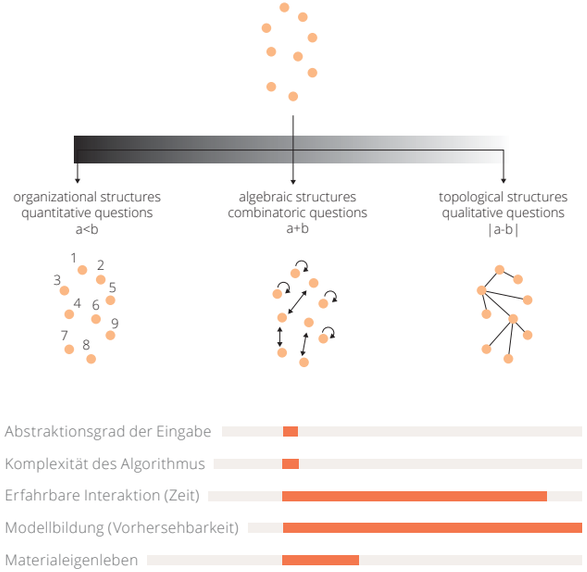
Virtueller Speicher der gebauten Struktur: man kann über die grafische Oberfläche in Rhino die Objekte verschieben oder löschen, und der Speicher wird dahingehend aktualisiert.

http://vimeo.com/37551029
Optical Tracking - Stacking Building Blocks

Beispiel:

Die Lage eines Holzwürfels wird fehlerhaft getrackt, er wird dadurch nicht am Mittelpunkt aufgehoben, falsch abgelegt und fällt runter. Dieser Baustein wird aus dem virtuellen Speicher entfernt und der nächste Baustein wird an dieselbe Stelle gelegt.

Es gibt viele Möglichkeiten nach welchen Ordnungsprinzipien strukturlose Ansammlungen geordnet werden können, die Bauteile linear anzuordnen ist der erste Versuch.



Mathematische Darstelllung:

$$f_u(s_n) = m \cdot L(x, y, \alpha)$$

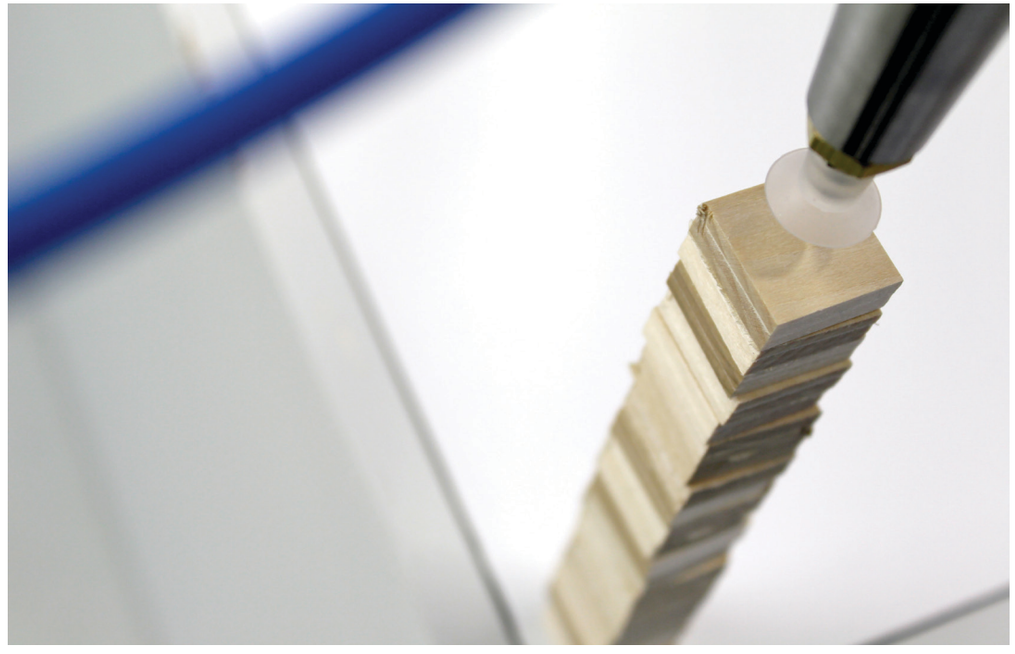
m ... Anzahl, L ...Position und Orientierung des Bausteins

mit $x, y, \alpha \in \mathbb{R}$, $m \in \mathbb{N}$

$s_0 = 0$ Zustand 0, Holzbausteine zufällig auf Platte verteilt

$s_{n+1} = f(f_u(s_n))$ s ... Zustand, n ... Iterationsschritte

vgl. Kotnik, Toni. (2011) Structuralism Reloaded. Rule-Based Design in Architecture and Urbanism. S.327



6. 9 Clustering 2D

Werkzeug: Vakuumheber

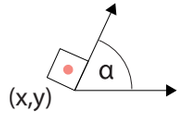
Externes Sensorsystem: Kamera

Abbildung: Kamerakoordinaten werden auf die eingemessene Basis des Roboters abgebildet

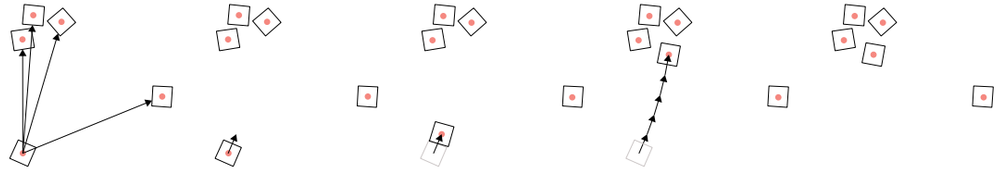
Material: Holzwürfel 25x25x8mm

Ablauf:

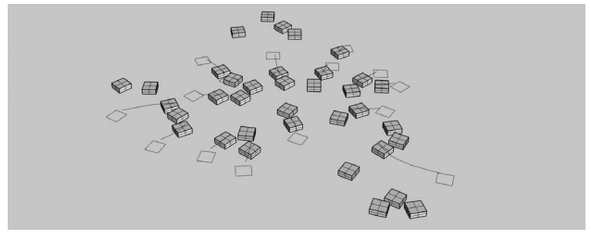
Auf der Platte, die von unten getrackt wird, werden Bausteine zufällig angeordnet und über die Kamera werden Anzahl, Position und Orientierung der Bausteine ermittelt. Die Bausteine werden vom Roboter, abhängig von den Berechnungen im Algorithmus, aufgenommen und auf der Platte versetzt.



Algorithmus:



Der Algorithmus basiert auf einem Partikelsystem. Die Bausteine besitzen eine Anziehungskraft, wenn mehrere Bausteine näher beieinander liegen, geht eine größere Anziehungskraft von ihnen aus, das bedeutet, dass Bereiche, wo es bereits Ansammlungen von Bausteinen gibt, verdichtet werden, und einzelne Bausteine von diesen angezogen werden.



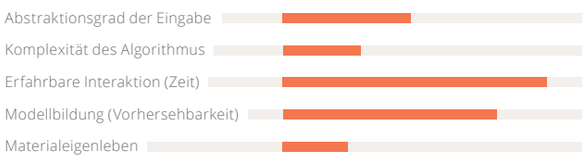
$$\vec{f} = \sum_{i=0,\ i \neq k}^{n} \frac{P_k - P_i}{|P_k - P_i|} \cdot \frac{c}{|P_k - P_i|^2}$$

$\vec{f}$ ... Kraftvektor für die Bewegung
P_k ... Mittelpunkt vom Baustein, der bewegt wird
P_i ... Mittelpunkt vom anderen Baustein
c ... Konstante
$|P_k - P_i|$ ... Abstand zwischen den beiden Bausteinen

Berechnung der Positionen der Bausteine in Rhino

Ein Iterationsschritt ist auf das Verschieben eines einzelnen Bausteines beschränkt, dazwischen werden die Kamerabilder immer neu ausgewertet, es gibt keinen virtuellen Speicher, da das Clustern nur in 2D stattfindet.

Es können, während das Programm läuft, immer Bausteine dazugelegt, weggenommen oder verschoben werden.



Mathematische Darstelllung:

$$f_u(s_n) = m \cdot L(x, y, \alpha)$$

m ... Anzahl, L ...Position und Orientierung des Bausteins

mit $x, y, \alpha \in \mathbb{R}$, $m \in \mathbb{N}$

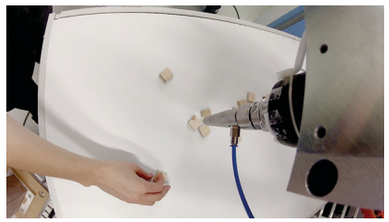
$s_0 = 0$ Zustand 0, Holzbausteine zufällig auf Platte verteilt

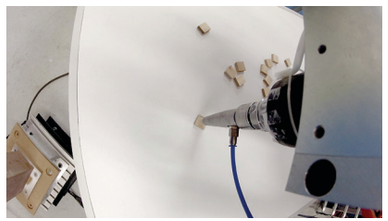
$s_{n+1} = f(f_u(s_n))$, s ... Zustand, n ... Iterationsschritte

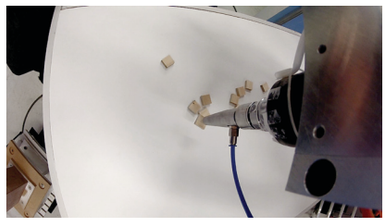
Beobachtungen:

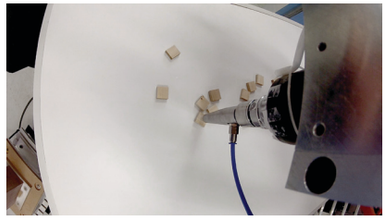
Der Zustand nach n Iterationsschritten ist immer davon abhängig wie die Bausteine beim Ausgangszustand s_0 hingelegt worden sind. Ihre Position und Rotation und die Entfernung zu den anderen ist entscheidend dafür, wie die Struktur nach mehreren Iterationsschritten aussieht und an welchen Stellen Anhäufungen passieren.

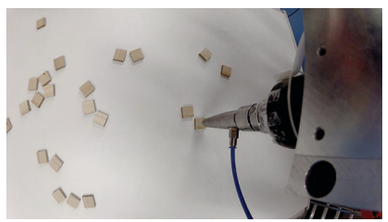
Man kann als Benutzer, ohne Kenntnis der Aufgabe, relativ schnell erfahren um was es im Algorithmus geht und kann darüber entscheiden wo es zu mehr Anhäufungen kommen soll. Legt man beispielsweise einen Baustein in die Nähe eines Baustein-Clusters, kann man davon ausgehen, dass der Roboter ihn abholt und zu diesem Baustein-Cluster dazulegt.











Versetzen der Holzbausteine mit dem Roboter

http://vimeo.com/39112794

6. 10 Clustering 3D

Werkzeug: Vakuumheber

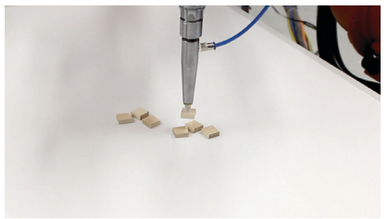
Externes Sensorsystem: Kamera

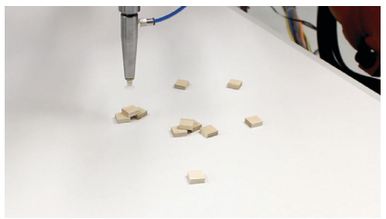
Abbildung: Kamerakoordinaten werden auf die eingemessene Basis des Roboters abgebildet

Material: Holzwürfel 25x25x8mm

Ablauf:

Auf der Platte, die von unten getrackt wird, werden Bausteine zufällig angeordnet und über die Kamera werden Anzahl, Position und Orientierung der Bausteine ermittelt. Die Bausteine werden vom Roboter, abhängig von den Berechnungen im Algorithmus, aufgenommen und auf der Platte versetzt, oder auf anderen Bausteinen abgelegt.





Roboter beim Clustern der Bausteine in 3D

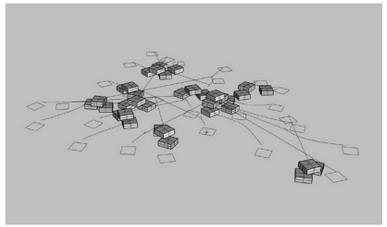
Da das Clustern der Bausteine nun um die dritte Dimension erweitert wird, werden die Bausteine, die auf anderen abgelegt werden, im Programm mitgespeichert.

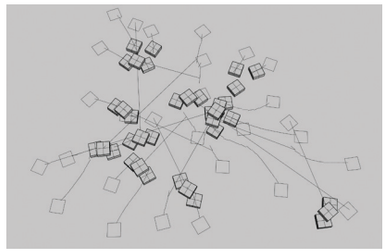
Algorithmus:

Der Algorithmus verläuft zunächst gleich wie beim Clustering 2D, die Bausteine häufen sich zu Ansammlungen zusammen. Wenn sie sich nicht mehr weiter zueinander bewegen können, werden sie auf zwei Bausteine draufgesetzt.

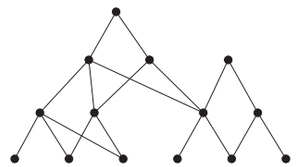
Hier wird ein virtueller Speicher benötigt, da von der Kamera Bausteine nicht mehr als solche erkannt werden, wenn sie übereinander liegen. Neue Bausteine können in der ersten Ebene ohne weiteres dazugelegt, weggenommen oder verschoben werden, wenn man aber einen Baustein in der zweiten oder dritten Ebene dazulegen oder wegnehmen will, muss man den virtuellen Speicher entsprechend den Veränderungen anpassen.

Die Bausteine werden im virtuellen Speicher in einer umgedrehten Baumstruktur gespeichert. Bausteine in der ersten Ebene sind Kinder und Bausteine in höheren Ebenen werden Eltern dieser Kinder. (Umgedreht deshalb, weil normalerweise zuerst die Eltern erzeugt werden, denen dann die Kinder zugefügt werden.)

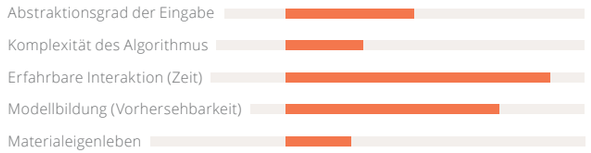




Virtueller Speicher der gebauten Struktur in Rhino mit Verlauf der Bewegung



Baumstruktur



http://vimeo.com/39119993
Roboter beim Stapeln

Mathematische Darstelllung:

$$f_u(s_n) = m \cdot L(x, y, \alpha)$$

m ... Anzahl, L ...Position und Orientierung des Bausteins

mit $x, y, \alpha \in \mathbb{R}$, $m \in \mathbb{N}$

$s_0 = 0$ Zustand 0, Holzbausteine zufällig auf Platte verteilt

$$s_{n+1} = f(f_u(s_n))$$

s ... Zustand, n ... Iterationsschritte

http://vimeo.com/40517026
Simulation in Rhino

Beobachtungen:

Zusätzlich zu den Abstandsbedingungen im Clustering 2D kommen hier die Gleichgewichtsbedingungungen dazu. Da die Bausteine nicht miteinander verbunden werden, müssen sie im Gleichgewicht auf den unteren aufliegen.

Es ist keine ideale Lösung, dass die Bausteine aus den höheren Ebenen virtuell mitgespeichert werden müssen, das passiert mangels an adäquaten Sensorsystemen.

6. 11 Fadenbilder

Werkzeug: Vorrichtung zum Faden abwickeln

Externes Sensorsystem: Kamera

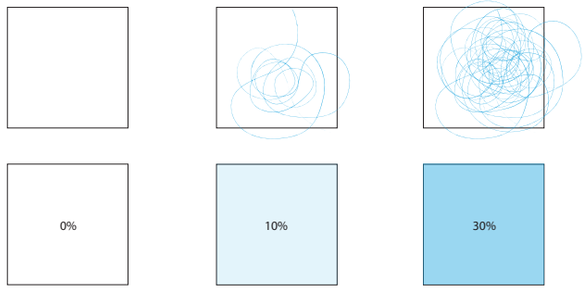
Abbildung: Kamerakoordinaten werden auf die eingemessene Basis des Roboters abgebildet

Material: Polyesterfaden

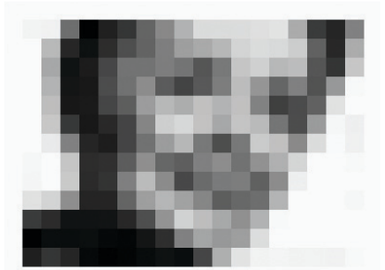
Ablauf:

Ein Faden wird über der Platte an vorgegebenen Positionen über eine Vorrichtung abgewickelt und fällt auf die Oberfläche. An dieser Stelle wird von unten über die Kamera der Graustufenwert getrackt, der sich mit der Menge an Faden an dieser Stelle verändert, und wenn dieser dem gewünschten Sollwert entspricht, verfährt der Roboter zur nächsten Position oder das Abwickeln wird gestoppt.

Den Graustufenwerten liegt ein Pixelbild mit der Auflösung 20 x 15px zu Grunde, ein Pixel enstpricht auf der Platte einem Bereich von 50x50mm.

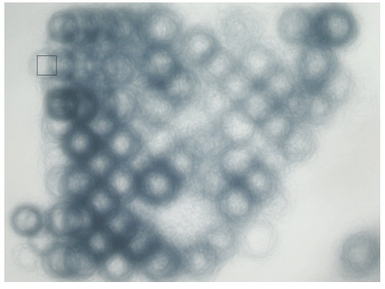


Auswertung der Kamera von definierten Bereichen bezüglich der Farbintensität bzw. Graustufen



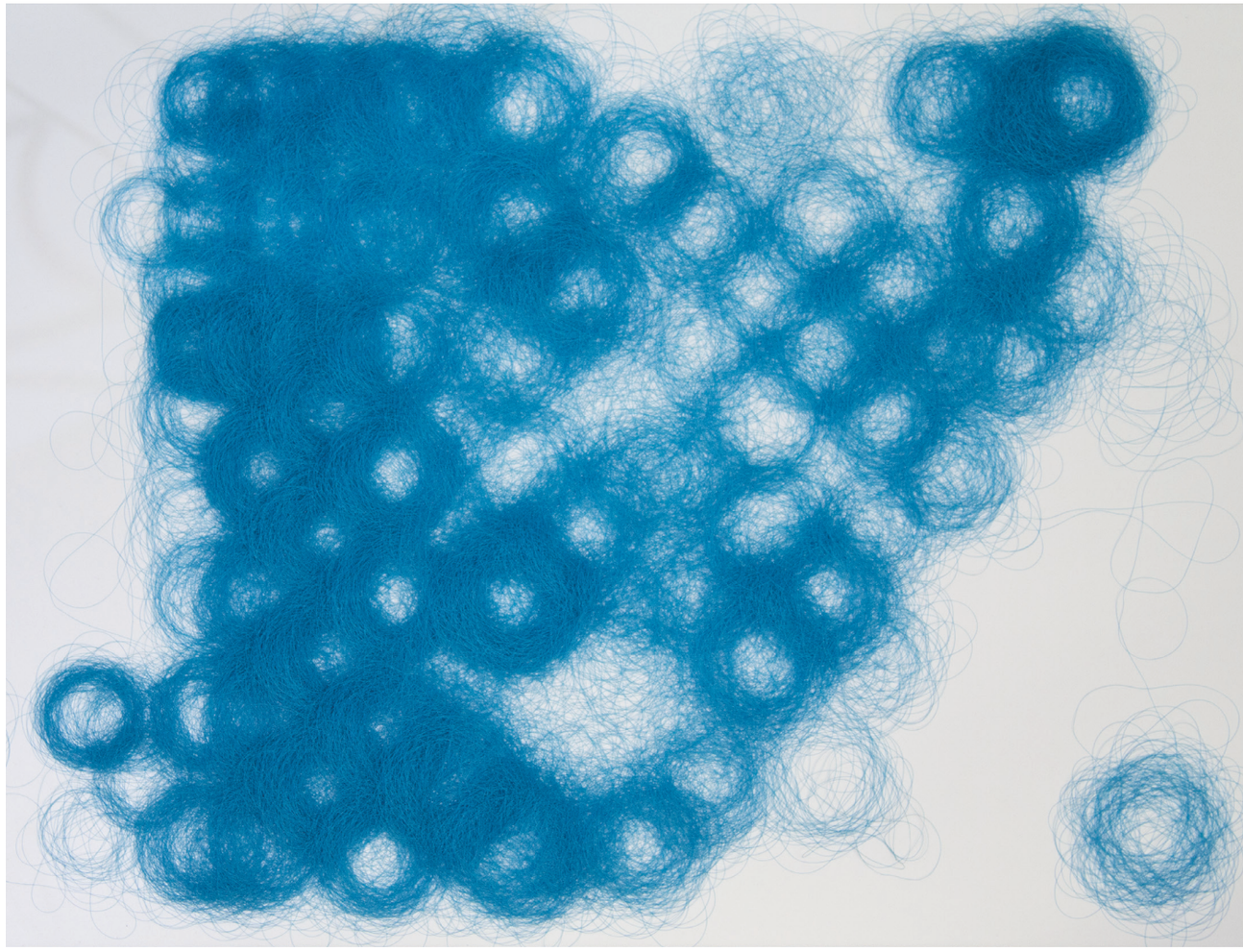
Graustufenbild 20 x 15px

Auswertung durch die Kamera 1600 x 1200px

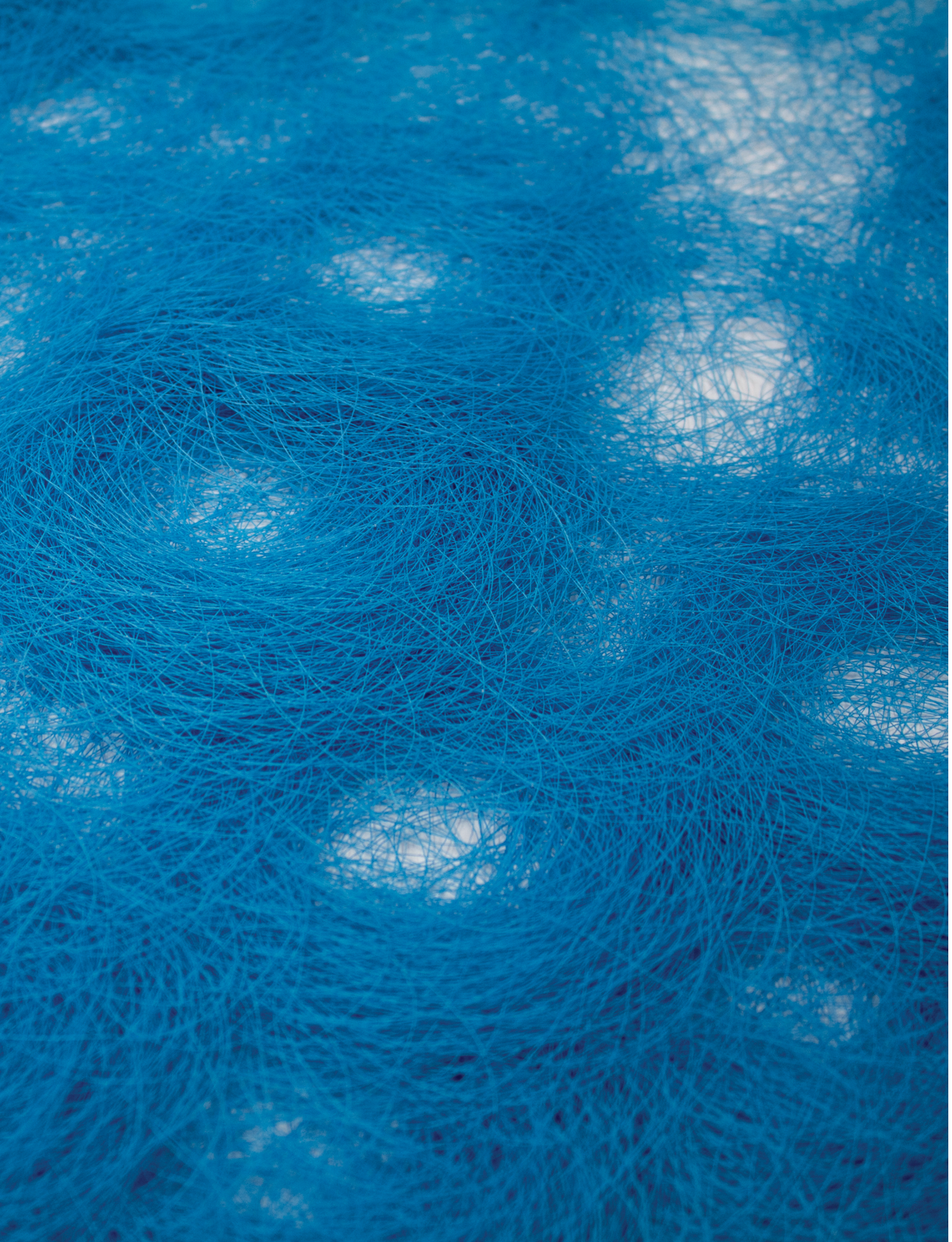


Kamerabild von unten

http://vimeo.com/39966095



Vollendetes Bild 1090 x 750mm

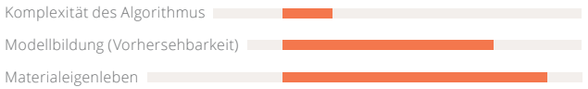


Beobachtungen:

Bei diesem Experiment geht es nicht um ein interaktives Setup für den/die Gestalter/in, sondern es handelt sich um ein geschlossenens System zwischen Kamera und Roboter.

Wie in Kapitel 5.12 in Umsetzung behandelt, geht es hier lediglich um den Versuch, durch Überprüfung lokaler Gegebenheiten (in diesem Fall durch die Kamera) und dem Reagieren auf den jeweiligen Zustand, zu einem Gesamtergebnis zu kommen.

Das bedeutet, dass man vorher wenig wissen muss, man muss das genaue Verhalten des auftreffenden Fadens auf der Platte nicht voraussagen können, noch die genaue Menge wissen, die in einer bestimmten Zeit von der Spule abgewickelt wird.



Mathematische Darstelllung:

$s_0 = 0$, Zustand 0

$s_{n+1} = f(s_n)$, s ... Zustand, n ... Iterationsschritte

6. 12 Malen nach Faden

Werkzeug: Vorrichtung zum Faden abwickeln

Externes Sensorsystem: Kamera

Abbildung: Kamerakoordinaten werden auf die eingemessene Basis des Roboters abgebildet

Material: Polyesterfaden

Ablauf:

Bei diesem Faden-Experiment gibt es keine Zielvorgabe, wie ein Graustufenbild, sondern vom Algorithmus werden nur gewisse lokale Anweisungen vorgegeben, beispielsweise dunkle Bereiche bis zu einem gewissen Wert zu verstärken. Hier wird der/die Gestalter/in wieder in den Prozess integriert. Man kann diesen moderieren, aber wenn man nicht eingreift, läuft der Prozess auch alleine und es kommt genauso zu einem Ergebnis.

Zu Beginn fährt der Roboter an eine zufällige Position über der Platte und lässt den Faden fallen. Das Kamerabild wird in kurzen Zeitabständen in einem gewissen Radius um die Position des Roboters ausgewertet.

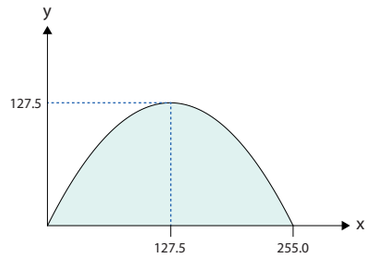
Nun könnte der Algorithmus so gestaltet sein, dass entweder an der Position des hellsten Pixels Faden fallen gelassen wird, oder an der Position des dunkelsten. Je nachdem würde nach mehreren Iterationen der ganze Tisch mit Faden belegt werden oder der Faden sich in einem Bereich häufen. Stattdessen wird in unserem Programm jeder Pixelwert, bzw. Grauwert (von 0 bis 255) mit einer parabolischen Funktion auf einen Bereich zwischen 0 bis 127.5 abgebildet.

Durch diese Funktion werden sehr kleine (dunkelgraue) oder sehr hohe (hellgraue) Werte auf den gleichen Zahlenwert abgebildet und die höchsten Werte werden in der Mitte (mittelgrau) erzielt. Damit erreicht man, dass der Faden sich zwar an Stellen anhäuft, aber nur bis zu einem gewissen Wert, wonach die Position des Roboters wieder geändert wird. In Abhängigkeit dieser Parameter entstehen unterschiedliche Muster.

http://vimeo.com/39909361
Roter Faden

http://vimeo.com/39971463
Grauer, blauer und grüner Faden

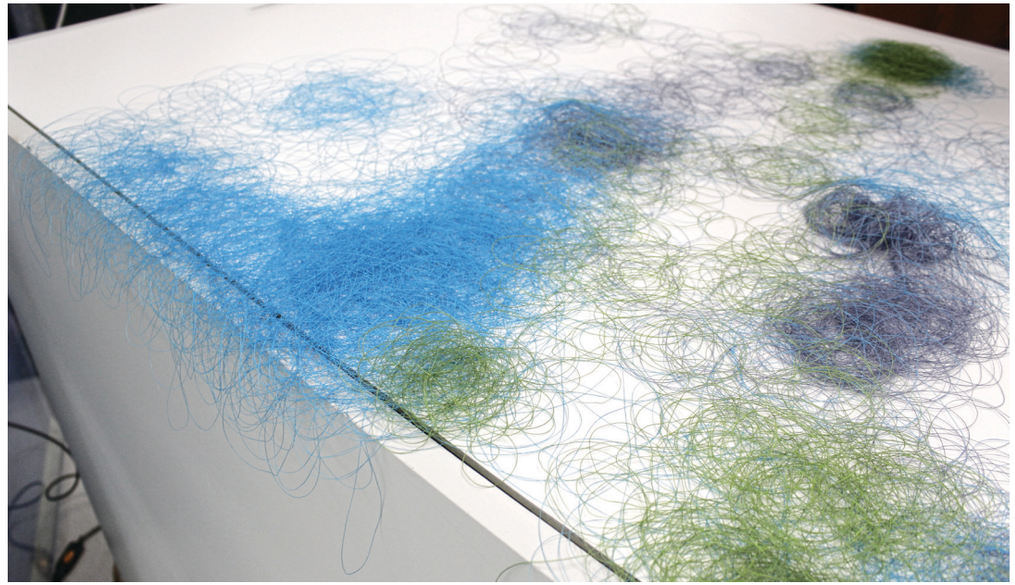
http://vimeo.com/39986901
Grauer Faden



$$y = \frac{r}{255} \cdot x \cdot (255 - x)$$

mit $r = 2$; $x \in \mathbb{N}_0$, $y \in \mathbb{R}_+$, $0 \leq x \leq 255$

Parabolische Funktion



Abstraktionsgrad der Eingabe
Komplexität des Algorithmus
Erfahrbare Interaktion (Zeit)
Modellbildung (Vorhersehbarkeit)
Materialeigenleben

Mathematische Darstelllung:

$$f_u(s_n) = L(x, y)$$

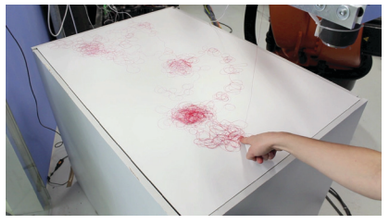
L ...Position an der der Faden zu liegen kommen soll

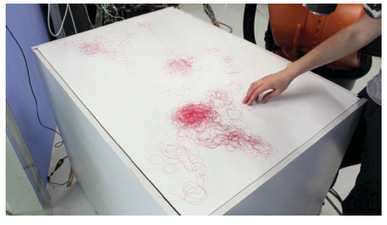
mit $x, y \in \mathbb{R}$

$s_0 = 0$ Zustand 0

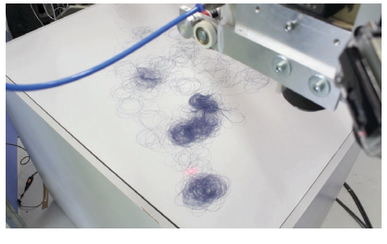
$$s_{n+1} = f(f_u(s_n))$$

s ... Zustand, n ... Iterationsschritte





Manipulation mit der Hand



Manipulation mit dem Laserpointer

6. 13 Stick Piling

Werkzeug: Vakuumheber

Externes Sensorsystem: Kamera

Abbildung: Kamerakoordinaten werden auf die eingemessene Basis des Roboters abgebildet

Material: Holzstäbe 5x5mm Querschnitt, mit variabler Länge von 100 bis 250mm

Ablauf: 

Auf der Platte, die von unten getrackt wird, werden in einem Bereich Stäbe zufällig angeordnet, die die erste Ebene einer Stuktur definieren die gebaut werden soll und in einem anderen Bereich werden Stäbe platziert, die für den Bau der Struktur verwendet werden. Über die Kamera werden Anzahl, Position und Orientierung der Stäbe ermittelt.

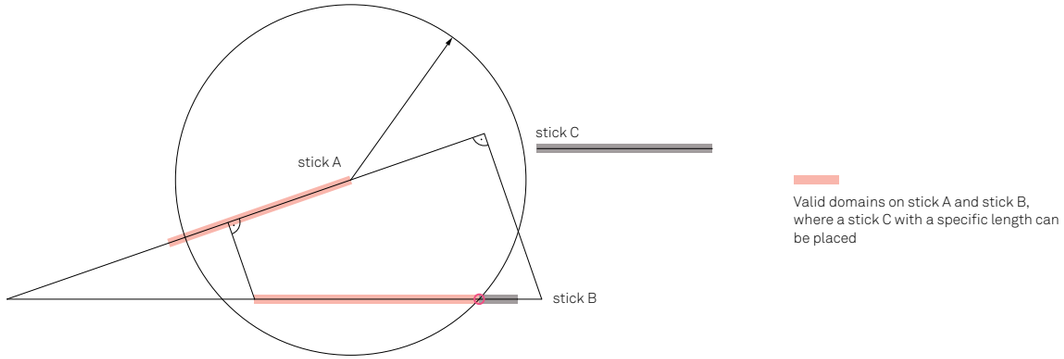
Algorithmus:

Das Ziel des Programmes ist, Stäbe unterschiedlicher Länge aufeinander zu stapeln, mit je zwei oder mehr Auflagern, so dass alle im Gleichgewicht bleiben.

Gegeben sind ein Stab A und ein Stab B in gleicher Ebene, innerhalb der Struktur, mit je zwei Auflagern und für jedes Auflager ein Maximalwert, der bestimmt, wieviel Kraft noch in diesen Auflagern wirken darf, ohne dass die Struktur aus dem Gleichgewicht gerät.

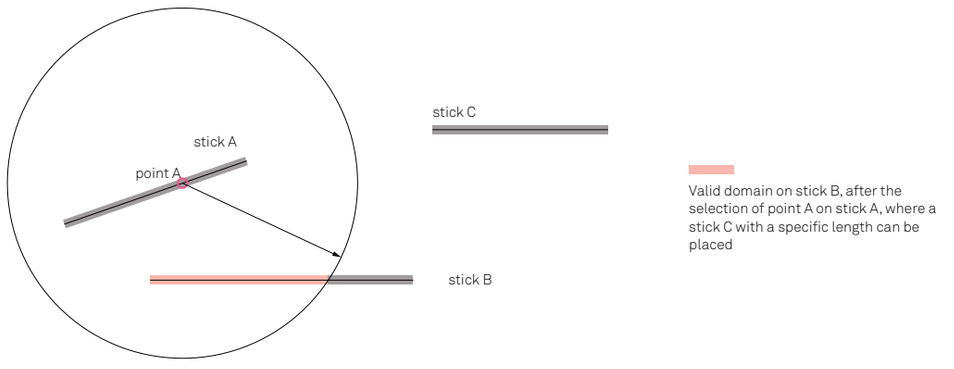
Dieser Maximalwert wird bei jedem hinzugelegten Stab neu berechnet und ergibt sich aus allen Stäben, die unterhalb desselben liegen. Außerdem müssen beim Hinzufügen eines neuen Stabes alle Maxima und Auflager für die Stäbe darunter aktualisiert werden.

Soll nun ein Stab C mit gegebener Länge und Gewicht auf die Stäbe A und B gelegt werden, so wird zuerst die mögliche geometrische Lage bestimmt:



Valid domains on stick A and stick B, where a stick C with a specific length can be placed

Innerhalb des geometrisch möglichen Bereiches auf Stab A wird nun ein Punkt A bestimmt, der wiederum den Bereich auf dem Stab B einschränkt.



Valid domain on stick B, after the selection of point A on stick A, where a stick C with a specific length can be placed

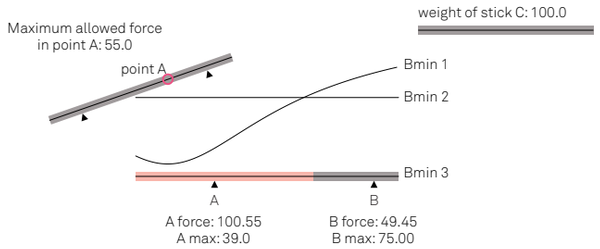
Nun werden die minimalen Auflagerkräfte auf Stab B berechnet, also für jeden Punkt auf Stab B, die minimale Auflagerkraft bestimmt, die in diesem Punkt wirken muss.

Das erste Minimum ergibt sich geometrisch. Der Mittelpunkt des draufzulegenden Stabes C muss innerhalb der Auflagerpunkte sein, damit er nicht aus dem Gleichgewicht gerät.

Das zweite Minimum ergibt sich dadurch, dass auf dem Stab A

schon ein Punkt A ausgewählt wurde und dieser aufgrund der Auflager- und Maximaberechnungen des Stabes A eine maximal mögliche Auflagerkraft in diesem Punkt besitzt. Das Minimum ist das Gewicht vom Stab C abzüglich des Maximums im Punkt A, sofern es nicht unter Null fällt.

Die dritte Minimumbedingung ist, dass B nicht kleiner als Null sein darf.



Bmin 1:
The midpoint of stick C must be within its supports, according to that each point on stick B must carry a minimum weight

Bmin 2:
According to the maximum allowed force in point A, there must be a minimum weight carried on stick B (100.0 - 55.0 = 45.0)

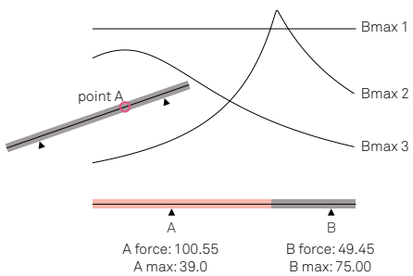
Bmin 3:
Must ge greater than zero

Nun werden die maximalen Auflagerkräfte auf Stab B berechnet, also für jeden Punkt auf Stab B, die maximale Auflagerkraft bestimmt, die in diesem Punkt wirken kann.

Die erste Maximumbedingung ist, dass B maximal das ganze Gewicht von Stab C tragen kann.

Das zweite Maximum berechnet sich aufgrund der Auflager und Maxima, die für die Auflager des Stabes B gelten.

Die dritte Maximumbedingung ist eigentlich die gespiegelte erste Minimumbedingung (geometrisch), da diese ja auch auf dem Stab A gelten muss, und ergibt sich durch die Subtraktion vom Gewicht des Stabes C mit dem Minimumwert.



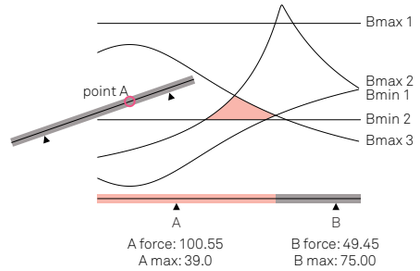
Bmax 1:
Can be maximal weight of stick C: 100.0

Bmax 2:
According to the maximum allowed forces in the supports A and B on stick B, for every point on stick B there exists a different maximum force

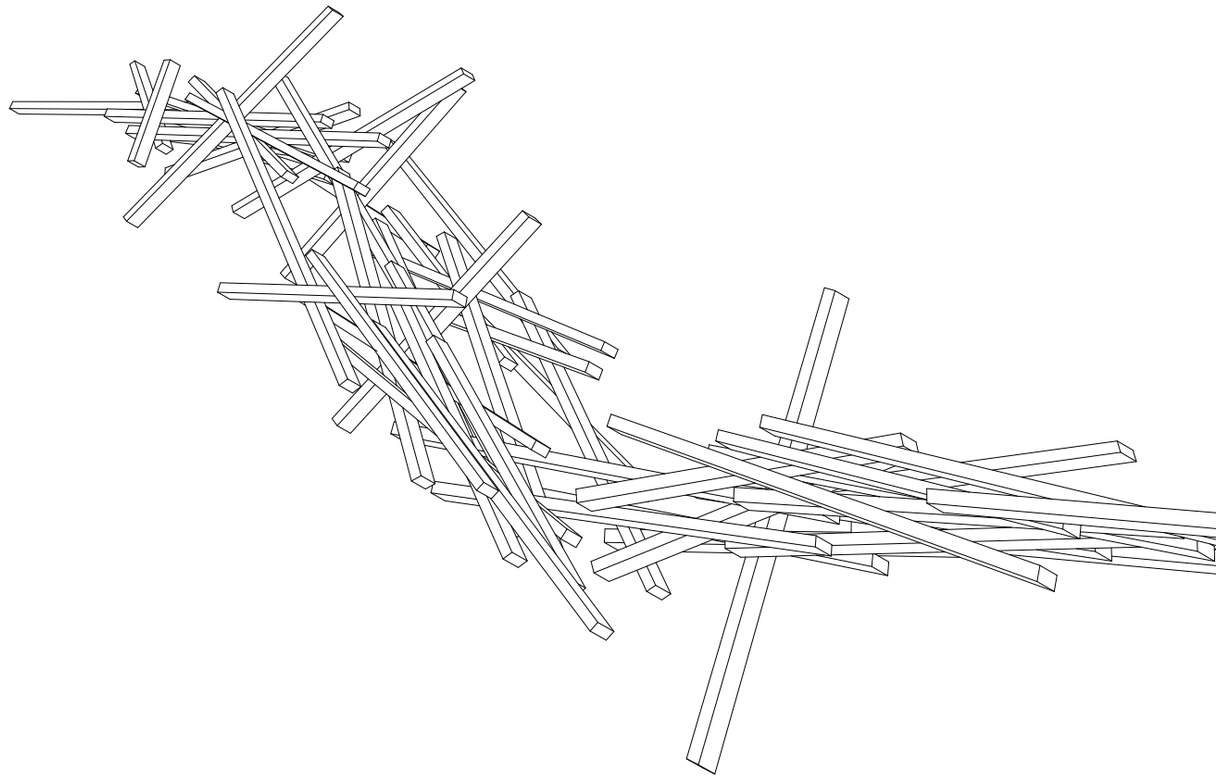
Bmax 3:
According to the length of stick C and the distance of the supports of C, there must be also a maximum weight carried on one side (The curve is the mirrored Bmin 2, Bmax 3 = weightC - Bmin 2)

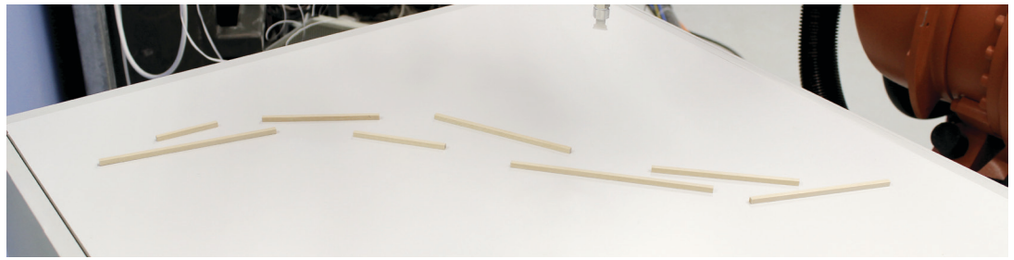
Als Lösungsmenge wird der Bereich bestimmt, in dem das minimale Maximum größer ist als das maximale Minimum.

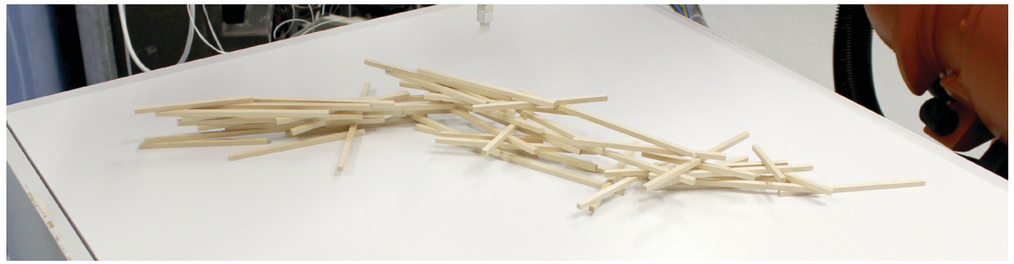
In diesem Bereich kann nun jeder beliebige Punkt P(x,B) gewählt werden, wobei x die Position des Auflagers am Stab B von Stab C ist, mit der Auflagerkraft B.

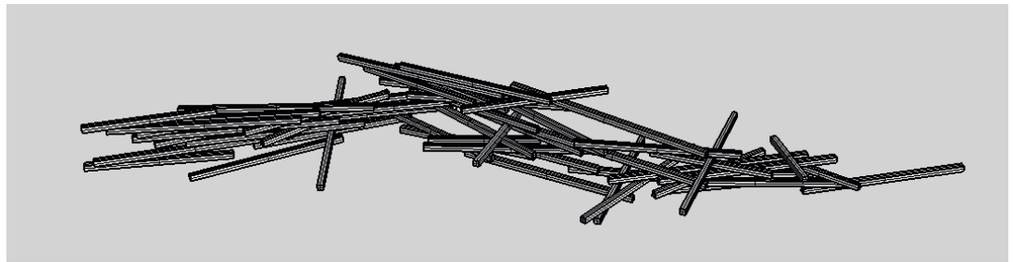


Valid domain where any Point P(x,B) leads to a solution where x is the position of the support of stick C on stick B and B is the resulting force in x







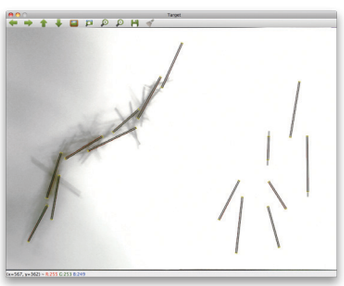


Ausgangslage, Endzustand und das parallel entstehende 3D-Modell. Mit den Stäben am Boden gibt man die Richtung für das Ergebnis vor, das weitere Aufbauen der Struktur wird über den Algorithmus bestimmt, und hängt von den Längen der gefundenen Stäben ab.

Beobachtungen:

In dieser Studie wird aufgezeigt, dass die einfache Aufgabenstellung, Stäbe im Gleichgewicht aufeinanderzustapeln, zu komplexen Berechnungen führen kann, die effizient mit dem Computer abgehandelt werden können. Würde ein Mensch dieselbe Aufgabe erledigen versuchen, wäre seine einzige Möglichkeit das „Trial and Error" - Prinzip, denn die möglichen Bereiche in höheren Ebenen, in denen Lösungen für das Auflegen weiterer Stäbe gefunden werden können, ist schrinkend klein.

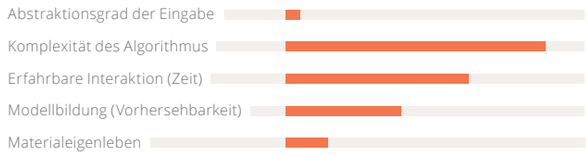
Die Gestalter/innen können durch die Definition der Lage und Rotation der Stäbe in der ersten Ebene Einfluss auf die resultierende Struktur nehmen. Außerdem können sie dem Roboter „helfen" wenn sie beispielsweise sehen, dass ein kurzer Stab für den Weiterbau benötigt wird.



Kamerabild mit mit den getrackten Stäben

Es können Stäbe auch wieder entfernt werden, sollte deren Position nicht den gestalterischen Kriterien entsprechen oder sollten diese fehlerhaft platziert worden sein, diese müssen dann gleichzeitig auch aus dem virtuellen Speicher gelöscht werden.

Die Stäbe im Speicher werden wie beim Clustering 3D in einer Baumstruktur gespeichert.



http://vimeo.com/40393402
Simulation in Rhino

http://vimeo.com/41288647
Roboter beim Stapeln

http://vimeo.com/41309596
Löschen eines Stabes aus dem Speicher

Mathematische Darstelllung:

$$f_u(s_n) = m \cdot L(x, y, \alpha)$$

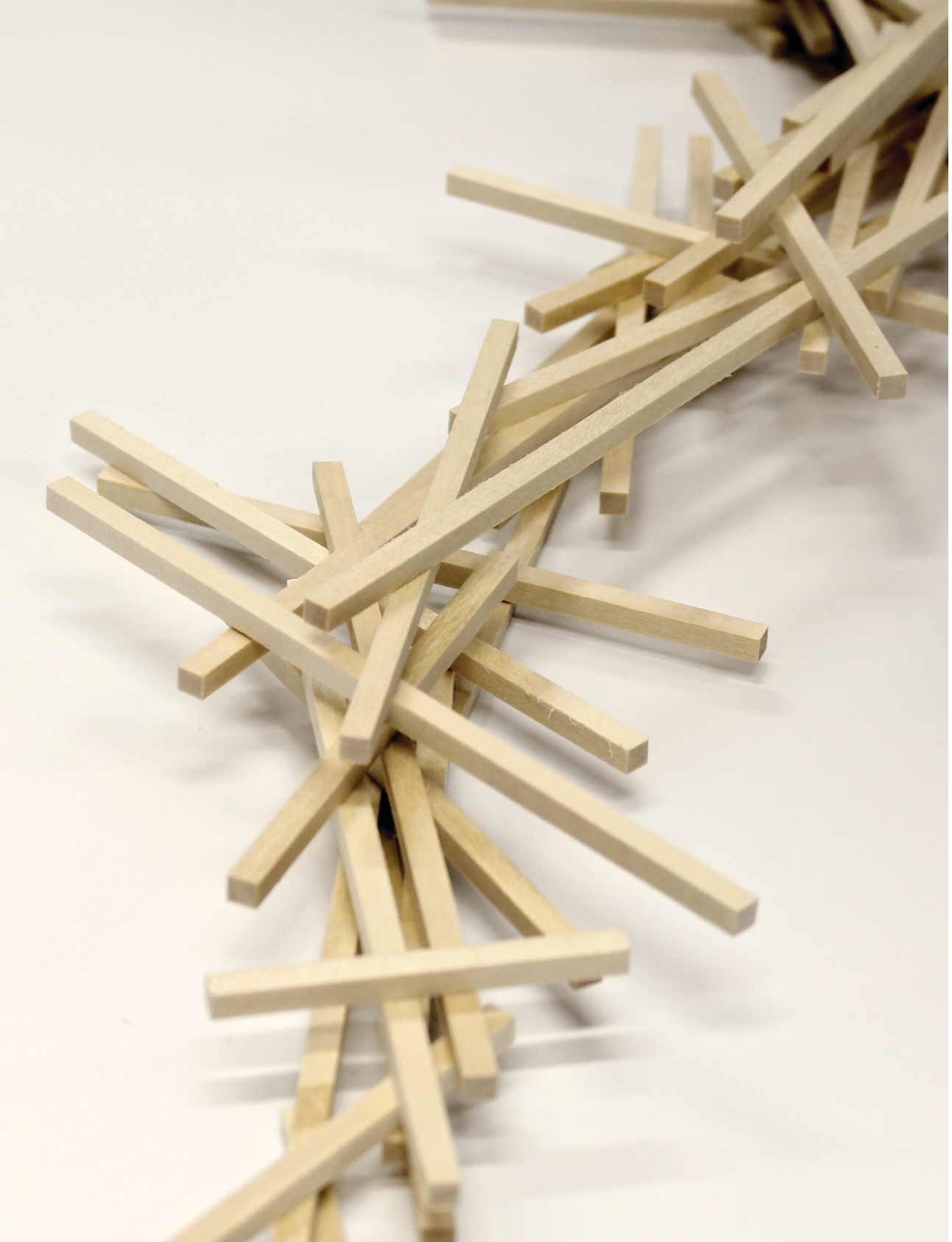
m ... Anzahl, L ...Position und Orientierung des Stabes

mit $x, y, \alpha \in \mathbb{R}$, $m \in \mathbb{N}$

s_0 ... Anfänglich platzierte Holzstäbe

$s_{n+1} = f(f_u(s_n))$, s ... Zustand, n ... Iterationsschritten

Seitenansicht der aufeinander liegenden Stäbe



6. 14 Nest

Werkzeug: Vakuumheber

Externes Sensorsystem: Kamera

Abbildung: Kamerakoordinaten werden auf die eingemessene Basis des Roboters abgebildet

Material: Holzstäbe 5x5mm Querschnitt, mit variabler Länge von 100mm bis 250mm

Ablauf:

Auf der Platte, die von unten getrackt wird, werden in einem Bereich Stäbe angeordnet, die die erste Ebene einer Stuktur definieren die gebaut werden soll und in einem anderen Bereich werden Stäbe platziert, die für den Bau der Struktur verwendet werden. Über die Kamera werden Anzahl, Position und Orientierung der Stäbe ermittelt.

http://vimeo.com/41299115
Roboter beim Stapeln des Nests

Algorithmus:

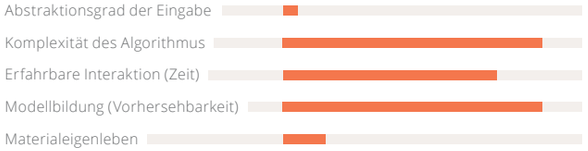
Auf die schon platzierten Stäbe werden Ebene für Ebene neue Stäbe hinzugelegt, die sich an der Lage der unteren orientieren. Im Gegensatz zum Stick Piling berechnet sich also die Lage des nächsten Stabes unter anderem auch aus der Richtung der anderen Stäbe in der Struktur. Die Stäbe im virtuellen Speicher werden in einer Baumstruktur gespeichert.

Beobachtungen:

Im Gegensatz zur vorherigen Fallstudie sind die Ergebnisse absehbarer, der/die Benutzer/in erkennt schneller, wie sich die Struktur entwickeln wird.

Durch die Definition der Lage und Rotation der Stäbe in der ersten Ebene wird die Struktur definiert. Wiederum kann man das System beim Bau unterstützen, wenn man sieht, dass ein kürzerer oder längerer Stab ein Loch füllen könnte.

Der virtuelle Speicher wird korrigiert, wenn ein Stab hinunterfällt oder aus gestalterischen Kriterien aus der Struktur entfernt wird.



Mathematische Darstelllung:

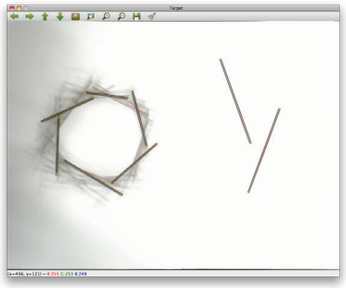
$$f_u(s_n) = m \cdot L(x, y, \alpha)$$

m ... Anzahl, L ...Position und Orientierung des Stabes

mit $x, y, \alpha \in \mathbb{R}$, $m \in \mathbb{N}$

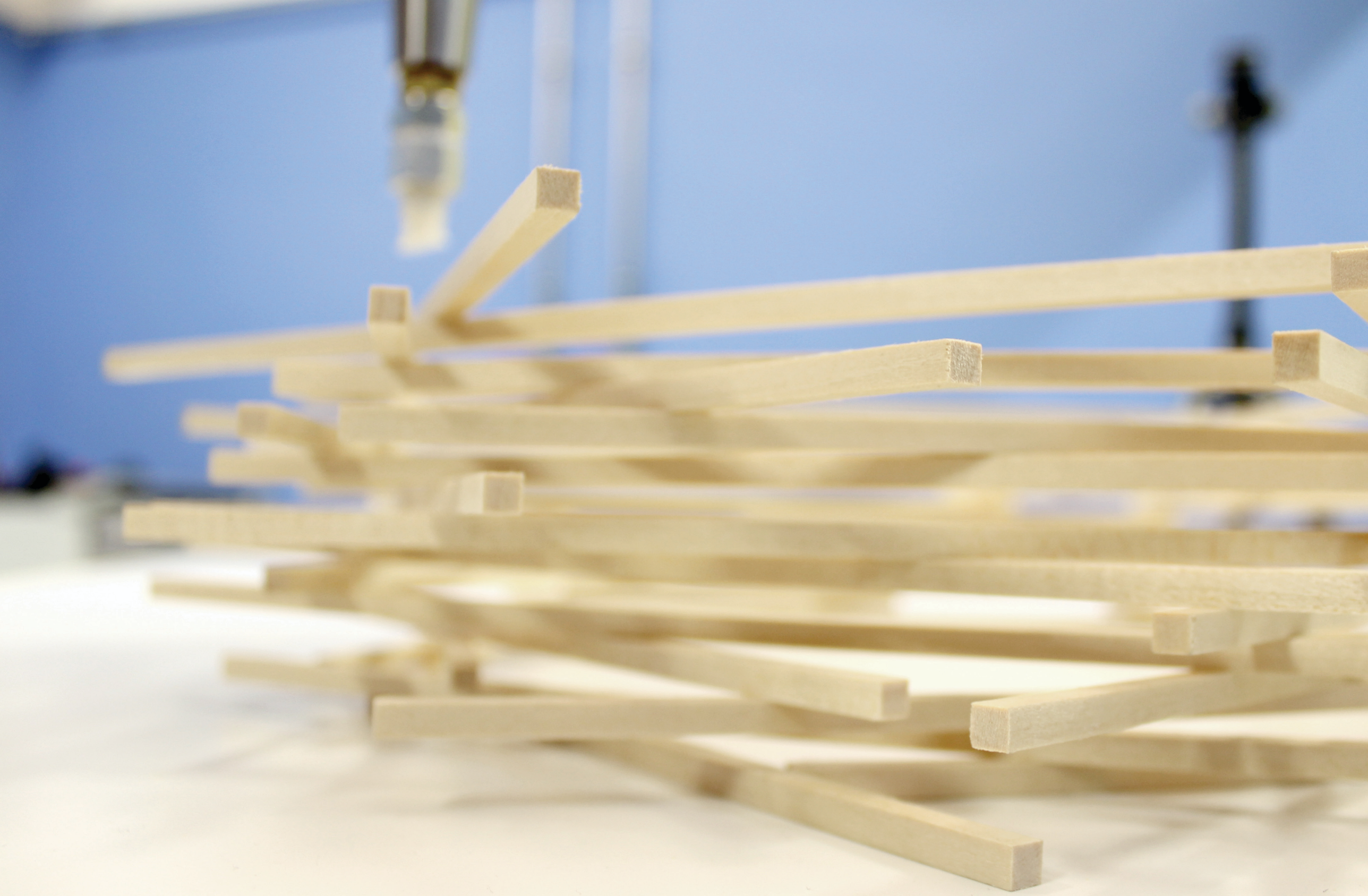
s_0 ... willkürlich platzierte Holzstäbe

$s_{n+1} = f(f_u(s_n))$, s ... Zustand, n ... Iterationsschritten



Kamerabild mit mit den eingezeichneten getrackten Stäben

Rendering der parallel zur physischen entstandenen virtuellen 3D Struktur.



7 Conclusio

7 Conclusio

7.1 Conclusio

Customizing

Mit der individuellen Anpassung des Roboters durch verschiedene Sensorsysteme und der im Rahmen der Arbeit entwickelten interaktiven Steuerung wird das Werkzeug aus seinem vorbestimmten Rahmen gehoben. Anstatt die Maschine als reine Anwender/innen zu benutzen, sind wir nun in der Lage, die Fähigkeiten der Maschine zu erweitern und sie dadurch als komplexes Gestaltungswerkzeug einsetzen zu können.

Komplexität

Durch die Studien konnte man zeigen, dass eine vollständige Erfassung eines Ist-Zustandes einer Struktur über verschiedene Sensoren in diesem Rahmen nicht unbedingt notwendig ist (z.B vollständiges Erfassen der drei Dimensionen eines dreidimensionalen Körpers). Es ist sinnvoller, nur die Parameter, die für ein spezielles Programm wichtig sind (z.B nur die Höhe, Farbe, Temperatur etc.), aus der gegebenen Situation zu extrahieren.

Robustheit

Die Verarbeitung realer Daten, gemessener Größen oder menschlichen Interaktionen zwingt zu einer Verwendung robuster und fehlertoleranter Verfahren. Das wiederholte Messen des Ist-Zustandes und iterative Verfahren unterstützen das Entwerfen robuster Algorithmen, wo sich kleine Fehler automatisch ausgleichen und sich nicht fatal auf das System auswirken.

Material und Immaterialität

Das parallele Speichern von Strukturen im CAD Programm, während sie vom Roboter gebaut werden bereicherte die Arbeit mit der Möglichkeit der Gegenüberstellung der physischen Struktur und des virtuellen Modells, und der Möglichkeit der Wiederverwendung oder Nachbearbeitung desselben.

Gleichzeitig erfährt man die Unterschiedlichkeit beider (in umgedrehter Weise wie es normalerweise geschieht, nämlich dass ein virtuelles Modell parallel zum physisch gebauten Objekt

entsteht), die gebaute physische Struktur kann von allen Seiten untersucht und mit allen Sinnen erfahren werden.

Während die Eigenschaften der verwendeten Materialien in den gebauten Objekten strukturgebend sind, unterliegen die Elemente neben ihrer materiellen Präsenz auch einer immateriellen digitalen Logik, die durch das Bauen mit dem Roboter sichtbar und erfahrbar gemacht wird.

Intuition

Das intuitive Eingreifen während des automatisierten Bauprozesses war erfolgreich und lieferte positive Ergebnisse, obwohl erst sehr einfache Interaktionsmöglichkeiten implementiert worden sind.

Interessanterweise eröffnen sich aus der intensiveren Auseinandersetzung mit der Maschine und ihrer Programmierung die Möglichkeiten zur intuitiven Gestaltung der Programme.

Interaktion

Damit der Prozess für interaktiv eingreifende Benutzer/innen spannend und überraschend bleibt, muss er so gestaltet sein, dass die Gestalt nicht vorab definiert ist, sondern man nur eine Richtung im Wachstumsprozess auf lokaler Ebene vorgibt.

Ausblicke

Durch die vorliegende Arbeit wurde eine Grundlage geschaffen, die vielfältigen Möglichkeiten aber noch lange nicht vollständig exploriert.

Der offene Aufbau des Systems erlaubt bzw. auch ermuntert zur Erweiterung durch verschiedene Clients und Sensorsysteme. Beispielsweise könnten ohne größeren Aufwand weitere Sensoren oder Aktuatoren hinzugefügt werden, um beispielsweise die Höhe von Bauteilen zu messen oder Bauteile miteinander zu verbinden.

Wir sehen beachtenswerte Möglichkeiten in den Verfahren, die auf Feedback beruhen, gerade in denen, die real gemessene Zustandsgrößen in die Feedbackschleife mit aufnehmen. Das Einbeziehen von gemessenen Größen aus der Realität kann Algorithmen etwa davon befreien, komplexes Verhalten von Materialien vollständig vorab zu simulieren, was eine Vereinfachung der Algorithmen nach sich zieht.

Verhalten von Materialien, mögliche Toleranzen etc. müssen nicht in vollem Umfang vorab berechnet oder simuliert werden, sondern Zustände werden in wiederholten Abständen (zwischen Iterationsschritten) überprüft.

Zum Beispiel wird nicht vorab eine benötigte Menge an Material für die Herstellung einer stabilen Form berechnet, sondern die Form wird wiederholt Belastungstests unterzogen, und Material wird solange aufgetragen, bis sie den Belastungstests standhält.

Vielleicht ist auf diesem Weg sogar denkbar, dass in der Zukunft auf der Baustelle ungeordnete, natürliche oder inhomogene Baustoffe automatisiert aneinandergefügt werden.

Anhang

Bibliographie

Alexander, Christopher: (1964) Notes on the Synthesis of Form. Cambridge, Massachusetts, London: Harvard University Press.

Bergson, Henri: (1993) Denken und schöpferisches Werden. Hamburg: eva-Taschenbuch. (Originalausgabe: Bergson, Henri. (1946) La pensée et le mouvant. Paris: Presses Universitaires de France)

Carpo, Mario/ Lemerla, Frédérique: (2008) Perspective, Projection & Design. Technologies of Architectural Representation. New York: Routledge.

Corser, Robert: (2010) Fabricating Architecture. Selected readings in digital design and manufacturing. New York: Princeton Architectural Press.

Foord, Michael J. und Muirhead, Christian: (2009) IronPython in Action. Greenwich: Manning Publications Co.

Geiser, Reto: (2008) Explorations in Architecture: Teaching, Design, Research. Basel. Boston. Berlin: Birkhäuser Verlag AG.

Gramazio, Fabio / Kohler, Matthias: (2008) Digital Materialilty in Architecture. Baden, Switzerland: Lars Müller Publishers.

Hertzberger, Hermann: (1995) Vom Bauen. Vorlesungen über Architektur. München: Aries Verlag.

Hovestadt, Ludger: (2010) Jenseits des Rasters – Architektur und Informationstechnologie. Basel. Boston. Berlin: Birkhäuser Verlag AG.

Kilian, Axel: (2006) Design Exploration through Bidirectional Modeling of Constraints. PhD Thesis at Massachusetts Institute of Technology.

KUKA.Ethernet KRL XML 1.1. Für KUKA System Software (KSS) 5.x. KUKA Roboter GmbH (2006)

Kraft, Sabine / Kuhnert, Nikolaus / Uhlig, Günther (Hrsg.): (2008) ARCH+ Ausgabe Nr. 189, Zeitschrift für Architektur und Städtebau, Aachen: ARCH+ Verlag GmbH.

Lutz, Mark: (2009) Learning Python, O'Reilly Media; Auflage: 4

Pfeifer, Rolf / Bongard, Josh: (2009) How the Body Shapes the Way We Think. A New View of Intelligence. London, Cambridge: The MIT Press.

Piper, Ian: (2009) Learn Xcode Tools for Mac OS X and iPhone Development. Apress: Auflage: 1.

Rider, Alistair: (2011) Carl Andre. Things In Their Elements. London: Phaidon Press.

Senett, Richard (2008): Handwerk. Berlin: Berlin Verlag.

Shannon, Claude E: (1948). A Mathematical Theory of Communication. Bell System Technical Journal Vol. 27.

Tibbits, Skylar J.E.: (2007) Logic Matter. Digital logic as heuristics for physical self-guided-assembly. Master-Thesis. Massachusetts Institute of Technology / Philadelphia University.

Valena, Tomás / Avermaete, Tom / Vrachliotis, Georg (Hrsg.): (2009) Structuralism Reloaded?: Rule-Based Designs in Architecture and Urbanism. Edition Axel Menges.

Bildnachweis

Bilder externer Quellen wurden direkt auf der sich befindenden Seite ausgewiesen, alle andere Fotos und Grafiken wurden von den Autorinnen selbst erstellt.

Aus Gründen der besseren Lesbarkeit wurden teilweise auf die gleichzeitige Verwendung männlicher und weiblicher Sprachformen verzichtet. Sämtliche Personenbezeichnungen gelten in diesen Fällen gleichwohl für beiderlei Geschlecht.

Danksagung

Für die Unterstützung bei der Diplomarbeit wollen wir uns bei folgenden Personen herzlich bedanken:

Unseren Betreuern Urs Hirschberg und Christian Kern für ihre hilfreichen Anregungen und ihre konstruktive Kritik bei der Erstellung dieser Arbeit.

Florian Rist für seine außerordentliche Mithilfe, Anregungen, Gedanken und Vorschläge. Ohne ihn wäre diese Arbeit nicht möglich gewesen.

Dem Institut für Kunst und Gestaltung in Wien für die Benutzung der Räumlichkeiten, der Maschinen und der Bereitstellung eines Arbeitsplatzes.

Ronald Buchinger für das ständige Ausborgen seiner Kamera und seine nette Gesellschaft, Kornelia Fischer und Walter Fritz für ihre Unterstützung in allen Werkstattangelegenheiten.

Christian Hoffelner für die Hilfe beim Layout.

Besonderer Dank gilt unseren Familien und Freunden für Hilfe, Beistand und Verständnis, und Inspiration während des ganzen Studiums.