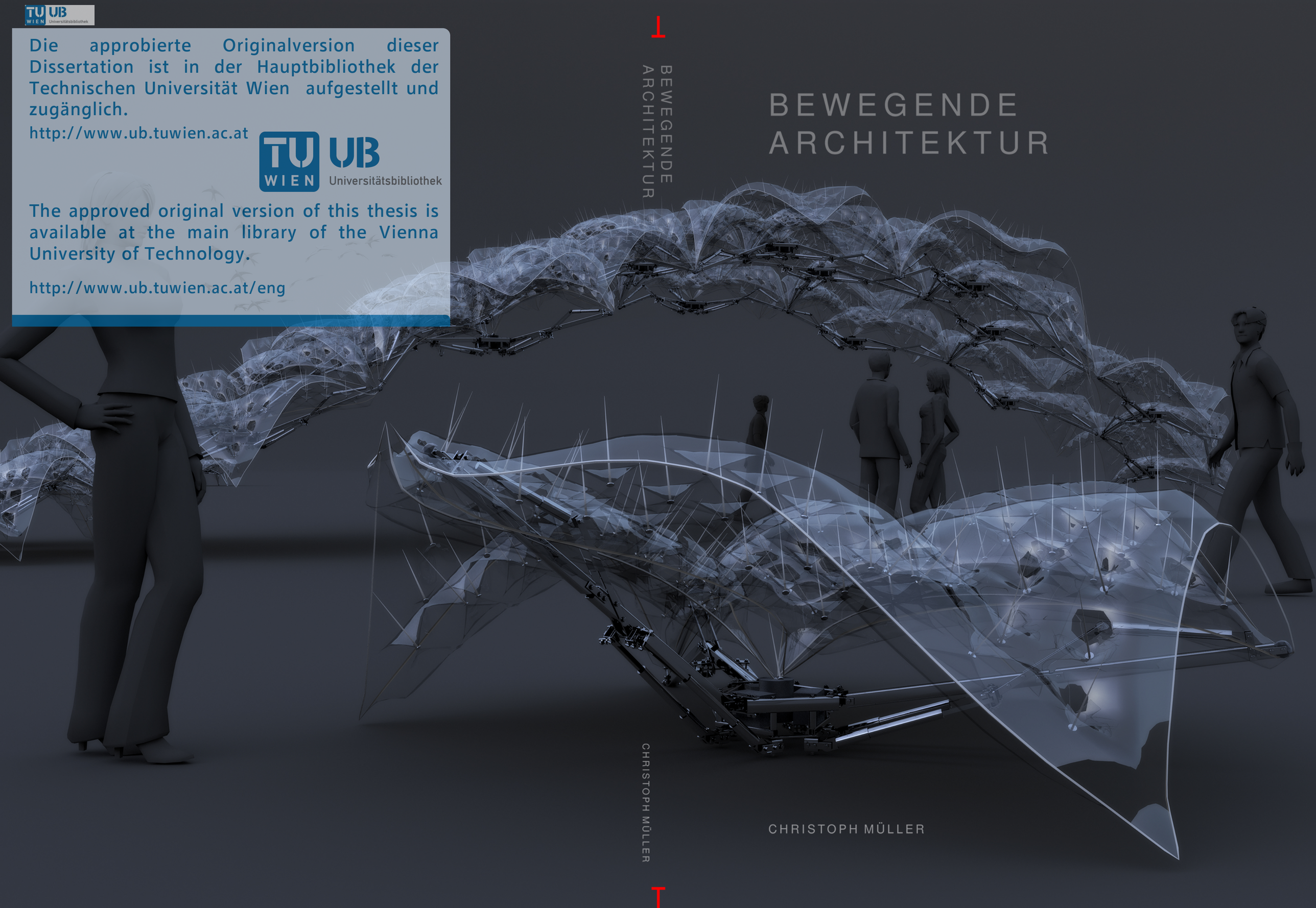
BEWEGENDE ARCHITEKTUR



BEWEGENDE ARCHITEKTUR

CHRISTOPH MÜLLER

BEWEGENDE ARCHITEKTUR

CHRISTOPH MÜLLER

Dissertation

B E W E G E N D E
A R C H I T E K T U R

ausgeführt zum Zwecke der Erlangung des akademischen Grades
eines Doktors der technischen Wissenschaften unter der Leitung von

Ao.Univ.Prof. Arch. Dipl.-Ing. Dr.techn. Manfred Berthold
Institut für Architektur und Entwerfen
E253 TU-Wien

und unter der Mitbetreuung von

Ao.Univ.Prof. Dipl.-Ing. Dr.techn. Margit Gföhler
Institut für Konstruktionswissenschaften und Technische Logistik
E307 TU-Wien

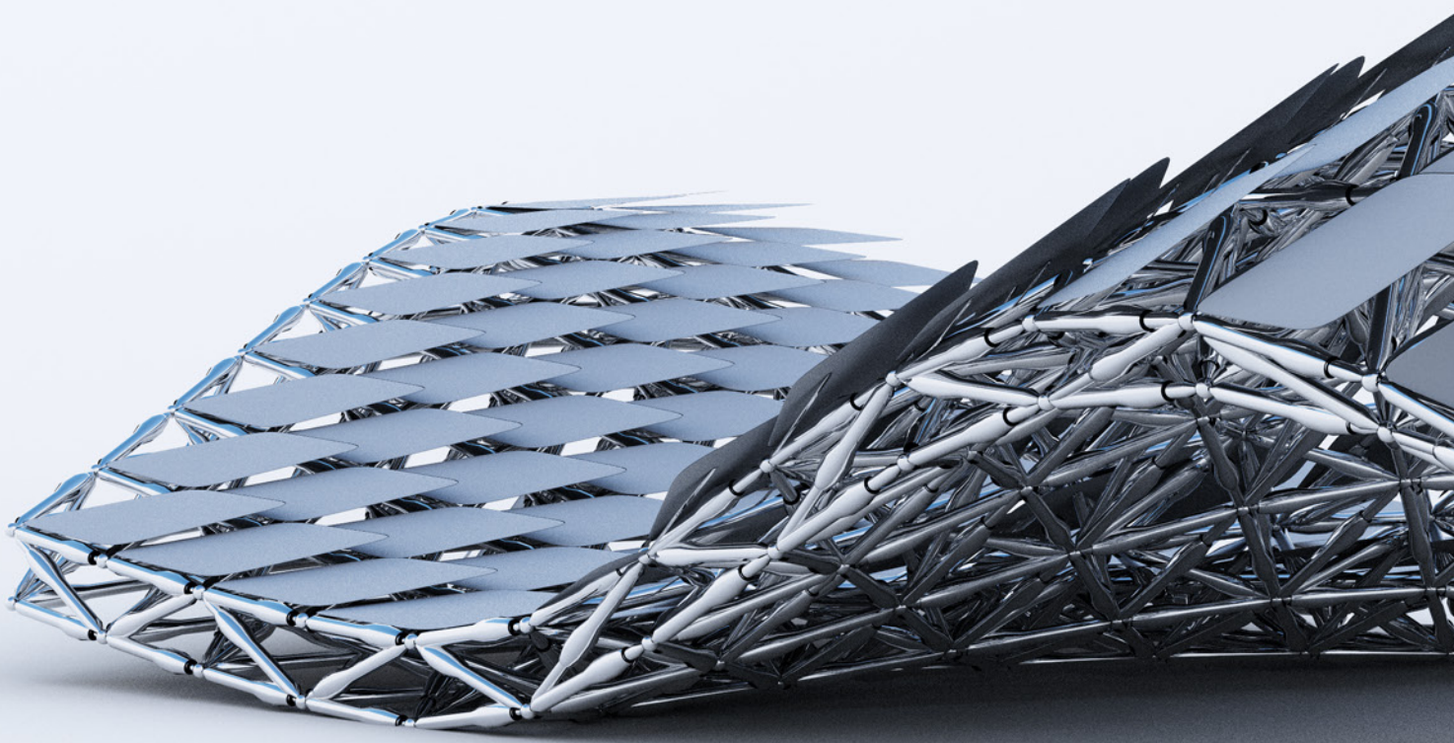
eingereicht an der Technischen Universität Wien Fakultät für
Architektur und Raumplanung von

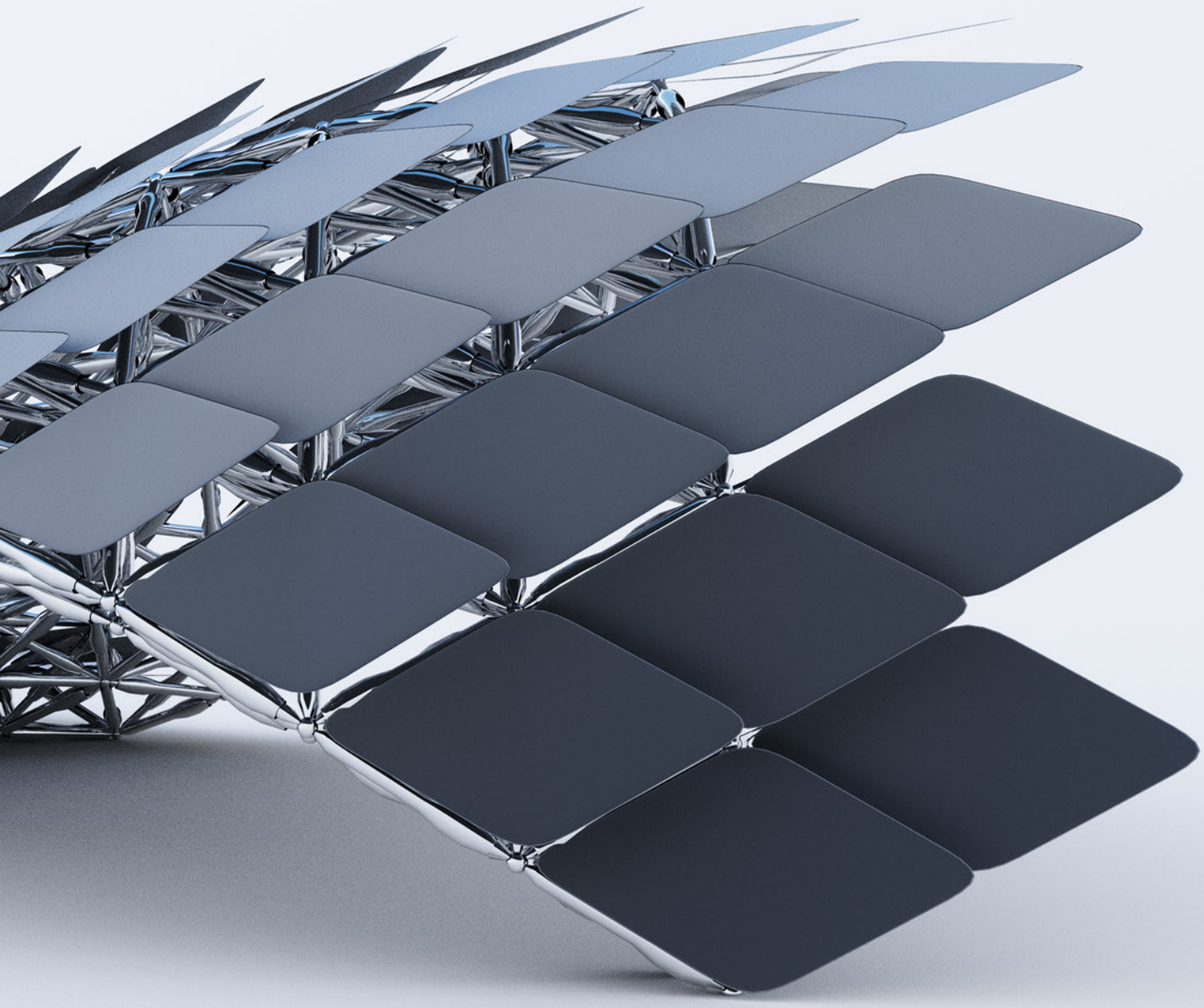
Univ.Lektor Dipl.-Ing. Christoph Müller BSc
Matrikelnummer 0728002
Eckertgasse 10 / 13
1100 Wien, Österreich

Wien, 12.10.2016

TECHNISCHE
UNIVERSITÄT
WIEN

Folgeseite:
Abb.0.0.0-1 Teaser





BEWEGENDE
ARCHITEKTUR

Entwicklung interaktiver und adaptiver Architektur mittels selbstorganisierenden Robotermodule

ABSTRACT

Kultur hat sich gewandelt. Vom Orchester in der Philharmonie zu Jugendlichen in Boxershorts, die sich mit Tablets ihre eigene Musik zusammenklicken. Die Grenzen des Museums, die Schwellen zwischen Kunst und Alltag, sind schon lange keine eindeutigen mehr. Die Serienproduktion als ein Schlüssel zur Moderne wurde zur robotischen Massenproduktion. Der Gegenpol ist eine Do-it-yourself-Bewegung, die sich eigene Alltagsgegenstände 3D-drucken kann und Lösungsvorschläge für sämtliche Lebenslagen auf Youtube bereit hält. Neben frontalen Medien wie dem Fernseher entstanden soziale Netzwerke, die jedem Einzelnen ein persönliches Zepter verliehen. Der digitale Raum hat den physischen erweitert und veränderte ihn nachhaltig. Gesellschaft, Kunst und Technologie sind komplexe Konstrukte geworden, die wechselwirkend ineinander greifen. Flexible, adaptive, dynamische und hierarchielose Organisationsprinzipien treffen auf bodenständige Vorstellungen. Der Zahn der Zeit nagt an den Grundfesten. Die Immobilie ist ein Auslaufmodell der Architektur.

Gegenstand von Bewegende Architektur ist die Entwicklung von beweglichen Architekturmodulen, abgekürzt bAm. Jeder einzelne dieser Roboter verfügt über einfache künstliche Intelligenz. Diese ermöglicht Umgebung zu erkennen, mit Menschen zu kommunizieren, sich kraftschlüssig mit baugleichen Nachbarn zu verbinden und schließlich interaktiven Raum zu erzeugen.

Dabei gibt es keine übergeordneten Regelwerke. Sämtliche Interaktion basiert auf der künstlichen Intelligenz der einzelnen Roboter, die sich durch maschinelles Lernen eigenständig weiterentwickelt.

Hierarchie, Organisation und Typologie werden dezentralisiert. Die Geometrie ist in dieser Architektur nicht vordefiniert oder gesteuert, sondern gestaltet und kontrolliert sich selbst. Besondere Stärke ist die Reaktion auf sich verändernde Umstände. Selbst wenn die Architektur starken Einflüssen ausgesetzt wird, können die einzelnen Roboter durch Aufspaltung oder Bündelung neue Anordnungsmöglichkeiten finden. Es existiert kein Zentrum, kein Raster und keine Symmetrie, die dabei gebrochen werden könnte.

Ist die Funktion einzelner Module eingeschränkt, erkennen diese die Fehlerquellen eigenständig und versuchen sich zu regenerieren. Sind einzelne Module defekt, klinken sie sich aus der Konstruktion aus und Nachbarn füllen die Lücke. Adaptive, selbstorganisierende und lernfähige Architektur entsteht. Das Design der bAm ist zudem so einfach gehalten, dass die Roboter auch von Laien repariert, angepasst und weiter entwickelt werden können.

Die bAm sind nicht auf Standpunkte und Standorte im wortwörtlichen Sinne fixiert. Bewegende Architektur ist auf vielen Ebenen, der Wunsch einer Loslösung von der Immobilie, hin zu lebendiger Architektur.

Culture has changed. From the orchestra in the concert hall, to youngsters in underpants, sitting there with tablets, making their own music with a few clicks. The borders of the museums, the threshold between art and ordinary, vanished a long time ago. Serial production, as a main key to modernism, has changed into the robotic mass production. The counter proposal is a do-it-yourself-movement, generating its own tools for daily life with 3D-printers and receiving tutorials for all possible situations on Youtube. Besides frontal media like television, there has been a rise of social media, enabling anyone to be chief editor. The digital space has expanded and changed the physical one fundamentally. Society, art and technology have become complex constructs, that are interweaved in strong interdependency. Flexible, adaptive and dynamic principles of organization without hierarchy, face the conservative mindset. The ravages of time take their toll on well-known fundamentals. The seemingly solid real estate is an obsolet model in architecture.

The Goal of the research is the development of moving architectural modules. Every single robot, called bAm (bewegende Architekturmodule), has got its own, simple, artificial intelligence. This artificial intelligence enables the bAm to recognize context, communicate with humans, connect with other bAm and finally create interactive space.

There are no overall rules. Every interaction is based on the artificial intelligence of each single bAm. With machine learning, the robots are able to develop themselves further autonomously.

Hierarchy, organization as well as typology are decentralized. The geometry of this architecture is not predefined or controlled in advance. It is happening, following clear rules. A big advantage of this kind of architecture is the adaptivity when confronted with changing context. With disconnecting and reconnecting, the single robots can even react on enormous changes of environment. There is no visible center, no grid and no symmetry that could be broken.

A bAm is able to detect a wide range of possible faults on its own, and tries to fix them autonomously. If a bAm is damaged, it is able to disconnect from the structure. Nearest neighbors will fill the gap. Adaptive, self-organizing and learning architecture is created. The design of the bAm is kept simple. Even amateurs can repair, adapt, or further develop them.

The bAm are not bound to point of views. They are not fixed on sites. In many ways, Bewegende Architektur is the idea of bringing architecture to live.

ENLIVENED ARCHITECTURE

Development of interactive and adaptive architecture with self organizing robot modules

INHALTSVERZEICHNIS

1 EINLEITEND

1.0.1 FORSCHUNGSFELD

Kinematische Architektur ist schon seit geraumer Zeit Gegenstand einer Dauerkontroverse in der Architekturgeschichte. Konzepte in allen Maßstäben wurden postuliert: vom tragbaren Büro im Koffer bis hin zu nomadischen Metropolen. Flexibilität, Platzersparnis und Materialeffizienz sind Beispiele für den Mehrwert, den Architektur leisten soll, wenn sie beweglich ist. Das Paradoxe, mit Starrheit Assoziiertes in Bewegung zu denken, ist gewiss auch ein emotionaler Anreiz.

Viele Ideen starben jedoch an den mangelnden Möglichkeiten der Umsetzung. Durch computertechnologischen Fortschritt und Weiterentwicklung von Steuerung und Regelungstechnik sind diese Ansätze heute greifbar geworden. Unter dem Titel „Bewegliche Tragwerke“ arbeitet beispielsweise das Team des Lehrstuhls für Tragwerksplanung an der TU-München an diesem Thema. Die Forschungsgruppe „Hyperbody“ an der TU-Delft ist eine bekanntee Vertreterin von Seiten der architektonischen Forschung. Darüber hinaus gibt es immer mehr gebaute Projekte, bei denen sich Bauteile oder gar ganze Gebäude verändern können.

Diese Arbeit liegt im Forschungsfeld und Interessengebiet der Schwarmarchitektur. Zu nennen sind dabei die Schlagworte: M-Blocks, Claytronics und Self-Reconfigurabele Robots.

1.0.2 METHODE

Interaktive Architektur ist ein Teilgebiet das viele Grundlagen aus unterschiedlichen Disziplinen hat. Architektur, Maschinenbau, Tragwerksplanung, Mathematik und Informatik sind nur einige davon.

Die Arbeit versucht zunächst essentielle Grundlagen in Teilbereiche aufzudröseln und separat zu diskutieren. Dazu zählen Themen wie die Konstruktion der beweglichen Einzelelemente, die zu Grunde liegenden Steuerungen, mögliche Hüllen, Knotenpunkte und Gelenke, statische Berechnungen und Weiteres.

„Here I am. Doing science."

Jake Sully im Spielfilm „Avatar" aus dem Jahre 2009

Jedoch stehen diese Teilbereiche in Bezug zueinander. Konstruktive Vorteile müssen nicht zwangsläufig mit funktionalen, architektonischen Ansprüchen einhergehen. Darüber hinaus ist es häufig nicht möglich, diese Zielkonflikte allgemein zu formulieren. Zweiter Zugang ist daher eine Rückführung und Zusammenfassung der Teilbereiche in ein Projekt. Im Rahmen einer bau-künstlerischen Forschung wird mittels zeitgenössischen Formfindungsmethoden ein Referenzmodell geschaffen.

„Aber gerade auf ihrer Verbindung, auf ihrer Einheit beruht Architektur. Konstruktion und Material sind die materiellen Voraussetzungen der architektonischen Gestaltung. Stehen in steter Wechselbeziehung."

Ludwig Hilberseimer, „Konstruktion und Form", 1924

Die Arbeit setzt darüber hinaus auf eine Verschränkung von digitalen und analogen Methoden. Während der Arbeit wurde neben Scripting, digitalen Simulationswerkzeugen und Analyseprogrammen, besonderer Wert auf analoge Modelle gelegt. Verwoben wird: Die Erkenntnis aus akademischer Forschung, künstlerischer und technischer Praxis sowie kollektivem Wissen in Form von Internet-Foren und Open-Source-Codes.

Schließlich entsteht ein Katalog in Form von Studien und Varianten, der unterschiedliche architektonische Qualitäten aufzeigt. Im Lauf der Forschung kristallisierten sich mögliche Maßstäbe, Einsatzbereiche und ein potentieller Mehrwert der Struktur heraus.

Verweis auf:
Potenzial, Seite 264

Mehrwert der Arbeit selbst ist ein Referenzmodell, das als Grundlagenforschung für andere Arbeiten dienen kann. Ziel ist: Das Forschungsfeld von selbstorganisierenden Architekturmodulen voranzutreiben. Sämtliche verwendete Software ist kostenfrei verfügbar und überwiegend open-source. Die zugrunde liegenden Entwicklungsplattformen sind ebenso kostengünstig erhältlich und von sehr aktiven Gemeinschaften und Hilfeforen unterstützt. Ausgewählte Materialien im Bereich Details sind auf hohe Verfügbarkeit sowie einfache Umsetzbarkeit ausgelegt. Erzeugte Scripts werden nach Abgabe der Arbeit als Download angeboten. Einfache Videos zur Dokumentatiion sollen im Laufe der Zeit entstehen. Eine möglichst breite Basis an Lesern und Anwendern soll angesprochen werden.

Der Text der Arbeit ist so einfach wie möglich formuliert. Er dient der begleitenden Dokumentation der einzelnen Schritte.

1.1 BEGLEITENDE STUDIEN

„Digital Grotesque is between chaos and order, both natural and the artificial, neither foreign nor familiar. Any references to nature or existing styles are not integrated into the design process, but are evoked only as associations in the eye of the beholder.“

Konzepttext von Michael Hansmeyer und Benjamin Dillenburger, „Digital Grotesque“, 2013

Es waren Statements wie jene von Michael Hansmeyer, die in mir Sehnsucht weckten. Digitale Formfindung ist kein Zugang, der konventionellen Architekturentwurf lediglich beschleunigen wird. Architekturentwurf, und mehr und mehr auch die gebaute Praxis, werden sich durch den Einzug des Computers weiterhin gravierend verändern. Was der Stahlbeton für die Moderne war, kann für unsere Generation der Computer werden. Von einer ungeliebten Technologie, die lediglich schneller und billiger ist, zu einer grundlegend neuen Architektur.

Vor und während der vorliegenden Arbeit entstand eine Vielzahl an digitalen Experimenten. Diese sehr freien Studien dienten der Erforschung verschiedener architektonischer Themen. Dazu zählen Logik, Struktur, komplexe Prozesse, Verhandlung von Raum, Hierarchie, Haptik und weitere Themen. Nicht zuletzt war das Austesten von Hard- und Software ein wichtiges Ziel.

Viele Kernideen der Dissertation leiten sich unmittelbar aus diesen Zugängen ab. Daher wird eine Auswahl von drei Experimenten genauer erläutert.

1.1.1 TOPOLOGIE

Klassische Architektur kennt klassische Ordnungsprinzipien. Dazu zählen die Grundrechenarten: Addition, Subtraktion, Multiplikation und Division. Zahlenmodelle wie Fibonacci-Folge, goldener Schnitt und Modulor leiten sich unmittelbar aus diesen ab.

Mit Einzug des Computers wird die Architektur mit neuen Ordnungsprinzipien konfrontiert. Perlin-Noise und Voronoi sind bekannte Beispiele dafür. Die Studie versucht neue Topologien für Formen zu erforschen. Alle Ergebnisse basieren auf der gleichen Grundlage: eine rechteckige Fläche, die mehrmals in kleinere Teile zerlegt wurde. Verschiedene Punkte werden definiert, an denen sich die Fläche aufwölbt. Anschließend wird die Fläche, abhängig von der Verformung, neu unterteilt. An Stellen mit weniger Steigung entstehen weniger neue Flächen. An Stellen mit mehr Steigung entstehen mehr neue Flächen. Dieser Prozess ist üblicherweise unsichtbar. Bei dieser Studie wird der Effekt, durch Extrusion der entstanden Kanten, bewusst hervorgehoben.

Es entsteht unerwartete Geometrie. Diese ist aber keinesfalls zufällig. Vielmehr wird die zugrunde liegende Logik inhärent spürbar. Der selbe Prozess, mit abgeänderten Parametern, führt zu unterschiedlichen Ergebnissen. Der Prozess kann Raum erzeugen, wie im oberen Ergebnis, objekthaft wirken wie im zweiten, oder ein Hybrid aus beiden werden, wie im dritten Beispiel.

Folgeseite:
Abb.1.1.1-1 Variante 4

Studie zu Topologie
Unterteilung anhand von Attraktoren
in verschiedenen Varianten

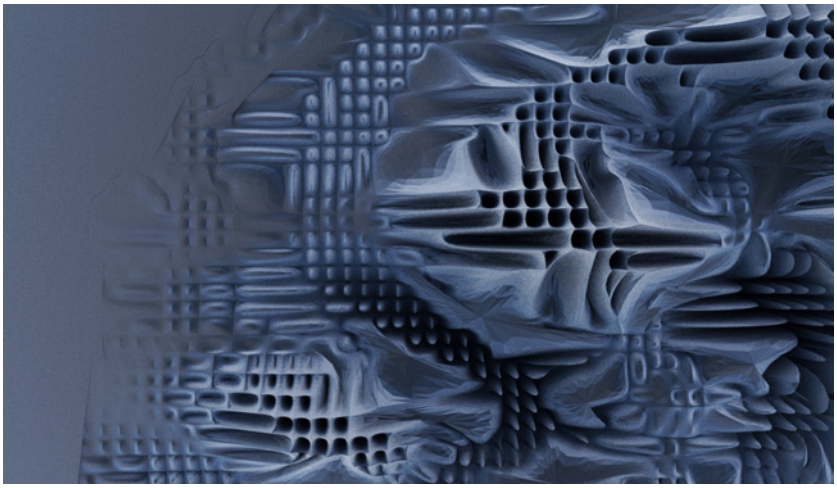


Abb.1.1.1-2 Variante 1

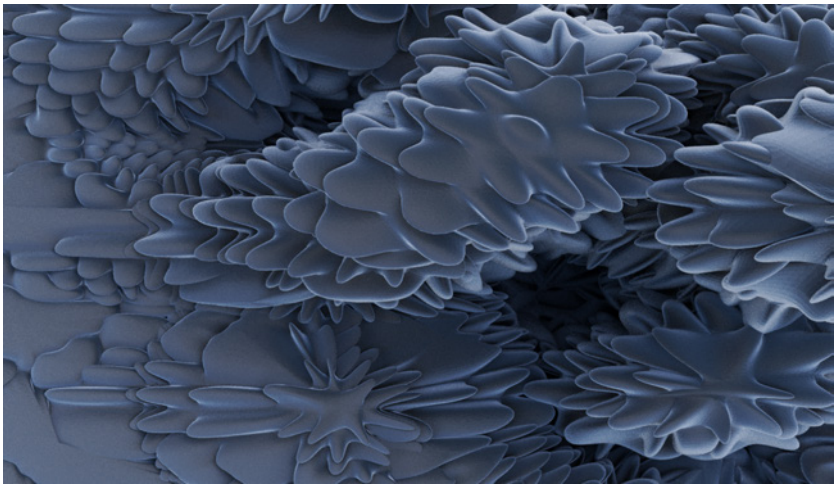


Abb.1.1.1-3 Variante 2

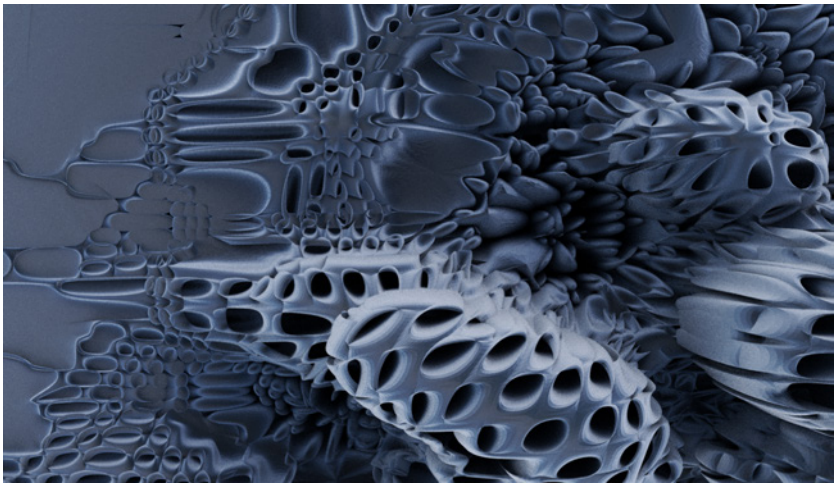
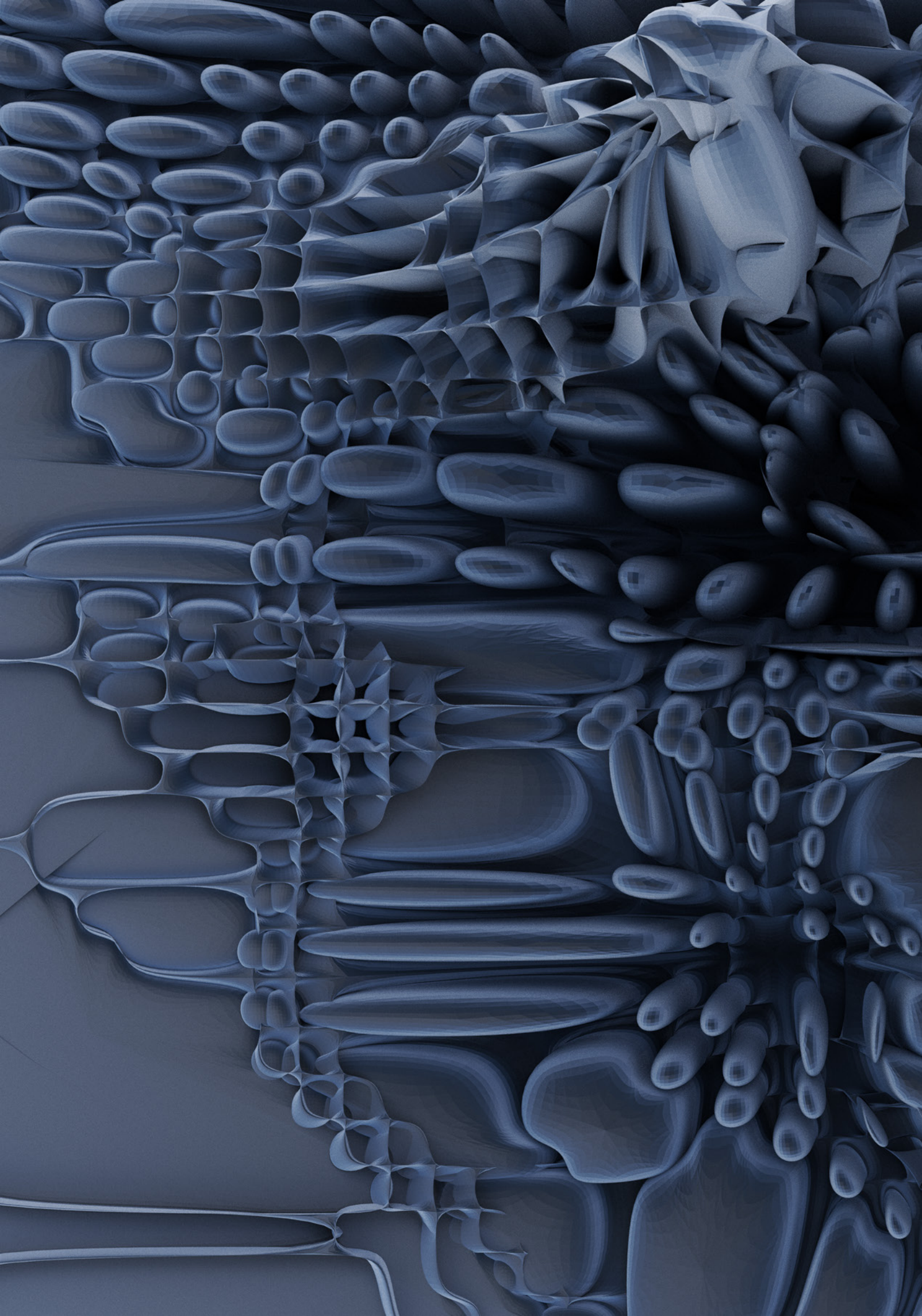
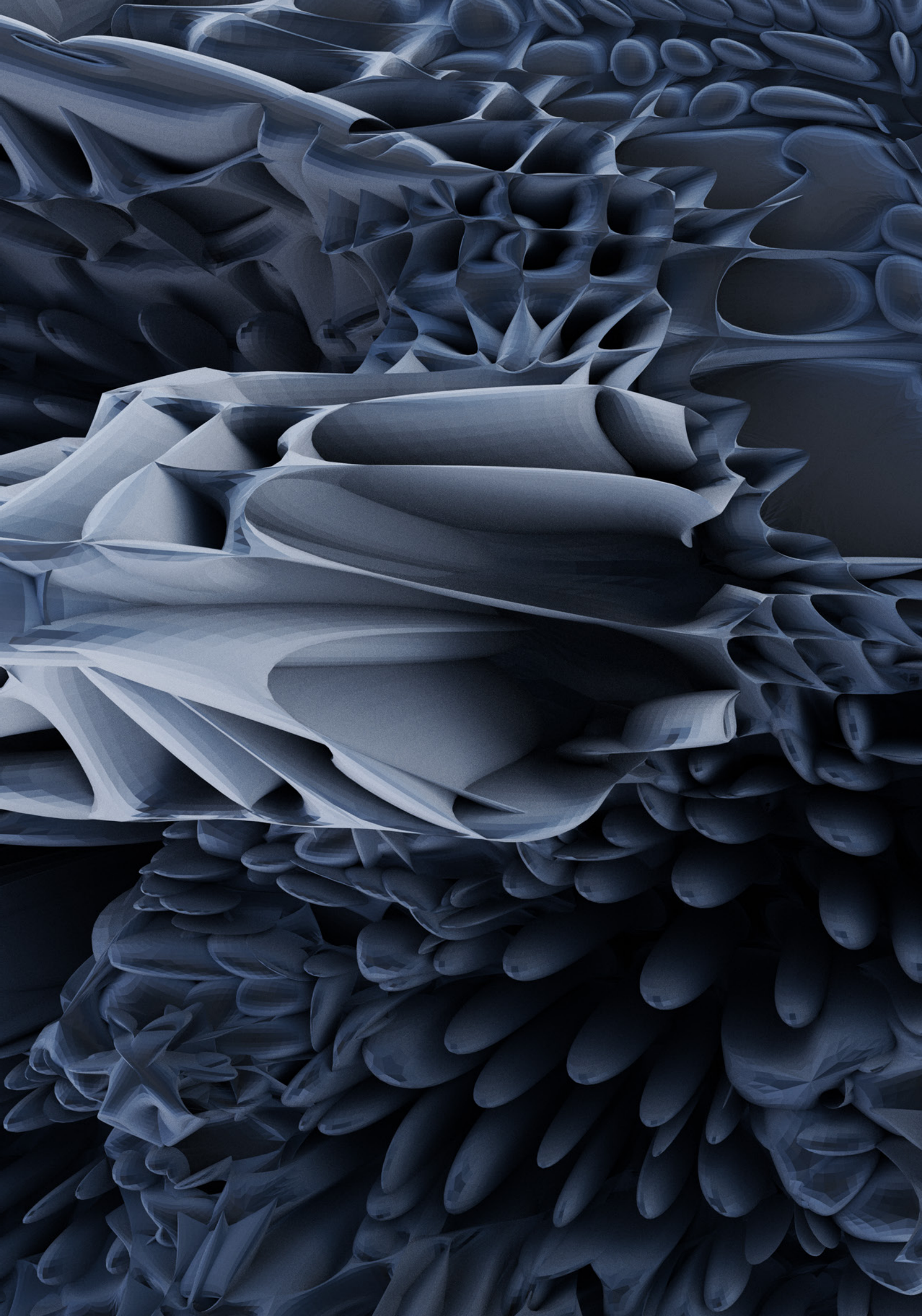


Abb.1.1.1-4 Variante 3





1.1.2 H I E R A R C H I E

Weitere zentrale Architekturthemen sind Maßstab, Materialoberfläche und Wiederholung.

Mittels Scripting wurde ein Teil der Ausgangsgeometrie mehrmals unterteilt. Bei jeder Unterteilung werden die neu entstandenen Flächen extrudiert und leicht verdreht. Jede Unterteilung basiert dabei stets auf den gleichen Parametern für Extrusion und Rotation. Bereits mit einer geringen Anzahl an Iteration entsteht eine äußerst komplexe Geometrie. Die linke Seite des unteren Ergebnisses verläuft fließend von solidem Volumen zu weichem Fell.

Weder der Anfang noch das Ende der erzeugten Geometrie wird vom Betrachter als Zwischenraum interpretiert. Der Anfang ist Volumen und das Ende ist annähernd Nichts. Im Bereich dazwischen entsteht raumbildende Geometrie.

Folgeseite:
Abb.1.1.2-1 Detail

Studie zu Hierarchie
von Volumen zu Fell
in mehreren Iterationen

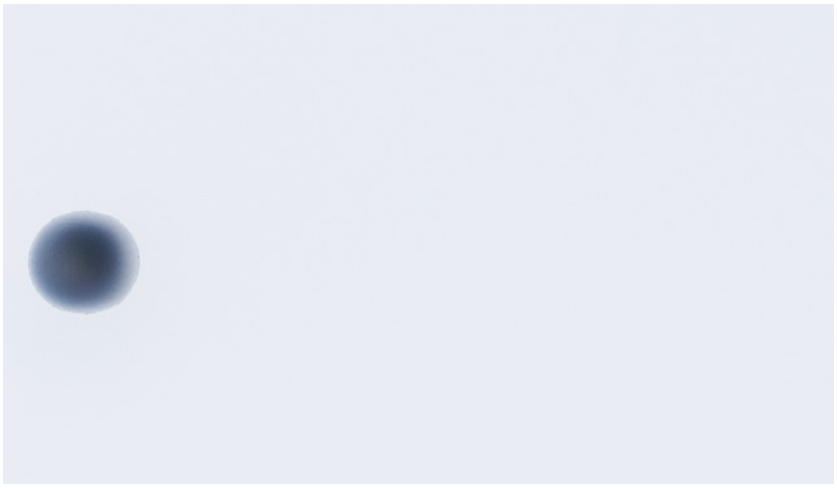
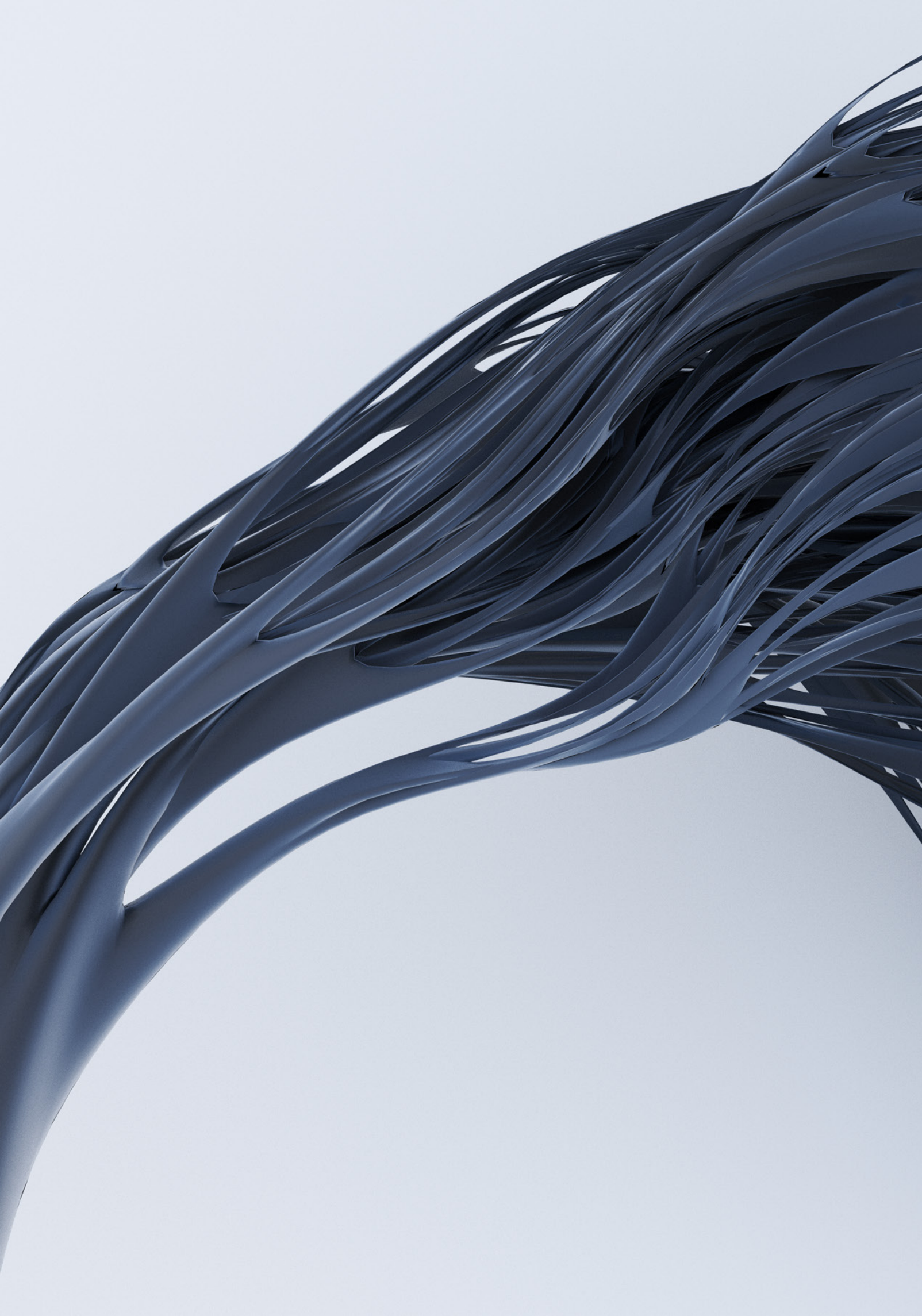
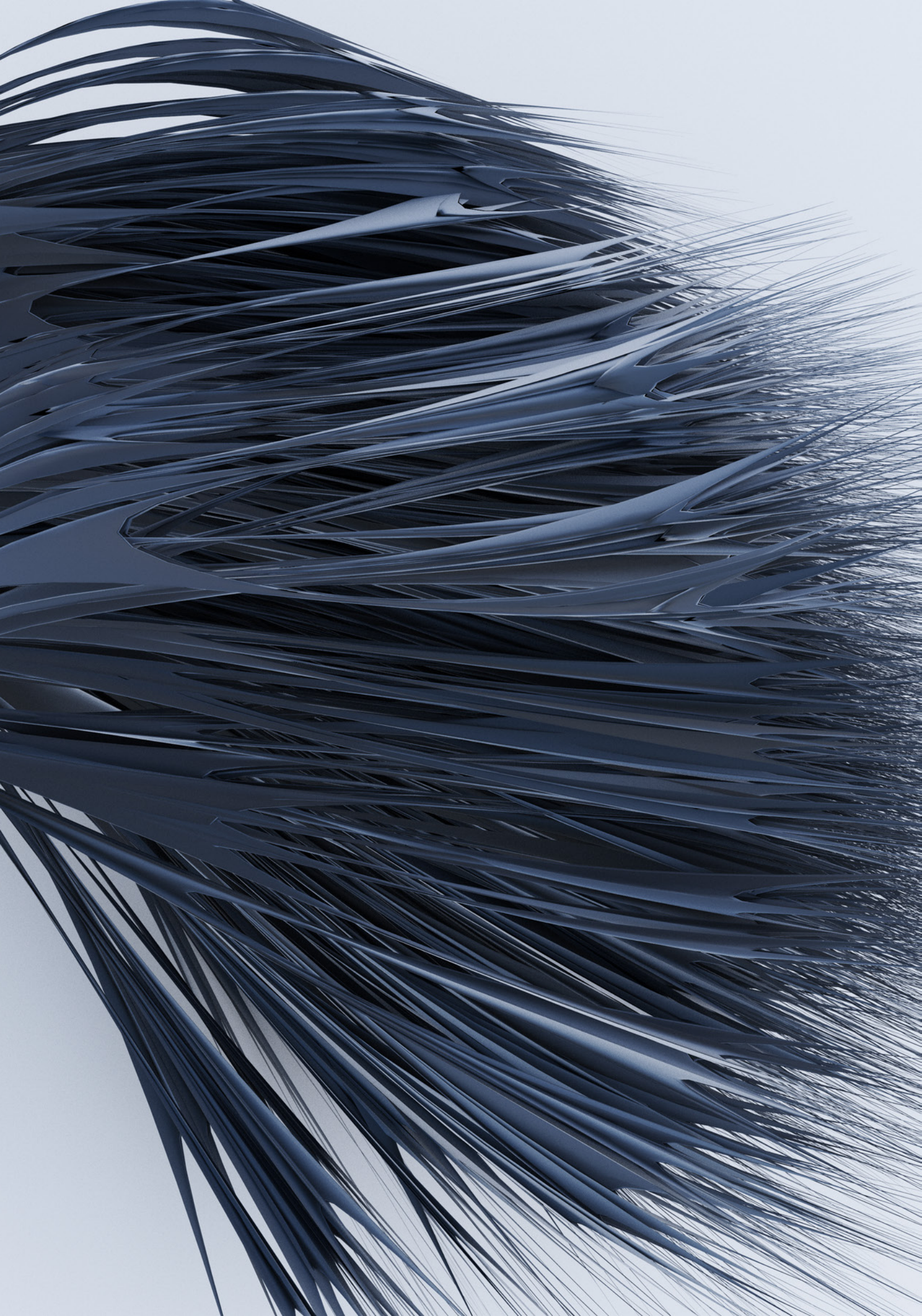


Abb.1.1.2-2 Ausgangsgeometrie

Abb.1.1.2-3 Zwischenschritt

Abb.1.1.2-4 Zustand nach Ablauf des Scripts





1.1.3 PARTIKEL

Üblicherweise werden in der Computergrafik Modelle aus Flächen zusammengesetzt. Ein Gegenmodell sind Voxel. Dabei werden so viele Punkte aneinandergereiht, dass diese als zusammenhängende Form erscheinen. Es entsteht gewissermaßen ein dreidimensionales Pixelbild.

Diese Studie setzt sich mit der Aggregation von Einzelobjekten zu raumbildenden Strukturen auseinander. Zugrunde liegt ein Partikelsystem. Dieses erlaubt die Koordination einer hohen Anzahl von Punkten. Diese abstrakten Punkte können schließlich mit Geometrie ersetzt werden.

Ein Turbulenzfeld treibt diese Punkte im nächsten Schritt auseinander. Das Feld startet dabei von rechts und verläuft nach links. Das vorerst erzeugte Volumen wird dabei zunehmend stärker verformt. Der linke Teil des letzten Ergebnisses wird als raue Materialoberfläche gelesen, der mittlere Teil als verformte Geometrie und der rechte Teil schließlich als lose Einzelelemente. Der fließende Verlauf zwischen diesen Zuständen war eine bewusste Zielsetzung.

In Bezug auf die räumliche Wahrnehmung der Geometrie wird ebenfalls der mittlere Teil als raumbildend interpretiert werden.

Folgeseite:
Abb.1.1.3-1 Detail

Studie zu Partikel
Verlauf von Materialoberfläche zu Einzelelement
zu verschiedenen Zeitpunkten

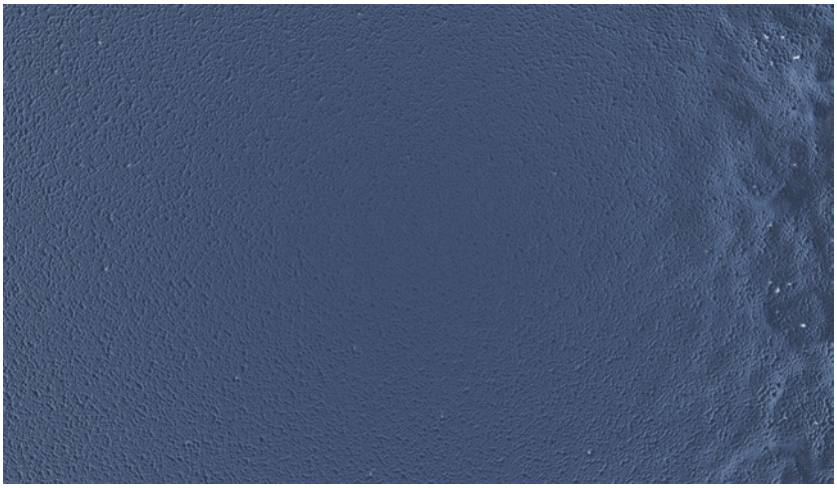


Abb.1.1.3-2 Zustand kurz nach Beginn

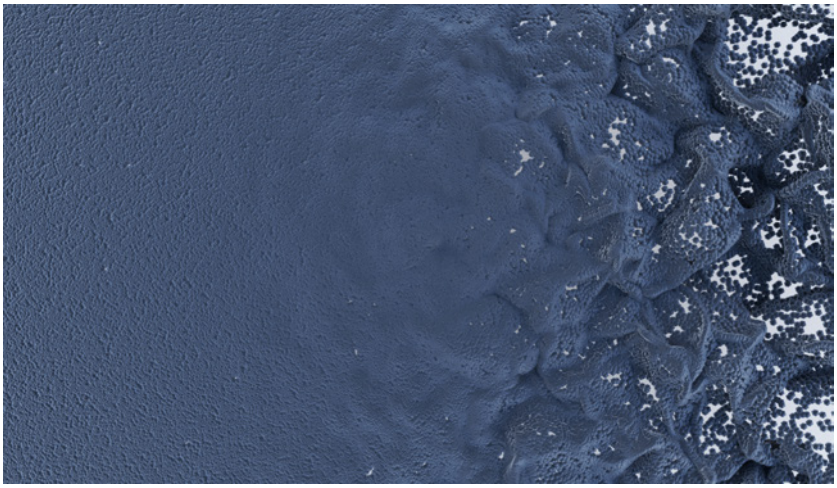


Abb.1.1.3-3 während des Prozesses

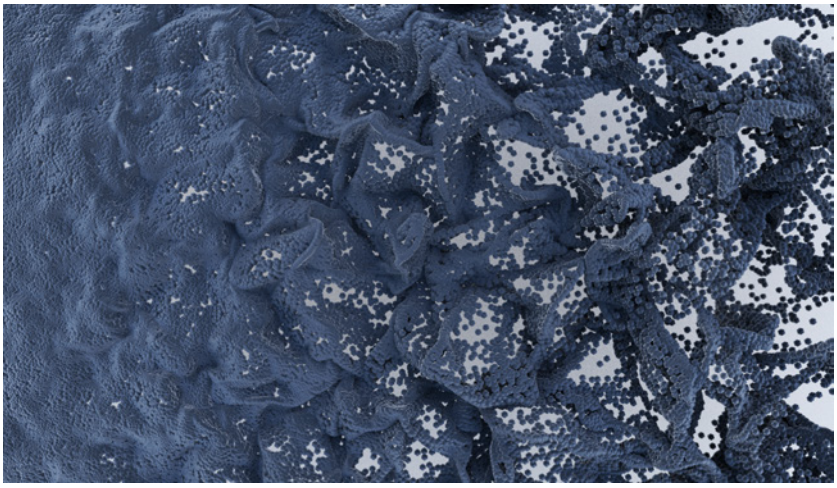
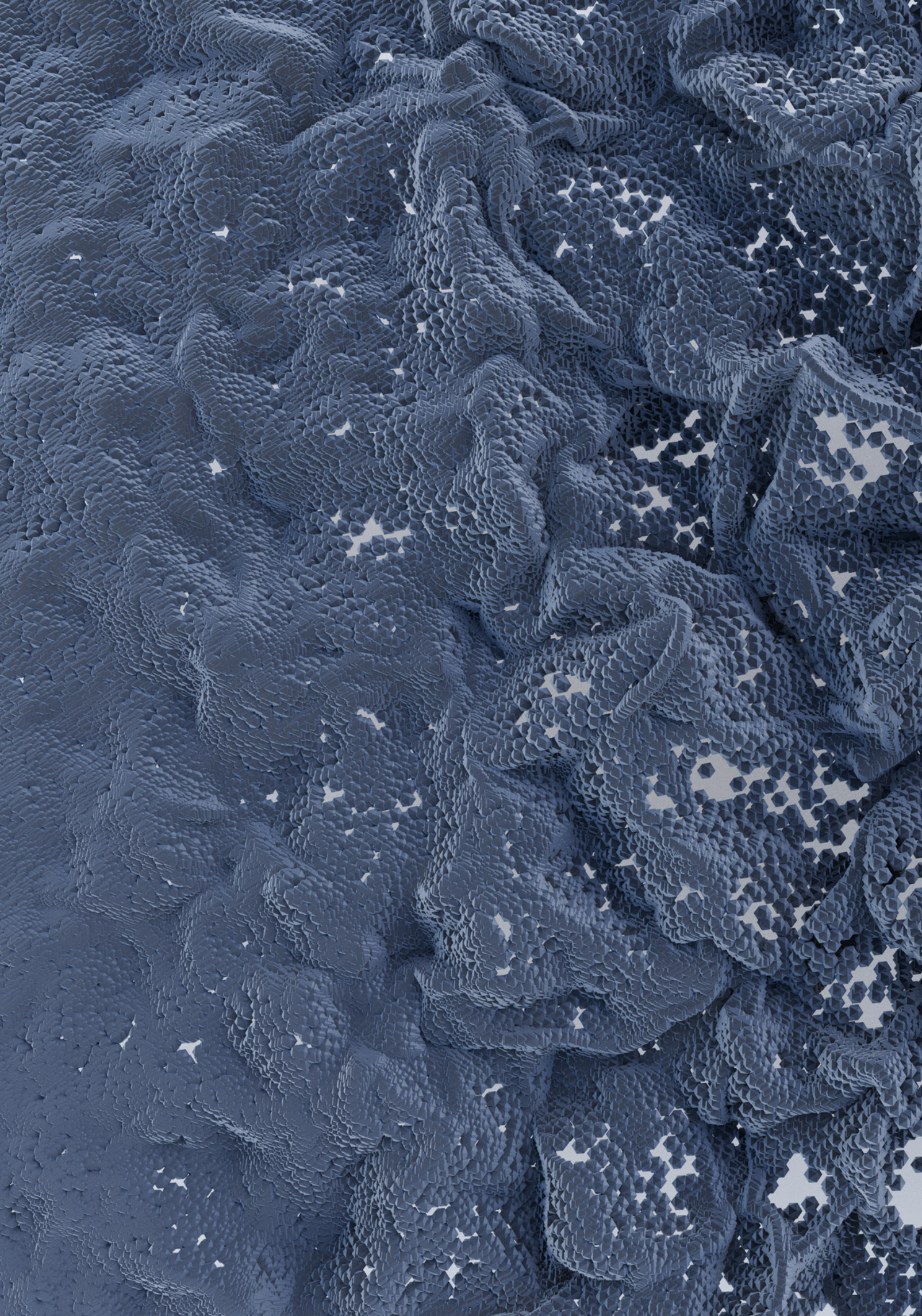
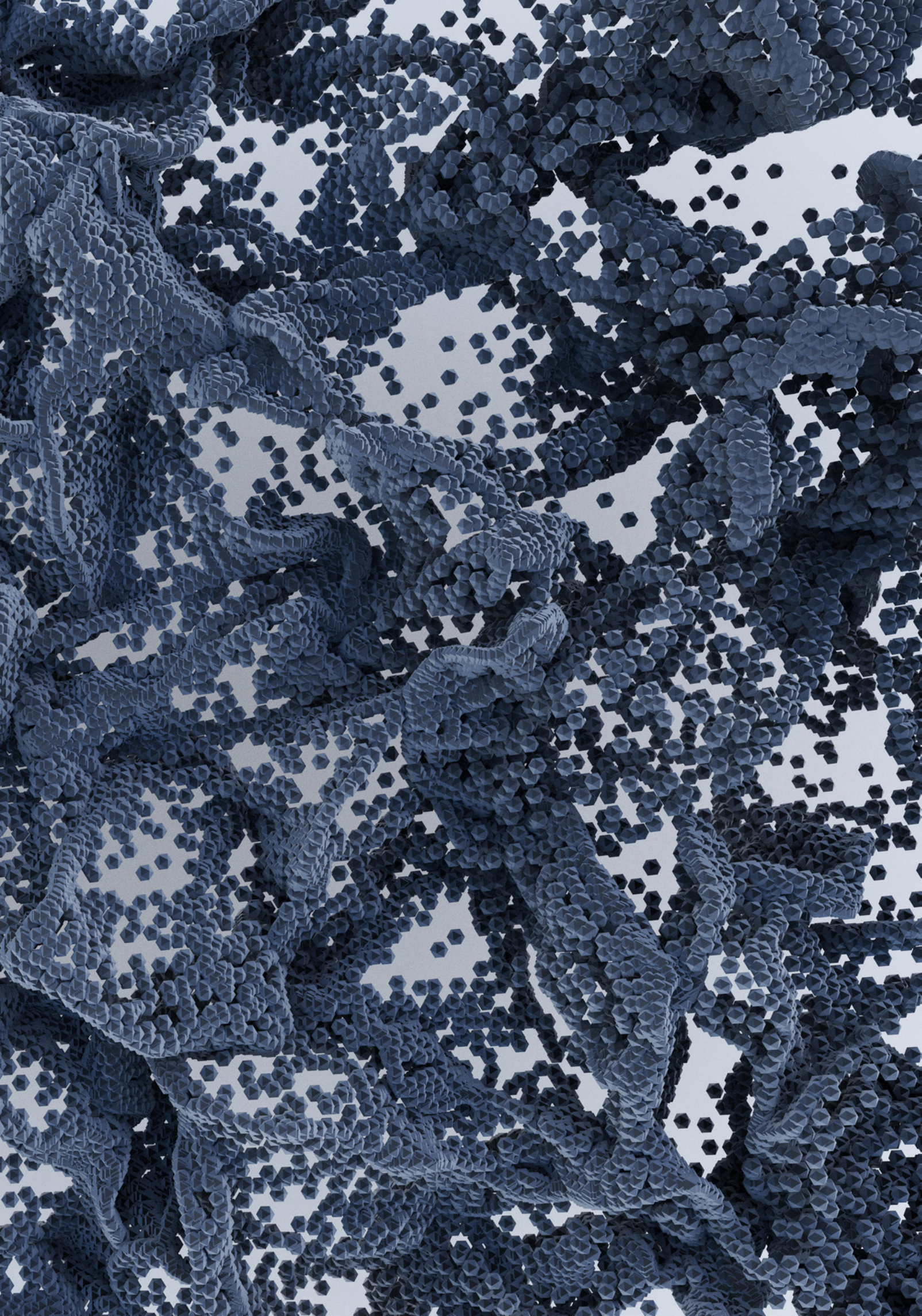


Abb.1.1.3-4 am Ende der Simulation





2 GRUNDLAGEN

Welche Teilbereiche können separat voneinander diskutiert werden? Welche Fragestellungen treten aus Sicht der Architektur dabei auf? Welche grundlegenden Entwurfszugänge sind möglich?

„Die Pfosten: Ein Problem auf wissenschaftlichem Wege lösen, heißt zunächst seine Elemente unterscheiden. Bei einem Bau kann man daher ohne weiteres die tragenden von den nichttragenden Teilen trennen.“

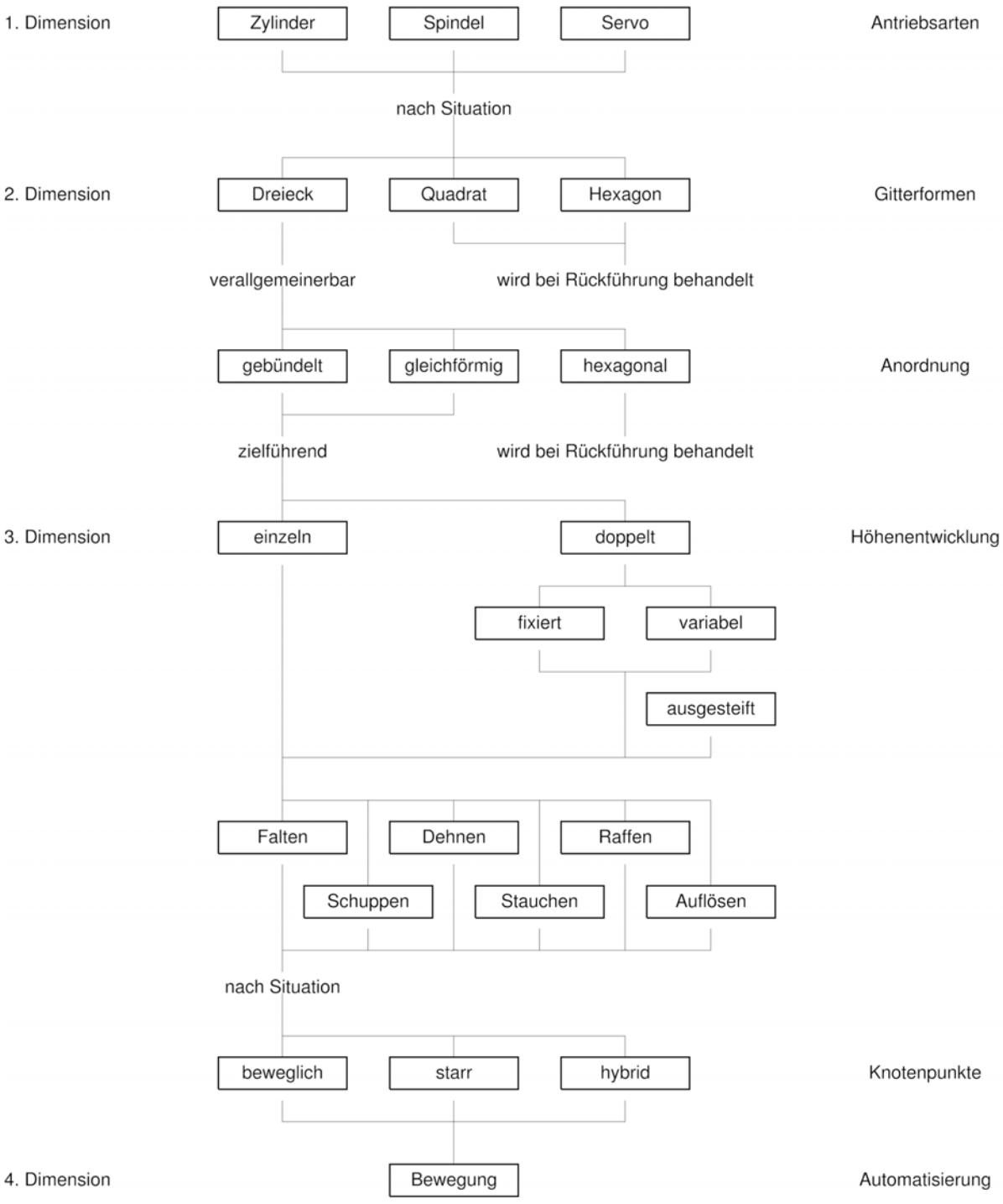
Le Corbusier und Pierre Jeanneret, „Fünf Punkte zu einer neuen Architektur“, 1927

2.1 TEILBEREICHE

Folgendes Organigramm zeigt den Aufbau des ersten Kapitels in Bezug auf die diskutierte Geometrie. Der Aufbau folgt den bekannten vier Dimensionen des kartesischen Raums inklusive Zeit. Begonnen wird bei der Linie, gefolgt von der Erweiterung zur Fläche, der Entwicklung zur Höhe und der abschließenden Diskussion von der möglicher Bewegung.

Parametrische und algorithmische Entwurfswerkzeuge erlauben es, Architektur zu generieren und ihre Bewegung zu simulieren. Da sämtliche Entwicklungsschritte mittels Scripting automatisiert sind, ist es möglich, den Entstehungsprozess auch nachträglich abzuändern und weiter zu verfeinern. Entdeckte Potenziale können so maximiert und Konsequenzen anhand der gesamten Struktur evaluiert werden.

Entwicklung
Organigramm der diskutierten Geometrie
Grundlage für Scripte



Diagr.2.1.0-1 Organigramm

2.2 EINZELELEMENT

Die Arbeit basiert auf dem Konzept des „Variable Geometry Truss“. Darunter wird ein Konzept verstanden, das ursprünglich als Kragarm in der Raumfahrt Verwendung finden sollte. Die Kernidee ist, statische Träger durch Linearmotoren zu ersetzen. Ein Hydraulikzylinder nimmt beispielsweise den Platz eines unbeweglichen Trägers in einem Fachwerk ein.

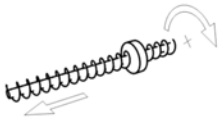
Koryo Miura, Hiroshi Furuya und Kenichi Suzuki, „Variable geometry truss and its application to deployable truss and space crane arm“, 1985

Wie können diese beweglichen Trägerelemente ausgebildet werden? Der Fokus dieses Abschnitts liegt auf der zugrundeliegenden Kinematik, die zur Steuerung erforderlich ist. Die Möglichkeit, bewegliche Architektur zu entwickeln, ist maßgeblich von diesen technischen Grundlagen beeinflusst. Aufgezeigt wird ein möglicher Zugang, wie bewegliche Einzelelemente in analogen und digitalen Modellen simuliert werden können. Erst so kann ihr Potenzial im konkreten Projekt evaluiert werden. Zudem werden Scripte beschrieben, die es ermöglichen, Bauteilbeziehungen automatisch herstellen zu können. Abschließend wird die Steuerung sowohl soft- als auch hardwareseitig erläutert und am Modell veranschaulicht.

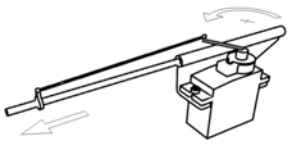
2.2.1 ANTRIEBSART

Die Studie rechts stellt mögliche Antriebsarten gegenüber. Hydraulik und Pneumatik würde bei einem architektonischen Maßstab in Frage kommen. Ebenso bringt ein Antrieb mit Spindel genügend Kraft auf, um größere Bauteile zu bewegen. Für die kleineren Prototypen werden Servoantriebe verwendet, da ihre Steuerung sehr einfach ist und eine kostengünstige Alternative darstellen.

Die Abbildung unten veranschaulicht, wie die Rotationsbewegung von Spindel- und Servoantrieb in ein Linearbewegung umgewandelt werden kann.



Diagr.2.2.1-1 Umwandlung Spindel



Diagr.2.2.1-2 Umwandlung Hebel

Mögliche Antriebsarten
Hydraulik, Spindel und Servo
Größenänderung von 1,1 auf 2,0 Einheiten

Abb.2.2.1-1 a,b,c Ausgangsposition

Abb.2.2.1-2 a,b,c 20 %

Abb.2.2.1-3 a,b,c 40 %

Abb.2.2.1-4 a,b,c 60 %

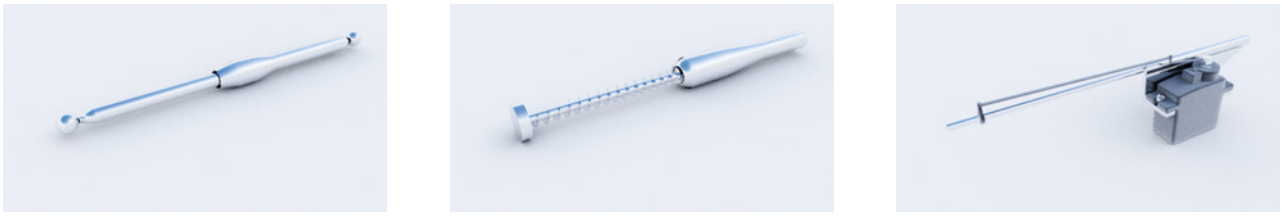


Abb.2.2.1-5 a,b,c 80 %

Abb.2.2.1-6 a,b,c Endposition

2.2.2 HYDRAULIK

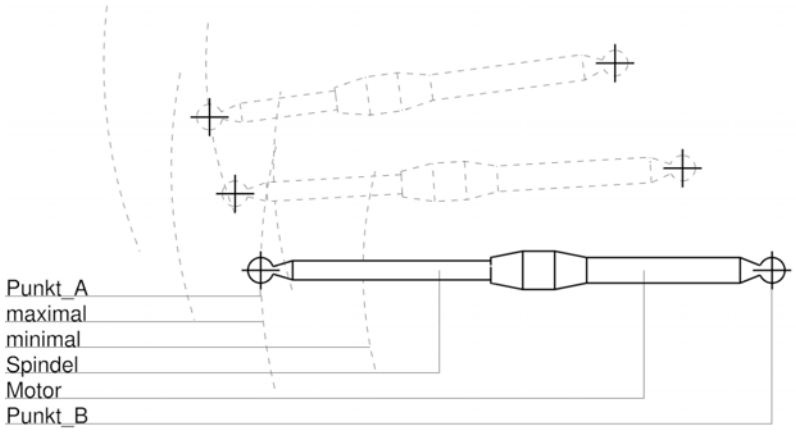
Hier wird die Kinematik von Kolben und Zylinder bei Hydraulik- und Pneumatikantrieben diskutiert. Im Bild unten ist das Ziel verdeutlicht. Um eine einfache Kontrolle der Modelle zu ermöglichen, soll der gesamte Antrieb anhand von zwei Bezugspunkten gesteuert werden können. Zwischen diesen Bezugspunkten sollen sich alle Bauteile selbständig aufspannen.

Verweis auf:
Blender, Seite 275

Wird die Lage von Punkt B verändert, muss der Ursprung des Zylinders die gleiche Position einnehmen. Die Software Blender bietet mit Constraints eine Lösung an, solche Beziehungen zu erstellen. Für den Zylinder wird die Beziehung Copy Location erzeugt und als Ziel Punkt B angegeben. Nun wird der Zylinder in Echtzeit die Position von Punkt B annehmen, wenn dieser verschoben wird. Zugleich muss er jedoch auch stets mit seiner Spitze UP Z auf den gegenüberliegenden Punkt A ausgerichtet sein. Das geschieht durch die Beziehung Track To. Für den Kolben funktioniert das gleiche Prinzip, nur umgekehrt.

Bis jetzt wird im Modell nicht berücksichtigt, wie weit ein Motor zusammen- oder ausgefahren werden kann. Deshalb werden für beiden Punkte die Beziehungen Min Distance und Max Distance erstellt, um den möglichen Abstand mit den eingesetzten Variablen Minimalabstand und Maximalabstand zu limitieren.

Anzumerken ist, dass diese Beziehungen im Programm auch manuell angelegt werden könnten. Für eine größere Anzahl an Elementen ist jedoch der Einsatz von Scripting weitaus effizienter. Zudem entsteht dadurch die Möglichkeit, sämtliche Schritte schnell mit anderen Parametern erneut auszuführen.



Zchng.2.2.2-1 Kinematik gesamt

Kinematik für Hydraulik und Pneumatik
Scriptsprache: Python
DCC: Blender

```python
import bpy

Maximalabstand = 2
Minimalabstand = 1.5

# Abhängigkeit Zylinder Position und Richtung
obj = bpy.data.objects
bpy.context.scene.objects.active = obj["Zylinder"]
bpy.ops.object.constraint_add(type='COPY_LOCATION')
bpy.context.object.constraints["Copy Location"].target = obj["Punkt_B"]

bpy.ops.object.constraint_add(type='TRACK_TO')
bpy.context.object.constraints["Track To"].target = obj["Punkt_A"]
bpy.context.object.constraints["Track To"].track_axis = 'TRACK_Y'
bpy.context.object.constraints["Track To"].up_axis = 'UP_Z'

# Abhängigkeit Kolben Position und Richtung
obj = bpy.data.objects
bpy.context.scene.objects.active = obj["Kolben"]
bpy.ops.object.constraint_add(type='COPY_LOCATION')
bpy.context.object.constraints["Copy Location"].target = obj["Punkt_A"]
bpy.ops.object.constraint_add(type='TRACK_TO')

bpy.context.object.constraints["Track To"].target = obj["Punkt_B"]
bpy.context.object.constraints["Track To"].track_axis = 'TRACK_Y'
bpy.context.object.constraints["Track To"].up_axis = 'UP_Z'

# Maximaler und minimaler Abstand
obj = bpy.data.objects
bpy.context.scene.objects.active = obj["Punkt_B"]
bpy.ops.object.constraint_add(type='LIMIT_DISTANCE')
bpy.context.object.constraints["Limit Distance"].name = "Maximalabstand"
bpy.context.object.constraints["Maximalabstand"].target = obj["Punkt_A"]
bpy.context.object.constraints["Maximalabstand"].distance = Maximalabstand
bpy.ops.object.constraint_add(type='LIMIT_DISTANCE')

bpy.context.object.constraints["Limit Distance"].name = "Minimalabstand"
bpy.context.object.constraints["Minimalabstand"].target = obj["Punkt_A"]
bpy.context.object.constraints["Minimalabstand"].distance = Minimalabstand
bpy.context.object.constraints["Minimalabstand"].limit_mode = 'LIMITDIST_OUTSIDE'
```

Scp.2.2.2-1 Einzelelement

$$Position = Position_B$$
$$Ausrichtung = Position_A$$

Frl.2.2.2-1 Kinematik Zylinder

$$Position = Position_A$$
$$Ausrichtung = Position_B$$

Frl.2.2.2-1 Kinematik Kolben

$$\overline{AB} \leq Maximalabstand$$
$$\overline{AB} \geq Minimalabstand$$

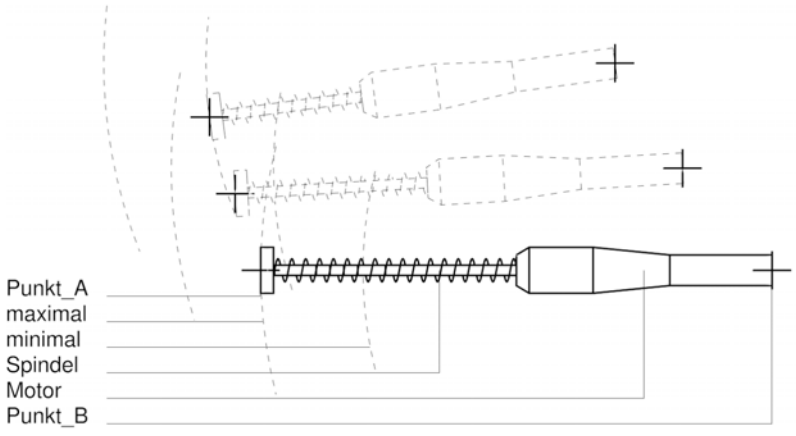
Frl.2.2.2-1 Kinematik Grenzen

2.2.3 SPINDEL

Für einen Antrieb mit einem Spindelmotor ist das Erstellen der Beziehungen identisch zu Hydraulik und Pneumatik. Lediglich die Objektnamen werden ersetzt.

Das lässt Optimierungspotenzial beim Scripting vermuten. Werden die gleichen Aufgaben lediglich mit anderen Objektnamen und Parametern ausgeführt, kann ein Script durch Funktionen und Klassen effizienter gestaltet werden. Beim Erstellen der Funktion werden Variablen für Objektnamen und weitere Parameter in den Handlungsablauf eingebaut. Später wird die Funktion aufgerufen und dabei mit den entsprechenden Informationen gefüttert. Klassen erlauben zudem, eigene Variablen innerhalb der Objekte zu erzeugen.

Diese allgemein anerkannte Praxis macht zudem möglich, dass verschiedene Personen vereinzelte Bausteine in Form von Scripten gemeinsam verwenden, teilen und verbessern können.



Zchng.2.2.3-1 Kinematik gesamt

Kinematik Spindel
Scriptsprache: Python
DCC: Blender

```python
import bpy

Maximalabstand = 2
Minimalabstand = 1.5

class Beziehung:
    def __init__(self, Name, Anfang, Ende):
        self.Name = Name
        self.Anfang = Anfang
        self.Ende = Ende

    def Abhaengigkeit (self):
        obj = bpy.data.objects
        bpy.context.scene.objects.active = obj[self.Name]
        bpy.ops.object.constraint_add(type='COPY_LOCATION')
        bpy.context.object.constraints["Copy Location"].target = obj[self.Anfang]
        bpy.ops.object.constraint_add(type='TRACK_TO')
        bpy.context.object.constraints["Track To"].target = obj[self.Ende]
        bpy.context.object.constraints["Track To"].track_axis = 'TRACK_Y'
        bpy.context.object.constraints["Track To"].up_axis = 'UP_Z'

    def Abstand (self):
        obj = bpy.data.objects
        bpy.context.scene.objects.active = obj[self.Anfang]
        bpy.ops.object.constraint_add(type='LIMIT_DISTANCE')
        bpy.context.object.constraints["Limit Distance"].name = "Maximalabstand"
        bpy.context.object.constraints["Maximalabstand"].target = obj[self.Ende]
        bpy.context.object.constraints["Maximalabstand"].distance = Maximalabstand
        bpy.ops.object.constraint_add(type='LIMIT_DISTANCE')
        bpy.context.object.constraints["Limit Distance"].name = "Minimalabstand"
        bpy.context.object.constraints["Minimalabstand"].target = obj[self.Ende]
        bpy.context.object.constraints["Minimalabstand"].distance = Minimalabstand
        bpy.context.object.constraints["Minimalabstand"].limit_mode = 'LIMITDIST_OUTSIDE'

# Abhängigkeit
Motor = Beziehung("Motor", "Punkt_B", "Punkt_A")
Motor.Abhaengigkeit()
Spindel = Beziehung("Spindel", "Punkt_A", "Punkt_B")
Spindel.Abhaengigkeit()

# Abstand begrenzen
Motor.Abstand()
```

Scp.2.2.3-1 Einzelelement

$$Position = Position_B$$
$$Ausrichtung = Position_A$$

Frl.2.2.3-1 Kinematik Motor

$$Position = Position_A$$
$$Ausrichtung = Position_B$$

Frl.2.2.3-1 Kinematik Spindel

$$\overline{AB} \leq Maximalabstand$$
$$\overline{AB} \geq Minimalabstand$$

Frl.2.2.3-1 Kinematik Grenzen

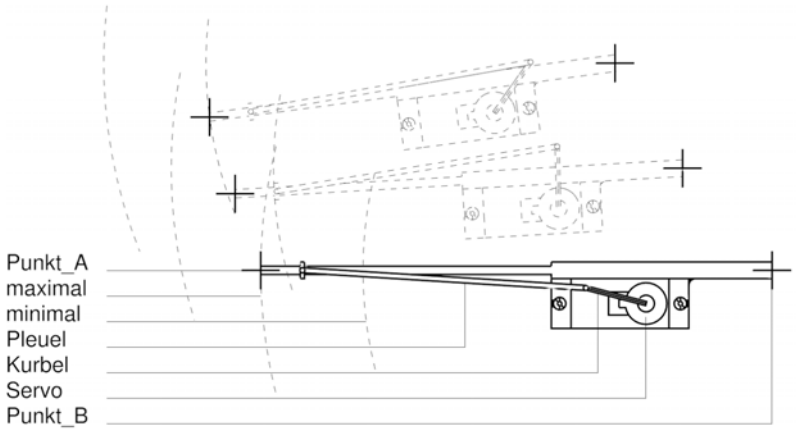
2.2.4 S E R V O

Die bereits diskutierte Effizienz von Klassen wird hier sichtbar. Die Beziehungen bei einem Antrieb mit Servo sind für Führung und Stange gleich den Objekten der vorausgegangen Antriebe. Diese werden durch das Laden der Klasse Beziehung und die Angabe der Namen beim Aufrufen der Funktionen automatisch erstellt. Ebenso verhält es sich mit den Abständen der Bezugspunkte.

Verweis auf:
Klasse Beziehung, Seite 36

Zusätzlich benötigt der Servo jedoch Beziehungen, um die Position und Ausrichtung des Pleuel und der Kurbel zu regeln. Es bietet sich die Beziehung IK, zum Lösen von inverser Kinematik an. Im nächsten Teil wird auch ein alternativer Lösungsansatz vorgeschlagen.

Verweis auf:
IK von Servo, Seite 40



Zchng.2.2.4-1 Kinematik gesamt

Kinematik Servo
Scriptsprache: Python
DCC: Blender

```python
import bpy

Maximalabstand = 2
Minimalabstand = 1.5

# class Beziehung

# Abhängigkeit
Fuehrung = Beziehung("Fuehrung", "Punkt_B", "Punkt_A")
Fuehrung.Abhaengigkeit()
Stange = Beziehung("Stange", "Punkt_A", "Punkt_B")
Stange.Abhaengigkeit()

# Abstand begrenzen
Fuehrung.Abstand()

# Knochen verbinden
bpy.ops.object.posemode_toggle()
obj = bpy.data.objects
Knochen = bpy.data.objects['Knochen'].pose.bones['Kurbel'].constraints
Knochen('COPY_LOCATION')
Knochen['Copy Location'].target = obj['Servo']
Knochen['Copy Location'].subtarget = 'Anker'
Knochen = bpy.data.objects['Knochen'].pose.bones['Pleuel'].constraints
Knochen('IK')
Knochen['IK'].target = obj['Stange']
Knochen['IK'].subtarget = 'Anker'
bpy.ops.object.posemode_toggle()

# Kurbel und Pleuel mit Knochen verbinden
bpy.context.scene.objects.active = bpy.data.objects["Pleuel"]
bpy.ops.object.constraint_add(type='COPY_LOCATION')
bpy.context.object.constraints["Copy Location"].target = bpy.data.objects["Knochen"]
bpy.context.object.constraints["Copy Location"].subtarget = "Pleuel"
bpy.ops.object.constraint_add(type='TRACK_TO')
bpy.context.object.constraints["Track To"].target = bpy.data.objects["Stange"]
bpy.context.object.constraints["Track To"].subtarget = "Anker"
bpy.context.scene.objects.active = bpy.data.objects["Kurbel"]
bpy.ops.object.constraint_add(type='COPY_LOCATION')
bpy.context.object.constraints["Copy Location"].target = bpy.data.objects["Servo"]
bpy.context.object.constraints["Copy Location"].subtarget = "Anker"
bpy.ops.object.constraint_add(type='TRACK_TO')
bpy.context.object.constraints["Track To"].target = bpy.data.objects["Knochen"]
bpy.context.object.constraints["Track To"].subtarget = "Pleuel"
```

Scp.2.2.4-1 Einzelelement

$$Position = Position_B$$
$$Ausrichtung = Position_A$$

Frl.2.2.4-1 Kinematik Führung

$$Position = Position_A$$
$$Ausrichtung = Position_B$$

Frl.2.2.4-1 Kinematik Stange

$$Position = Position_{Servo}$$
$$Ausrichtung = Position_{Pleuel}$$

Frl.2.2.4-1 Kinematik Kurbel

$$Position = Position_{Kurbel}$$
$$Ausrichtung = Position_A$$

Frl.2.2.4-1 Kinematik Pleuel

$$\overline{AB} \leq Maximalabstand$$
$$\overline{AB} \geq Minimalabstand$$

Frl.2.2.4-1 Kinematik Grenzen

2.2.5 MODELLBAU

In diesem Abschnitt wird eine gängige Methode beschrieben, wie die zur Steuerung erforderlichen Daten aus dem virtuellen Modell abgelesen, verarbeitet und an die Hardware gesendet werden können. Genauere Informationen sind beispielsweise im Github von Alvaro Ferrán Cifuentes zu finden.

Alvaro Ferrán Cifuentes, „Blender Controller“, 2016

Die Schnittstelle zwischen Computer und Motor ist der auf dem Arduino-Chip basierende Microcontroller Dagu Redbackspider. Dabei handelt es sich um einen speziellen Controller, der für die Steuerung von Servos konzipiert wurde. Die verwendeten Motoren haben ein Gewicht von 9 g und ein Drehmoment von 13,7 Ncm bei 4,8 V sowie 15,7 Ncm bei 6,0 V.

AMAX, „Servo AMAX G09AP Micro“, 2016

Damit der Controller einen Servo steuern kann, benötigt er einen Drehwinkel. Dieser wird wie folgt errechnet: Erst muss der Abstand der beiden Bezugspunkte gemäß des Satzes des Pythagoras ermittelt werden, wie rechts beschrieben. Um den Drehwinkel Alpha zu ermitteln, wurde mit Hilfe von Maxima der Kosinussatz auf Alpha aufgelöst. Mit dieser abgeleiteten Formel kann der Winkel für die Servos in Echtzeit errechnet werden.

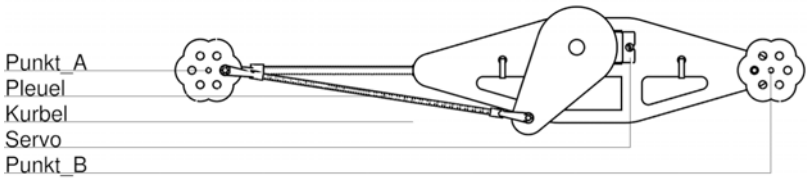
Verweis auf:
Maxima, Seite 275

Lothar Papula, „Mathematische Formelsammlung für Ingenieure und Naturwissenschaftler“, 2009

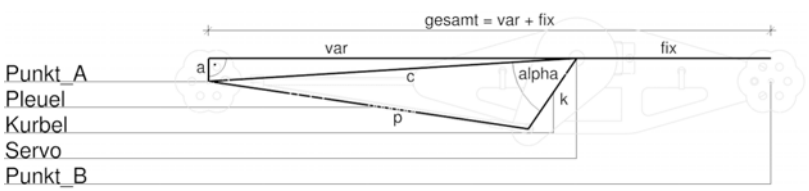
Jeder Servo ist an einem speziellen Pin am Controller angeschlossen. Der errechnete Winkel des Servos wird gemeinsam mit dem ihm entsprechenden Pin als Datensatz an die serielle Schnittstelle gesendet. Die Verarbeitung wird auf den folgenden Seiten beschrieben.

Wichtige Anmerkung: Der rechts gezeigte Script funktioniert nur, wenn er mittels Operator in Blender eingebunden wird. Dieser ist im Bereich Templates des Scripting-Editors von Blender zu finden.

Folgeseite:
Abb.2.2.5-1 Rendering



Zchng.2.2.5-1 Einzelteile



Zchng.2.2.5-2 Grundlage für Formel

Steuerung von Seiten der Software
Scriptsprache: Python
DCC: Blender

```python
import bpy
import math
from math import degrees
import serial

# Informationen zur Schnittstelle
Port = "/dev/tty.usbserial-A601FZBJ"
ser = serial.Serial(Port,9600,timeout=1)

# Operator aus den Vorlagen von Blender
class ModalTimerOperator(bpy.types.Operator):
    """Operator which runs its self from a timer"""
    bl_idname = "wm.modal_timer_operator"
    bl_label = "Modal Timer Operator"
    _timer = None
    def modal(self, context, event):
        if event.type == 'ESC':
            return self.cancel(context)

        if event.type == 'TIMER':

            # Erhält die Rotation der Servos
            Servo_11 = degrees(bpy.data.objects['Servo_11'].rotation_euler.x)
            print("Winkel", Servo_11)
            Servo_12 = degrees(bpy.data.objects['Servo_12'].rotation_euler.x)
            print("Winkel", Servo_12)
            Servo_13 = degrees(bpy.data.objects['Servo_13'].rotation_euler.x)
            print("Winkel", Servo_13)

            # Umwandeln in Byte und and die serielle Schnittstelle senden
            data = bytes( [11, int(Servo_11), 12, int(Servo_12), 13, int(Servo_13)] )

            ser.write(data)

        return {'PASS_THROUGH'}

    def execute(self, context):
        self._timer = context.window_manager.event_timer_add(0.1, context.window)
        context.window_manager.modal_handler_add(self)
        return {'RUNNING_MODAL'}
    def cancel(self, context):
        context.window_manager.event_timer_remove(self._timer)
        return {'CANCELLED'}

def register():
    bpy.utils.register_class(ModalTimerOperator)
def unregister():
    bpy.utils.unregister_class(ModalTimerOperator)
if __name__ == "__main__":
    register()
    bpy.ops.wm.modal_timer_operator()
```

Scp.2.2.5-1 Steuerung im DCC-Programm

$$gesamt = \sqrt{(x_B - x_A)^2 + (y_B - y_A)^2 + (z_B - z_A)^2}$$

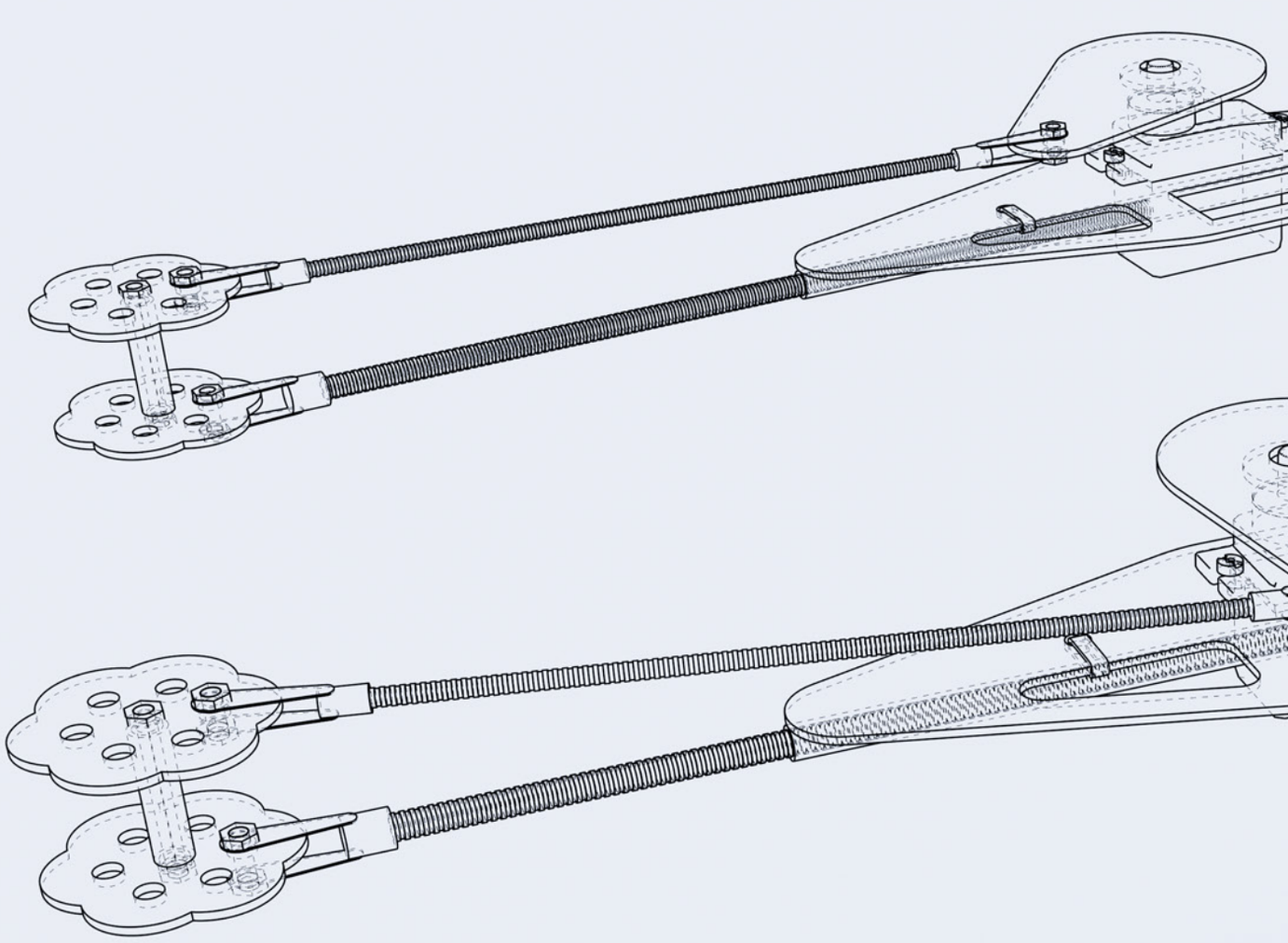
Frl.2.2.5-1 Abstand mittels Pythagoras

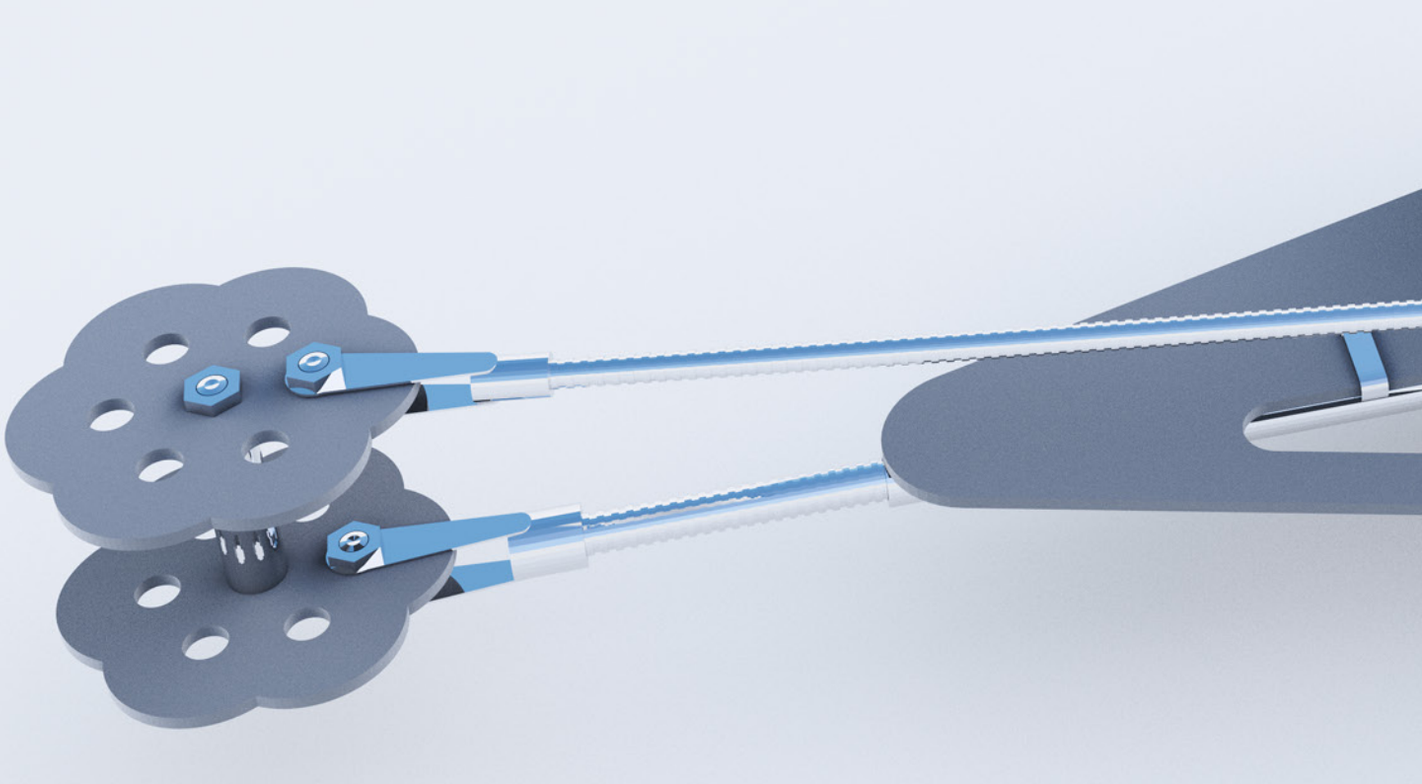
$$a^2 = b^2 + c^2 - 2bc \cdot \cos(\alpha)$$

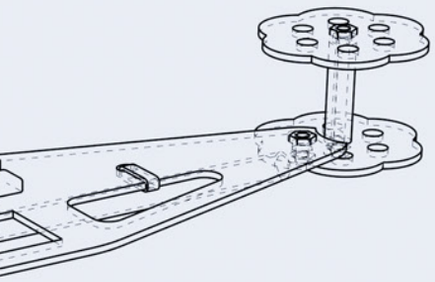
Frl.2.2.5-1 Grundlage mittels Kosinussatz

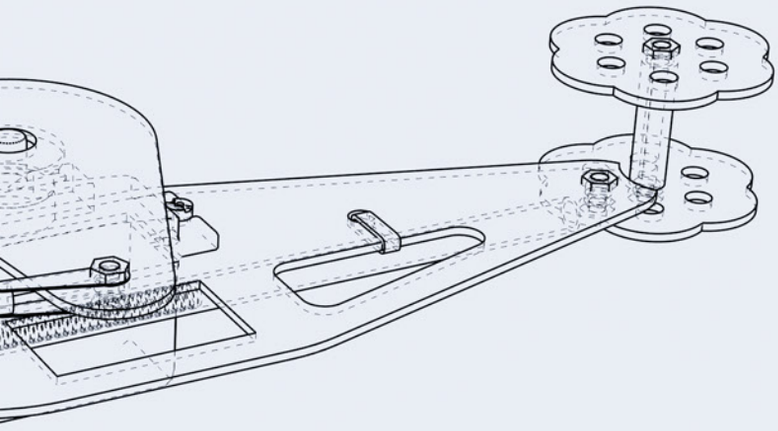
$$\alpha = \arccos\left(\frac{c^2 + b^2 - c^2}{2bc}\right)$$

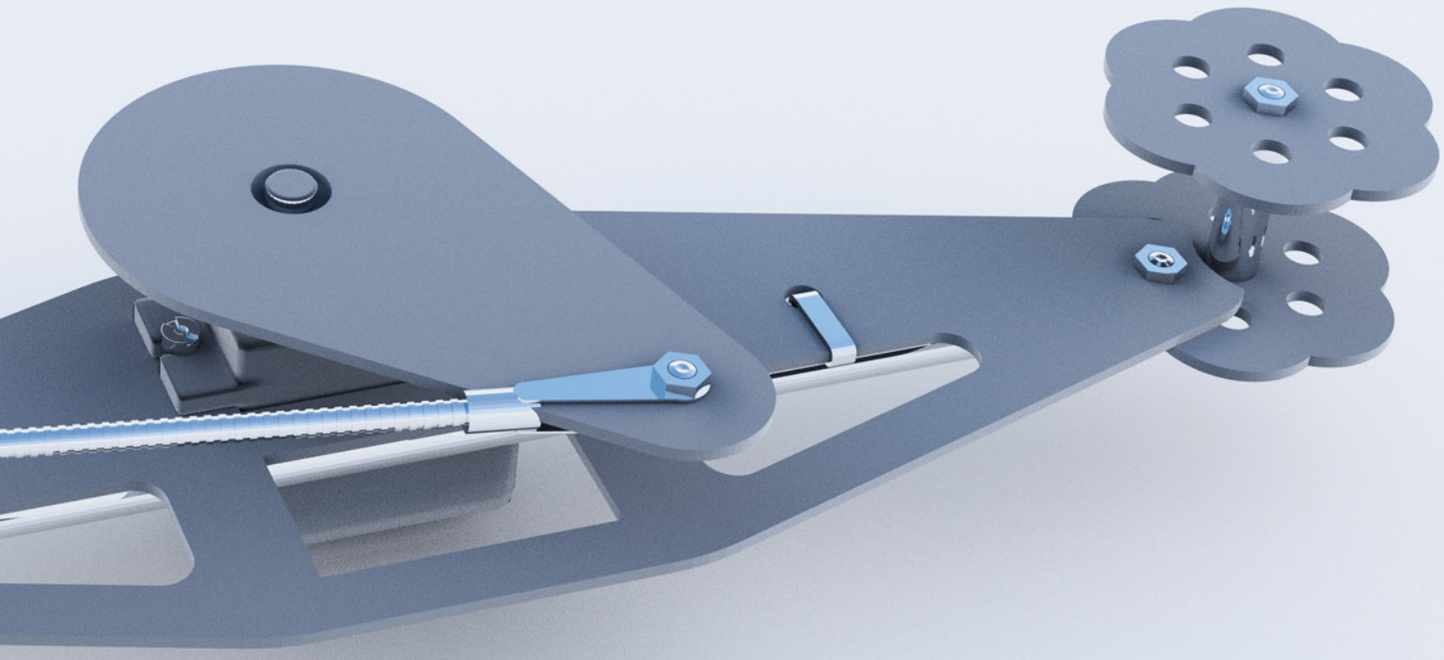
Frl.2.2.5-1 Winkel mittels Maxima











Mittels des folgenden Scripts kann der Controller so programmiert werden, dass er die im vorausgegangen Script gesendeten Daten empfangen und die Servos steuern kann. Dazu wird der Script mittels der Software von Arduino permanent auf dem Controller gespeichert.

Verweis auf:
Arduino, Seite 275

Im ersten Abschnitt wird mitgeteilt, welche Pins verwendet werden. Der zweite Abschnitt des Scripts lässt den Controller schließlich permanent die Daten des vorausgegangen Scripts im Abschnitt Software empfangen. Zur Erinnerung: Der erste Teil des Datensatzes ist der Pin, der zweite der Drehwinkel. Beginnt also ein Datensatz mit 13 und setzt mit 90 fort, wird an Pin 13 der Winkel 90 gesendet.

Verweis auf:
Datensatz für Controller, Seite 40

Servos drehen sich zwar grundsätzlich so lange, bis ein bestimmter Winkel erreicht wird, bleiben aber bei einem zu hohen Widerstand stehen. Lässt der Widerstand nach, drehen Sie sich weiter, bis der Winkel erreicht ist. Festzuhalten ist, dass es kein unmittelbares Feedback zur aktuellen Position im analogen Modell gibt.

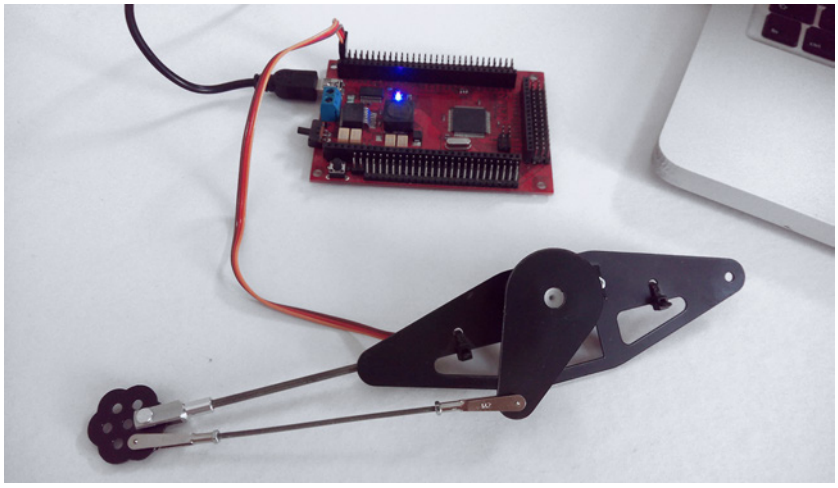
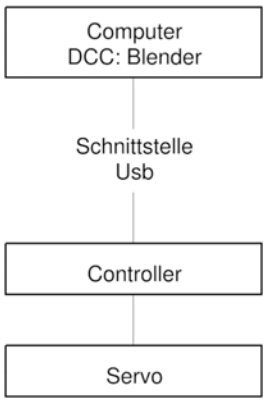


Abb.2.2.5-2 Hardware

Programmierung des Microcontrollers
Scriptsprache: C++
Controller: Dagu Redbackspider

```
\begin{lstlisting}
#include <Servo.h>

// Servos festlegen
Servo A;
Servo B;
Servo C;

void setup()
{
  A.attach(11);
  B.attach(12);
  C.attach(13);
   Serial.begin(9600);

  A.write(0);
  B.write(0);
}

# Variablen festlegen

int num;
int angle;

void loop()
{
  while(Serial.available() == 0);
  num = Serial.read();

  while(Serial.available() == 0);
  angle = Serial.read();

  if (num == 11)
    A.write(angle);
  if (num == 12)
    B.write(angle);
  if (num == 13)
    C.write(angle);
}
```

Scp.2.2.5-2 Programmierung des Controllers



Diagr.2.2.5-1 Ablauf

2.2.6 ARCHITEKTUR

Der Einsatz im architektonischen Maßstab erfordert stärkere Motoren. Häufig ist es ausreichend, wenn sich diese langsamer bewegen. Die Motoren, die letztendlich auch im finalen Prototyp verbaut wurden, haben ein Planetengetriebe und zudem eine Übersetzung durch die Spindel von 71:1. Es ergibt sich eine Geschwindigkeit von 3 mm/s und eine maximale Kraft von 2.200 N bei einem Gewicht von 1,1 kg. Benötigt wird eine Stromstärke von maximal 2,4 A und eine Spannung von 24 V.

Datenblatt von ELRA Antriebstechnik Vertriebs GmbH, „Con 35“, 2014

Anders als bei den kleinen Servos, können die Motoren daher ihren Strom nicht über den Controller beziehen. Benötigt werden spezielle Controller oder Eigen-Bau-Varianten mittels Relais. Diese erlauben das Öffnen und Schließen von Stromkreisen bis 240 V durch einen Controller basierend auf 5 V. Um den Motor nicht nur an und aus zu schalten, sondern auch die Richtung zu ändern, werden pro Motor zwei Relais benötigt. Robert Van Deest beschreibt mögliche Varianten auf seinem Blog.

Robert Van Deest, „Motor Control using Relays“, 2007

Die Steuerung von Linearaktuatoren funktioniert ähnlich wie bei Servos. Der Wert wird im digitalen Modell ausgelesen, jedoch von Seiten des Controllers etwas anders verarbeitet. Grund dafür ist, dass der gewählte Linearaktuator sich wesentlich langsamer bewegt als die Servos. Der Computer sendet daher ein Signal mit dem zu bewegenden Weg an den Controller und pausiert solange, bis dieser Wert erreicht ist. Dann sendet der Controller ein Signal zurück und informiert so den Computer. Diese Methode erscheint insofern sinnvoll, da Linearaktuatoren häufig auch maximale Einschaltdauern besitzen, welche dadurch berücksichtig werden können.

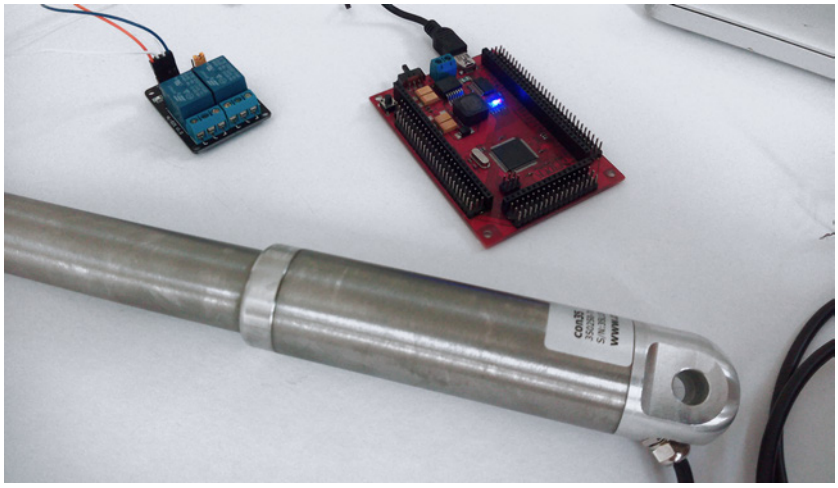
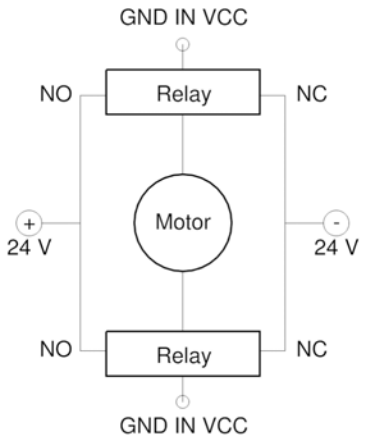


Abb.2.2.6-1 Hardware

Programmierung des Microcontrollers
Scriptsprache: C++
Controller: Dagu Redbackspider

```
\begin{lstlisting}
// Variablen für Pins festlegen

int Motor_1a = 47;
int Motor_1b = 46;

void setup()
{
  // Servos an Pins binden

  digitalWrite(M1_a, HIGH);
  digitalWrite(M1_b, HIGH);
  pinMode(M1_a, OUTPUT);
  pinMode(M1_b, OUTPUT);

  Serial.begin(9600);
}

int Motor;
int Weg;

void loop()
{
  // wenn Schnittstelle erreichbar Motor und Weg lesen
  while (Serial.available() == 0);
  Motor = Serial.read();
  Serial.println("Motor: ");
  Serial.println(Motor);

  while (Serial.available() == 0);
  Weg = Serial.read();

  Serial.println("Weg:");
  Serial.println(Weg);

  // Motor bewegen (Motor 11 bedeutet Motor 1 dreht rückwärts↩
      ↪ )
  if (Motor == 1){
    Serial.println("Motor fährt ein");
    digitalWrite(Motor_1a, LOW);
    delay(Weg/3*1000);
    digitalWrite(Motor_1a, HIGH);
    Serial.println("Motor hat Position erreicht");
    }
  if (Motor == 11){
    Serial.println("Motor fährt aus");
    digitalWrite(Motor_1b, LOW);
    delay(Weg/3*1000);
    digitalWrite(Motor_1b, HIGH);
    Serial.println("Motor hat Position erreicht");
    }

  // Variablen zurücksetzen
  Motor = 0;
  Weg = 0;
}
```

Scp.2.2.6-1 Programmierung des Controllers



Diagr.2.2.6-1 Schaltplan

Relay 1	aus	aus	an	an
Relay 2	aus	an	aus	an
Motor	aus	links	rechts	aus

Tab.2.2.6-1 Schaltung

2.3 TRAGWERK

Verweis auf:
Details, Seite 204

Entgegen dem Trend hybrider Bauteile in der Architektur, setzt diese Arbeit auf eine klare Differenzierung in einzelne Funktionen. Tragstruktur, Hülle und Innenleben werden als getrennte Systeme verstanden, die ausgetauscht werden können, um langfristig auf sich ändernde Anforderungen zu reagieren. Ihre Beziehung untereinander stellt eine besondere Fragestellung im Rahmen der Arbeit dar. Genauere Erörterungen der Entwurfs-Philosophie der bAm sind im Abschnitt Details zu finden.

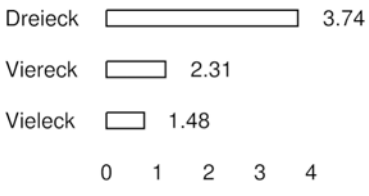
2.3.1 GITTERFORM

Die Computergraphik kennt drei grundlegende Arten, Gitternetze zu erzeugen: aus Dreiecken, Vierecken und Vielecken. Diese drei Arten wurden in der Studie rechts gegenübergestellt. Es wird davon ausgegangen, dass aus Gründen der Ökonomie und Effizienz alle VGT gleich lang sind. Die Kantenlängen der Raster betragen innerhalb dieser Studie jeweils 1,00 m Länge.

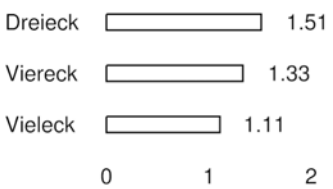
Klar erkennbar ist, dass wie allgemein anerkannt, das Vieleck die effizienteste Form ist, Flächen zu erzeugen. Sowohl das Verhältnis von Kantenlänge zu Quadratmeter, Stückzahl zu Knotenpunkt, als auch Stückzahl zu Kantenlänge ist geringer als bei den Alternativen. Das Viereck liegt im Mittelfeld und das Dreieck ist am wenigsten effizient. Zudem sind die Knotenpunkte bei Viereck und Dreieck komplexer. Das Vieleck hat drei, das Viereck schon vier und das Dreieck benötigt schließlich sechs Anschlüsse.

Das Dreieck bietet aber auch Vorteile gegenüber den Konkurrenten. Es ist immer planar, egal in welcher Position sich die einzelnen Punkte befinden und zugleich immer statisch bestimmt. Letzteres ist besonders bei einer Tragstruktur, die lediglich aus beweglichen Knotenpunkten besteht, von besonderer Bedeutung.

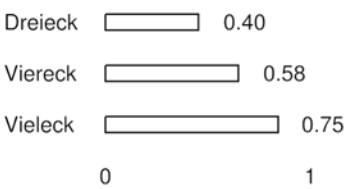
Verweis auf:
Knotenpunkte, Seite 110



Diagr.2.3.1-1 m/qm

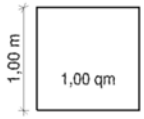


Diagr.2.3.1-2 Stück/qm

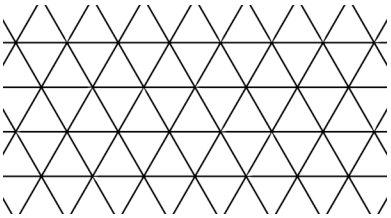


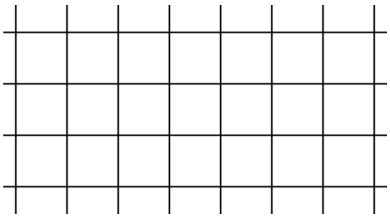
Diagr.2.3.1-3 Stück/m

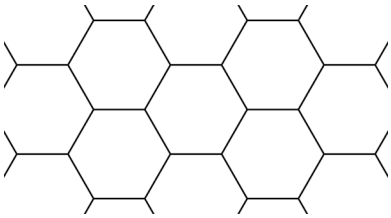
Studie zur Gitterform
Vergleich von Dreieck, Viereck, Vieleck
alle Varianten basieren auf gleicher Kantenlänge



Zchng.2.3.1-1 a,b,c Grundform



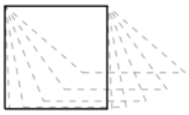


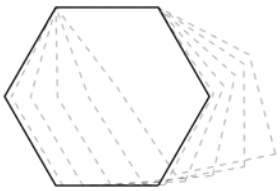


Zchng.2.3.1-2 a,b,c als Raster

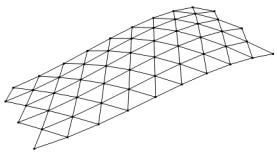
57 Knotenpunkte 141 Träger 82 Flächen	60 Knotenpunkte 104 Träger 45 Flächen	72 Knotenpunkte 96 Träger 25 Flächen
167 m Trägerlänge in Summe 37,69 qm Flächeninhalt in Summe	104 m Trägerlänge in Summe 45,00 qm Flächeninhalt in Summe	96 m Trägerlänge in Summe 64,95 qm Flächeninhalt in Summe

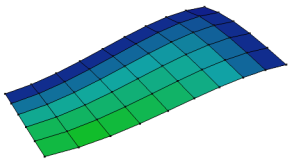
Tab.2.3.1-1 a,b,c Anzahl

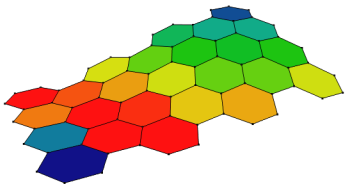




Zchng.2.3.1-3 a,b,c Bestimmtheit







Diagr.2.3.1-4 a,b,c Planarität

6 Verbindungen, also kompliziertere Knotenpunkte	4 Verbindungen, also einfachere Knotenpunkte	3 Verbindungen, also einfachere Knotenpunkte
ausgesteift stets planar Anschluss in Aggregation gerade	nicht ausgesteift nicht planar, zu approximieren Anschluss in Aggregation gerade	nicht ausgesteift nicht planar, zu approximieren Anschluss in Aggregation gerade

Tab.2.3.1-2 a,b,c Vergleich

Das Raster aus Dreiecken, wie auf der vorausgegangen Doppelseite gezeigt, wird mit dem Script rechts automatisch erstellt. Die grundlegende Methode, Gitternetze zu erzeugen, baut auf dem offiziellen Handbuch auf.

wiki.blender.org, „Geometry“, 2013

Die Ausgangsgeometrie wird aus Elementen mit gleicher Länge erzeugt. Daraus ist zu schließen: Alle Dreiecke sind gleichseitig. Die Höhe der Dreiecke ergibt sich aus der bekannten Formel zur Rechten.

Im Bereich Variablen werden grundlegende Parameter wie die Elementgröße, Breite und Tiefe der zu erzeugenden Struktur festgelegt. Die Variable Höhe wird ebenso definiert, ist aber erst in weiterer Folge von Bedeutung. Anhand der gegebenen Parameter wird ein Viereck-Raster aus Punkten erzeugt. Schließlich werden anhand dieser Punkte für jedes Viereck zwei Dreiecke erzeugt.

Verweis auf:
Tragwerkshöhe, Seite 56

Der letzte Abschnitt benennt die erzeugte Geometrie, verpackt sie in ein gleichnamiges Objekt, bindet dieses in die Szene ein und macht es schließlich durch ein Update der Szene sichtbar.

Entwicklung des Rasters
Scriptsprache: Python
DCC: Blender

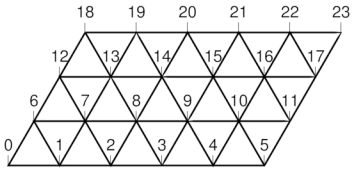
```python
import bpy
import math

# Variablen
Raster_X = 6
Raster_Y = 9
Elementgroesse = 2
Hoehe = 1

# Koordinaten für Punkte erzeugen
Zaehler_X = 0
Zaehler_Y = 0
Bewege_X = Elementgroesse /2
Bewege_Y = math.sqrt(3) /2 *Elementgroesse
Punkte = []

# Flächen definieren
for i in range(Raster_X):
    for i in range(Raster_Y):
        Punkte.extend([(Zaehler_Y * Elementgroesse + Bewege_X, Bewege_Y, 0)])
        Zaehler_Y += 1
    Bewege_X += Elementgroesse /2
    Bewege_Y += math.sqrt(3) /2 * Elementgroesse
    Zaehler_Y = 0
    Zaehler_X += 1

# create triangles from vertices
Index = 0
Flaechen = []
Zaehler_X = 0
Zaehler_Y = 0

# Dreiecke erzeugen
for i in range(Raster_X -1):
    for i in range(Raster_Y -1):
        Flaechen.extend ([(Index, Index +1, Index + Raster_Y)])
        Flaechen.extend ([(Index +1, Index + Raster_Y +1, Index + Raster_Y)])
        Index += 1
    Index += 1

# Gitternetz erzeugen, Objekt erzeugen, mit Objekte und Szene verlinken und updaten
Gitternetz = bpy.data.meshes.new("Kontrollflaeche")
Objekt = bpy.data.objects.new("Kontrollflaeche", Gitternetz)
Objekt.location = 0,0,0
bpy.context.scene.objects.link(Objekt)
Gitternetz.from_pydata(Punkte,[],Flaechen)
Gitternetz.update(calc_edges=True)
```

Scp.2.3.1-1 Grundriss

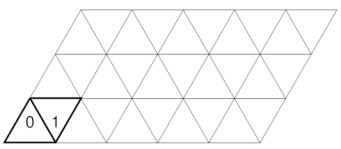
Zchng.2.3.1-4 Grundelement

$$h = \frac{\sqrt{3}}{2} \cdot Elementgroesse$$

Frl.2.3.1-1 Höhe Dreieck



Diagr.2.3.1-5 Punkte erzeugen



Diagr.2.3.1-6 Flächen erzeugen

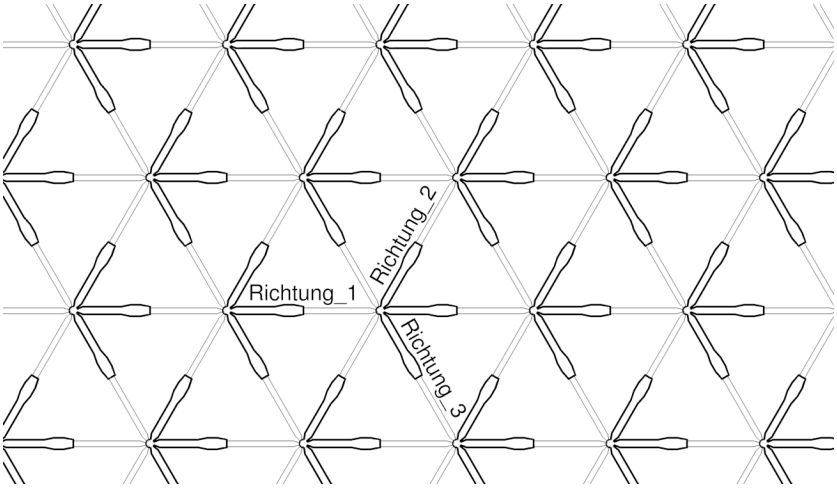
2.3.2 S T R U K T U R

Aus architektonischer Sicht stellt sich die Frage, welche Auswirkung die Ausrichtung eines einzelnen Motors auf die Erscheinung der Gesamtfläche hat. Gezeigt wird eine Methode, Flächen mit unterschiedlichen Richtungen zu erzeugen.

Die Studie gebündelt zeigt eine mögliche Ausrichtung der Motoren. Es entsteht eine Fläche, die klar in eine Richtung zu zeigen scheint. Lediglich das Ändern von Richtung 1 führt zu einer andern Struktur. Die Fläche wird neutraler, wie bei der zweiten Variante zu sehen. Abschließend zeigt die Studie hexagonal die Option, einzelne Elemente auszulassen. Das entstandene Muster wird transparenter. Verloren geht die statische Bestimmtheit, wie im Abschnitt Rasterform diskutiert.

Verweis auf:
Gitterform, Seite 48

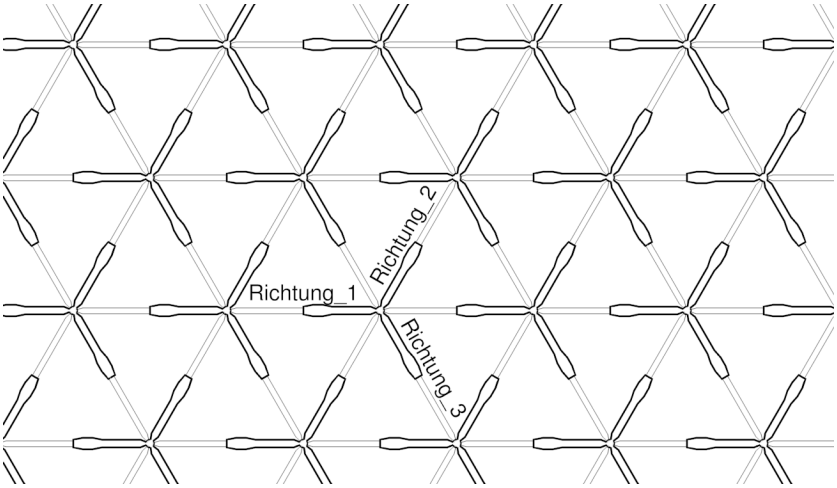
Studie zu Ausrichtung
gebündelt, gleichförmig, hexagonal
Ausrichten und Entfernen von Elementen



Diagr.2.3.2-1 gebündelt

Richtung_1 = n
Richtung_2 = n
Richtung_3 = g

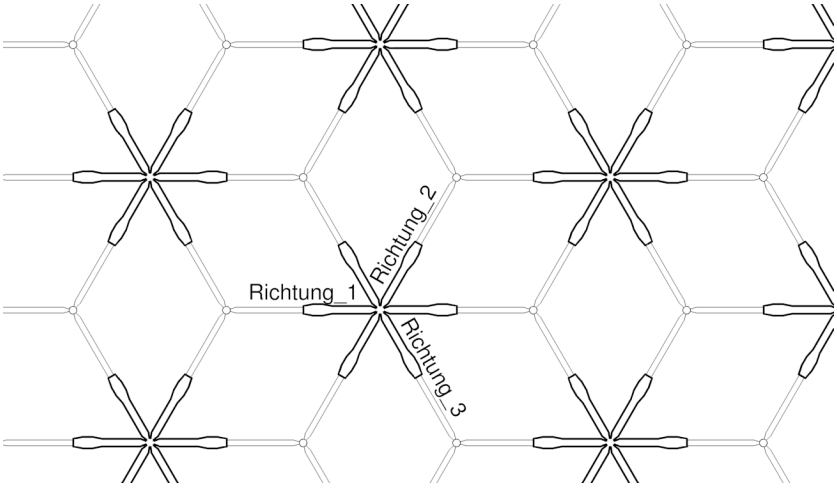
Tab.2.3.2-1 gebündelt



Diagr.2.3.2-2 gleichförmig

Richtung_1 = g
Richtung_2 = n
Richtung_3 = g

Tab.2.3.2-2 gleichförmig



Diagr.2.3.2-3 hexagonal

Richtung_1 = g, n, 0
Richtung_2 = g, n, 0
Richtung_3 = g, n, 0

Tab.2.3.2-3 hexagonal

Der Script zeigt die Umsetzung der vorab diskutierten Strukturen am Beispiel der Ausrichtung der Zylinder in die Richtung 1. Erweitert wurde der Ablauf zudem durch das Kopieren der Fläche in Z-Richtung und der Aussteifung der Flächen untereinander. Ergebnis ist schließlich ein Gitternetz, das mit VGT bestückt ist. Sämtliche Kinematik wurde dabei in Form von Constrains automatisch aufgesetzt. Ein Verändern der Kontrollfläche führt dazu, dass sämtliche Motoren sich der neuen Situation anpassen. Darauf wird im Abschnitt Steuerung noch genauer eingegangen.

Verweis auf:
Steuerung, Seite 60

Auf den Folgeseiten wird diskutiert, welche unterschiedlichen Möglichkeiten der Höhenentwicklung sich ergeben. Konsequenzen, sowie Potenzial werden dabei aufgezeigt.

Einzelelemente ausrichten
Scriptsprache: Python
DCC: Blender

```python
# Erstelle Vertex-Groups
Index = 0
for i in range(Raster_X * Raster_Y * 2):
    bpy.ops.object.mode_set(mode='OBJECT')
    bpy.data.objects["Kontrollflaeche"].vertex_groups.new(name=str(Index))
    bpy.context.tool_settings.mesh_select_mode = (True , False , False)
    bpy.ops.object.mode_set(mode="OBJECT")
    bpy.context.object.data.vertices[Index].select = True
    bpy.ops.object.mode_set(mode="EDIT")
    bpy.ops.object.vertex_group_assign()
    bpy.ops.mesh.select_all(action='TOGGLE')
    bpy.ops.object.editmode_toggle()
    Index += 1

# Zylinder Richtung_1 duplizieren
Index = 0
Zaehler_X = 0
Zaehler_Y = 0
for i in range(2):
    for i in range(Raster_X * Raster_Y - Raster_Y):
        bpy.data.objects["Zylinder"].select = True
        bpy.context.scene.objects.active = bpy.data.objects["Zylinder"]
        bpy.ops.object.duplicate_move_linked(OBJECT_OT_duplicate={"linked":True})

        bpy.context.object.name = "Zylinder Richtung_1 " +str(Index)
        bpy.ops.object.constraint_add(type='COPY_LOCATION')
        bpy.context.object.constraints["Copy Location"].target = bpy.data.objects["Kontrollflaeche"]
        bpy.context.object.constraints["Copy Location"].subtarget = str(Index)

        bpy.ops.object.constraint_add(type='TRACK_TO')
        bpy.context.object.constraints["Track To"].target = bpy.data.objects["Kontrollflaeche"]
        bpy.context.object.constraints["Track To"].subtarget = str(Index + Raster_Y)
        bpy.context.object.constraints["Track To"].track_axis = 'TRACK_Z'
        bpy.context.object.constraints["Track To"].up_axis = 'UP_Y'

        bpy.ops.object.select_all(action='TOGGLE')

        Index += 1
    Index = Index + Raster_Y
```

Scp.2.3.2-1 Struktur

2.3.3 TRAGWERKSHÖHE

Papier lässt sich leicht in alle Richtungen verformen. Ist ein Blatt jedoch in eine Richtung gebogen, erreicht es in Gegenrichtung Stabilität. Wölben Sie ein Blatt von links nach rechts auf und versuchen Sie es nun gleichzeitig von oben nach unten durchzubiegen. Das Papier wirkt der Kraft entgegen. Drehen Sie den Vorgang nun um. Wölben Sie das Blatt leicht von oben nach unten und versuchen Sie es gleichzeitig von rechts nach links durchzubiegen. Das Blatt ist nun in Gegenrichtung stabil. Eine Verformung in beide Richtungen, sprich Doppelkrümmung, würde das Papier zum Faltenwurf oder Reißen bringen. Wäre das jedoch möglich, würde das Blatt auch Stabilität in beide Richtungen erreichen. Wir sprechen vom Prinzip der Schale.

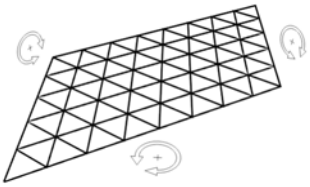
Die gezeigte Gegenüberstellung diskutiert die Verformbarkeit der im vorausgegangen Script erstellten Gitterschalen. Angenommen ist vorerst, dass sämtliche Knoten der ebenen Gitterstruktur gelenkig sind. Zudem ist angenommen, dass die Gitterstruktur nur aus einer Ebene besteht. Sämtliche durchgehenden Linien bilden dabei eine Art Scharnier, das die Fläche in alle Richtungen verformbar macht. Wird die Fläche in eine Richtung aufgebogen, wird sie in die beiden anderen Richtungen stabil. Wird die Fläche doppelt gekrümmt, wird auch die Verformung in die dritte Richtung unterbunden.

Als weitere Strategie kann angedacht werden, die Fläche zu verdoppeln und mit gelenkigen Knoten zu verbinden. Die Fläche bleibt dabei weiterhin drehbar, wie im ersten Beispiel. Zugleich ist die Fläche im planaren Zustand, analog zum ersten Beispiel, in alle Richtungen verschiebbar. Einfachgekrümmt wird sie in Gegenrichtungen stabil. Doppeltgekrümmt wird sie in alle Richtungen stabil.

Stabil in allen Zuständen ist lediglich eine Konstruktion, die zugleich diagonale Aussteifungen aus VGT besitzt. Mögliche Strategie kann auch eine biegesteife Verbindung der oberen und unteren Fläche sein.

Festzuhalten ist also, dass die drei gegenübergestellten Modelle eine unterschiedliche Leistungsstärke in Bezug auf das Spektrum möglicher Geometrien haben. Diese Leistungsstärke ist selbstverständlich einer erhöhten Komplexität gegenüberzustellen, wie auf den Folgeseiten gezeigt wird.

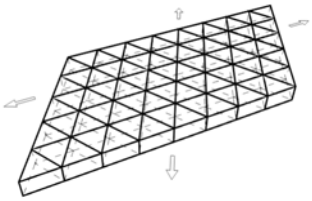
Statische Bestimmtheit
Vergleich von Krümmungen
einfach, doppelt, ausgesteift



Diagr.2.3.3-1 a,b,c einfach

planar	einfach gekrümmt	doppelt gekrümmt
frei beweglich in allen Richtungen	beweglich in einer Richtung	ausgesteift in alle Richtungen
Ecken sind nach oben verschiebbar	Ecken sind nach oben verschiebbar	Ecken sind nach oben verschiebbar

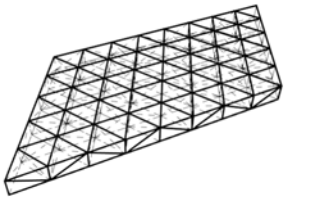
Tab.2.3.3-1 a,b,c einfach



Diagr.2.3.3-2 a,b,c doppelt

planar	einfach gekrümmt	doppelt gekrümmt
frei verschiebbar Ausnahme: steifer Knoten	verschiebbar in einer Richtung	ausgesteift in alle Richtungen
Ecken bleiben verschiebbar	Ecken bleiben verschiebbar	Ecken bleiben verschiebbar

Tab.2.3.3-2 a,b,c doppelt

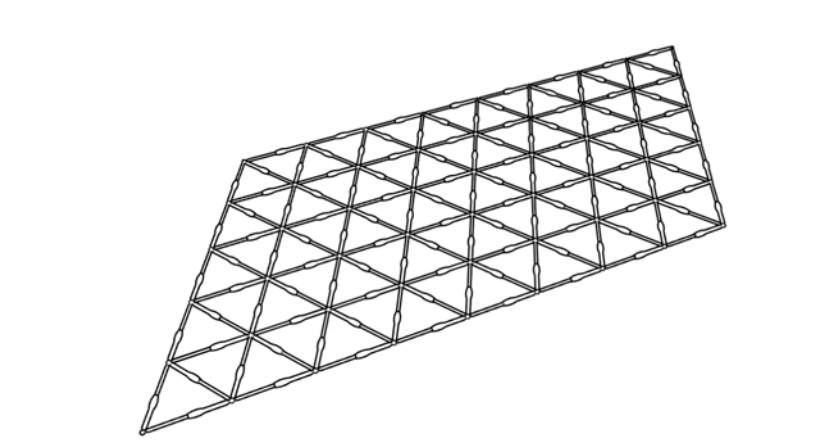


Diagr.2.3.3-3 a,b,c ausgekreuzt

planar	einfach gekrümmt	doppelt gekrümmt
ausgesteift in alle Richtungen	ausgesteift in alle Richtungen	ausgesteift in alle Richtungen
Ecken stets ausgesteift	Ecken stets ausgesteift	Ecken stets ausgesteift

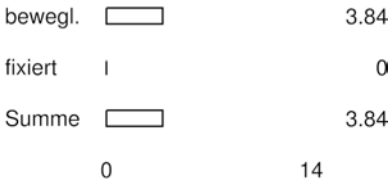
Tab.2.3.3-3 a,b,c ausgekreuzt

mögliche Tragwerke
Einfach, doppelt und ausgekreuzt
Fixiert und variabel

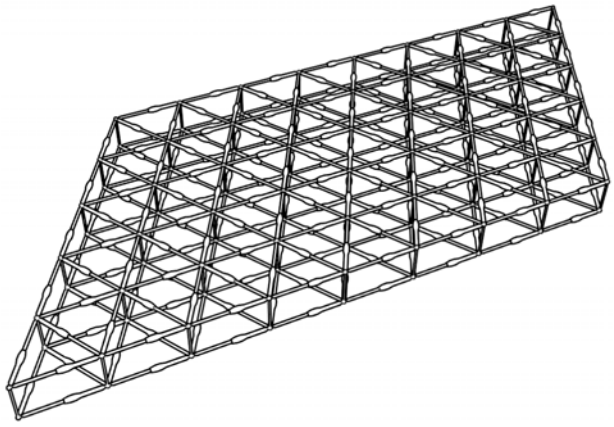


Zchng.2.3.3-1 einzeln

horizontale Elemente	133 Stück
vertikale Elemente	0 Stück
ausgesteifte Elemente	0 Stück
bewegliche Elemente	133 Stück
fixierte Elemente	0 Stück
Summe	133 Stück



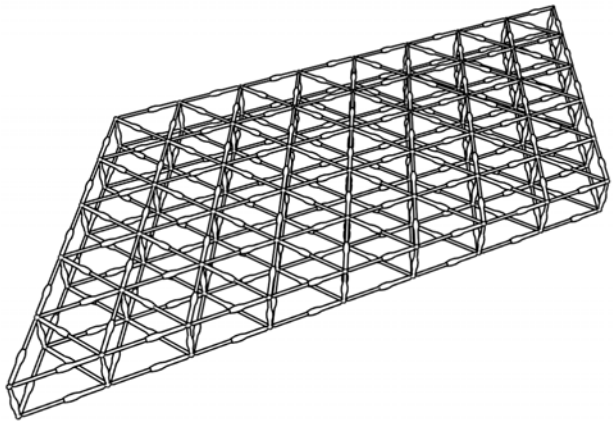
Diagr.2.3.3-4 einzeln



Zchng.2.3.3-2 doppelt fixiert

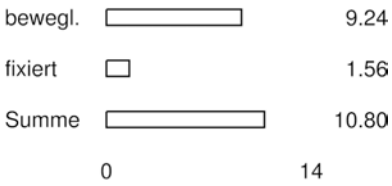
horizontale Elemente	266 Stück
vertikale Elemente	54 Stück
ausgesteifte Elemente	0 Stück
bewegliche Elemente	266 Stück
fixierte Elemente	54 Stück
Summe	320 Stück

Diagr.2.3.3-5 doppelt fixiert



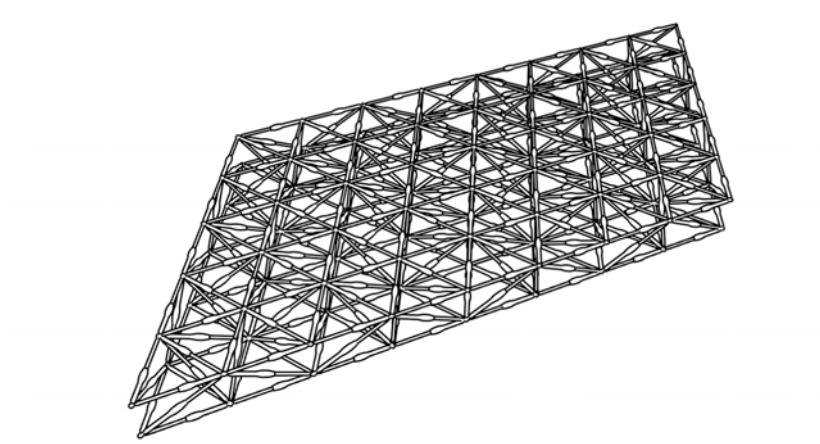
Zchng.2.3.3-3 doppelt variabel

horizontale Elemente	266 Stück
vertikale Elemente	54 Stück
ausgesteifte Elemente	0 Stück
bewegliche Elemente	320 Stück
fixierte Elemente	54 Stück
Summe	320 Stück



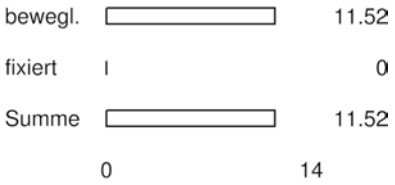
Diagr.2.3.3-6 doppelt variabel

mögliche Tragwerke
einfach, doppelt und ausgekreuzt
fixiert und variabel

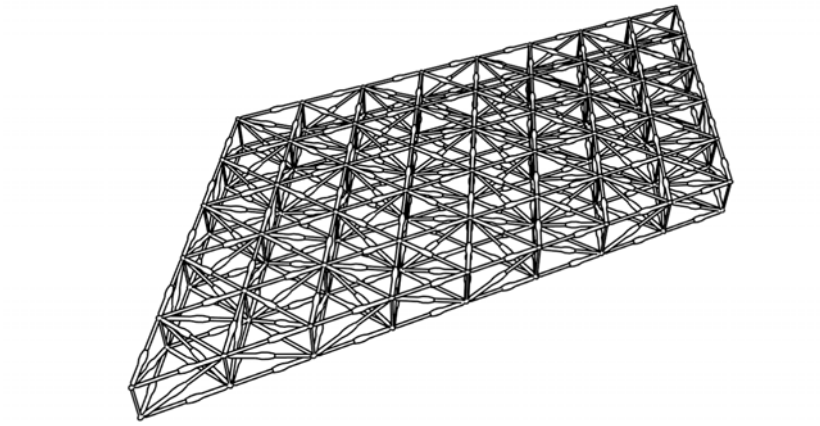


Zchng.2.3.3-4 doppelt ausgesteift

horizontale Elemente	266 Stück
vertikale Elemente	0 Stück
ausgesteifte Elemente	133 Stück
bewegliche Elemente	399 Stück
fixierte Elemente	0 Stück
Summe	399 Stück

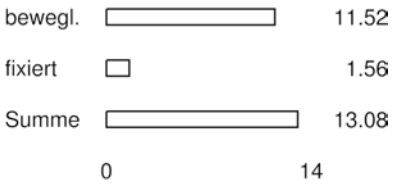


Diagr.2.3.3-7 doppelt ausgesteift

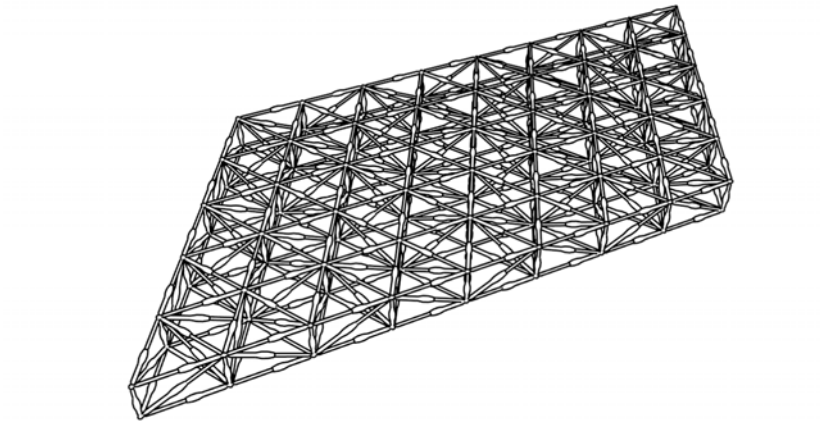


Zchng.2.3.3-5 doppelt ausgesteift fixiert

horizontale Elemente	266 Stück
vertikale Elemente	54 Stück
ausgesteifte Elemente	133 Stück
bewegliche Elemente	399 Stück
fixierte Elemente	54 Stück
Summe	453 Stück



Diagr.2.3.3-8 doppelt ausgesteift fixiert



Zchng.2.3.3-6 doppelt ausgesteift variabel

horizontale Elemente	266 Stück
vertikale Elemente	54 Stück
ausgesteifte Elemente	133 Stück
bewegliche Elemente	453 Stück
fixierte Elemente	0 Stück
Summe	453 Stück

Diagr.2.3.3-9 doppelt ausgesteift variabel

2.3.4 STEUERUNG

Möglichkeiten zur Veränderung der Fläche entstehen durch die Anwendungen von sämtlichen Werkzeuge zur Bearbeitung von Gitternetzen, die nicht die Topologie des Netzes oder den Index der Punkte des Netzes ändern. Werkzeuge wie Proportional-Editing eignen sich. Die Veränderung eines Punktes bewirkt dabei, dass alle angrenzenden Nachbarn folgen. Im gezeigten Beispiel wird im mittleren Bild ein Punkt nach oben verschoben. Der Radius des weißen Kreises gibt dabei den Einflussfaktor des Werkzeuges an.

Steuerung
Variation des Gitternetzes
Proportional-Editing

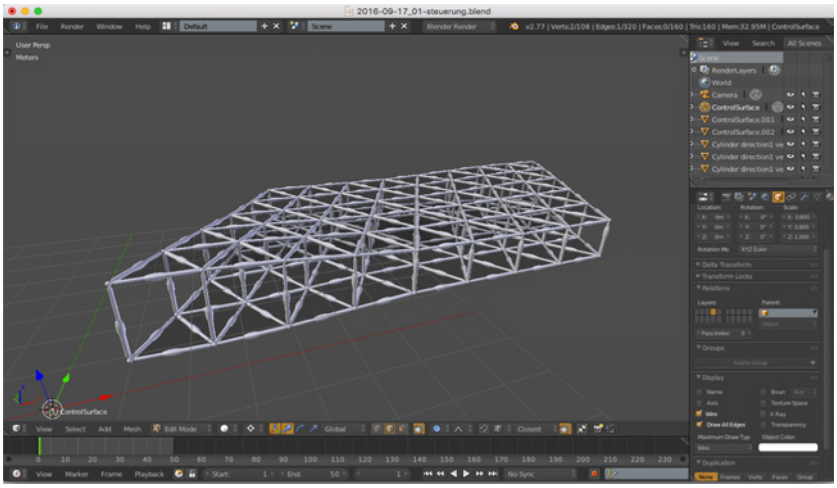


Abb.2.3.4-1 Zustand vor Transformation

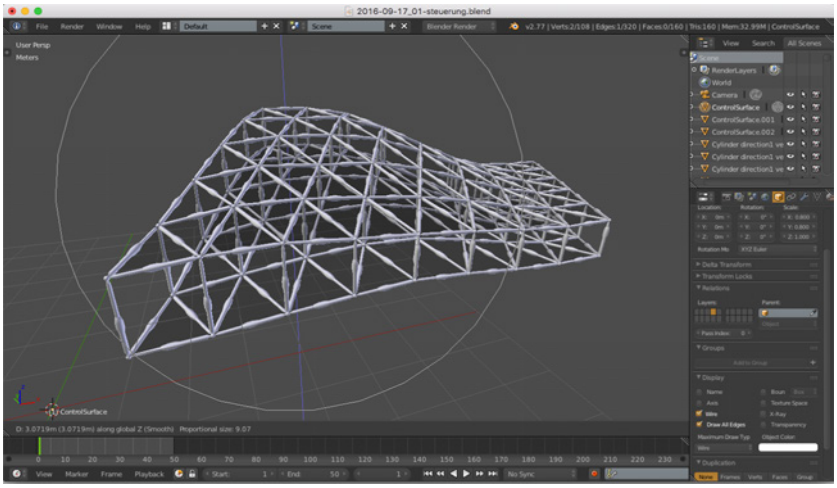


Abb.2.3.4-2 Einflussfaktor mit weißem Kreis

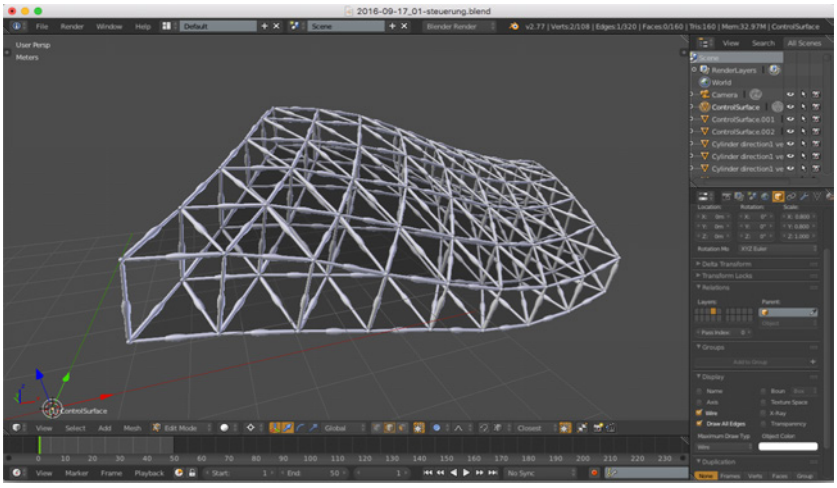


Abb.2.3.4-3 Zustand nach Transformation

2.3.5 BERECHNUNG

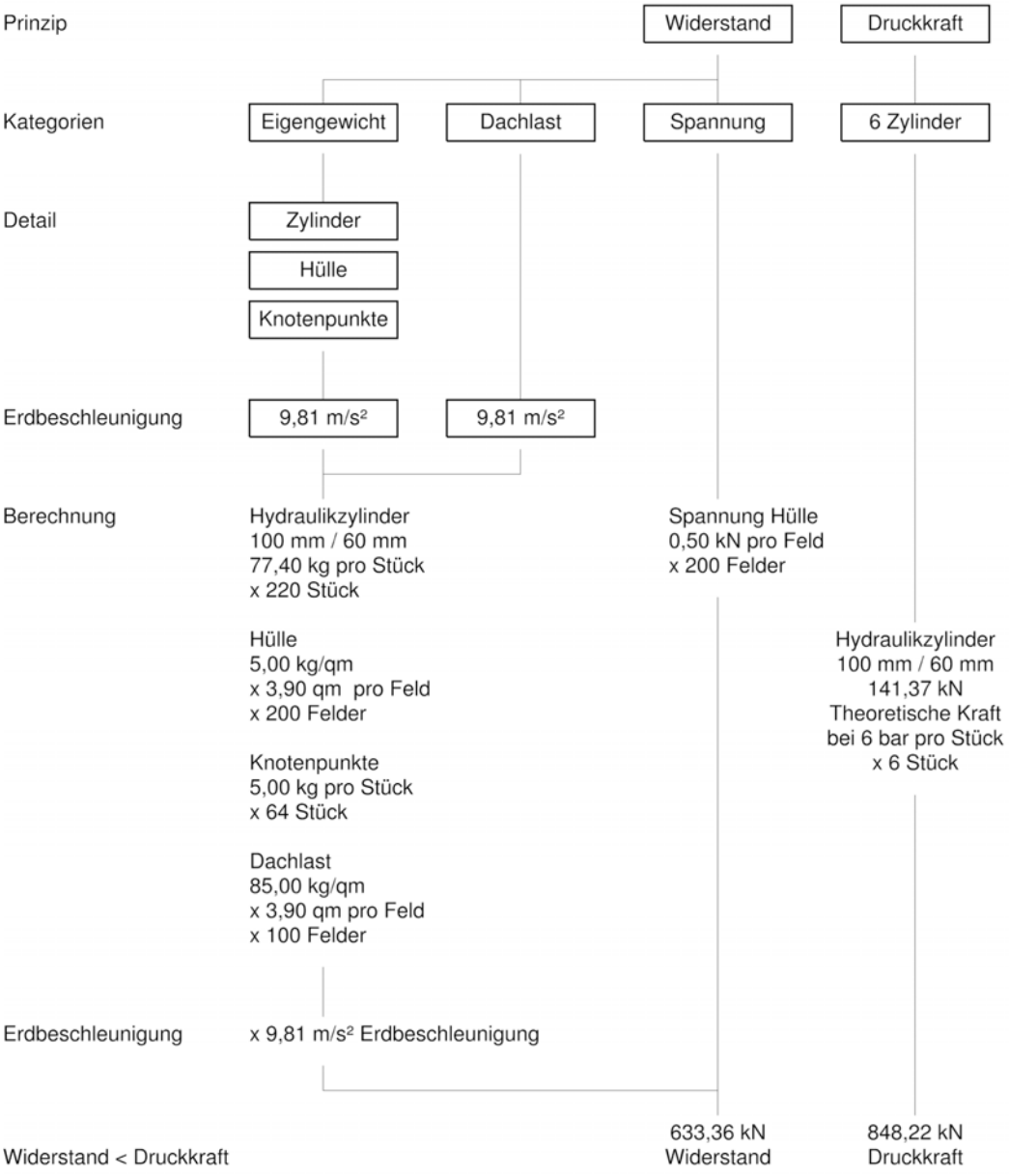
Dieses Kapitel beschäftigt sich mit einer ersten statischen Annäherung an die Konstruktion, sowie der materialeffizienten Dimensionierung. Da es sich um ein bewegliches Tragwerk handelt, existiert keine endgültige zu analysierende Momentaufnahme. Stattdessen soll die Konstruktion nach Fertigstellung so viele Formen wie möglich einnehmen können. Begonnen wird daher mit möglichen Worst-Case-Szenarien. Dieser Abschnitt beschäftigt sich dabei vorerst nur mit einer Gegenüberstellung der Kräfte. Welche Kräfte wirken ein? Welche Kräfte setzt die Konstruktion entgegen?

Es werden 200 kg Verkehrslast, sowie 85 kg Schneellast pro Quadratmeter angenommen. Auf Grund der niedrigen Höhe, der geringen Angriffsfläche und dem relativ hohen Eigengewicht werden keine Windkräfte berücksichtig. Als Grundlage für die Zylinder wurden Normzylinder mit einem Durchmesser von 100 mm gewählt. Diese verfügen über 141,37 kN Druckkraft.

Schema Hydraulik GmbH, „Normzylinder“, 2016

Grundsätzlich muss die Struktur an mindesten drei Positionen aufliegen. Als Auflagerpunkte definiert sind Knotenpunkte, zu denen sechs Motoren führen.

Statische Annäherung
Worst-Case-Szenario
Stand auf 6 Punkten

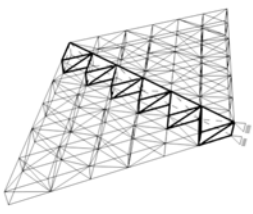


Diagr.2.3.5-1 Gegenüberstellung

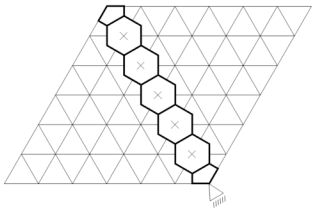
Dieser Zugang ist konkreter. Das Worst-Case-Szenario bezieht die maximal möglichen Spannweiten und Auskragungen der Konstruktion mit ein.

Je steiler ein Bogen wird, desto kürzer wird die zu überbrückende Spannweite. Zudem nehmen die Zylinder an den Rändern, welche die meiste Last tragen, einen günstigeren Winkel ein, um der Schwerkraft entgegen zu wirken. Als erstes Worst-Case-Szenario ist also anzunehmen: Sämtliche Zylinder sind voll ausgefahren und die Konstruktion bildet einen Bogen, der annähernd flach ist.

Analog dazu verhält es sich bei Auskragungen. Die Konstruktion kann dabei höchstens zur Hälfte auskragen, da sonst das Gleichgewicht nicht mehr gegeben ist. Wie im Szenario Spannweite gilt, dass eine Verformung in die Höhe zu einer geringeren Auskragung und zu einem günstigeren Winkel der Zylinder entgegen der Schwerkraft führt. Als weiteres Worst-Case-Szenario ist also anzunehmen: Sämtliche Zylinder sind voll ausgefahren und die Konstruktion bildet eine Auskragung, die annähernd flach ist.

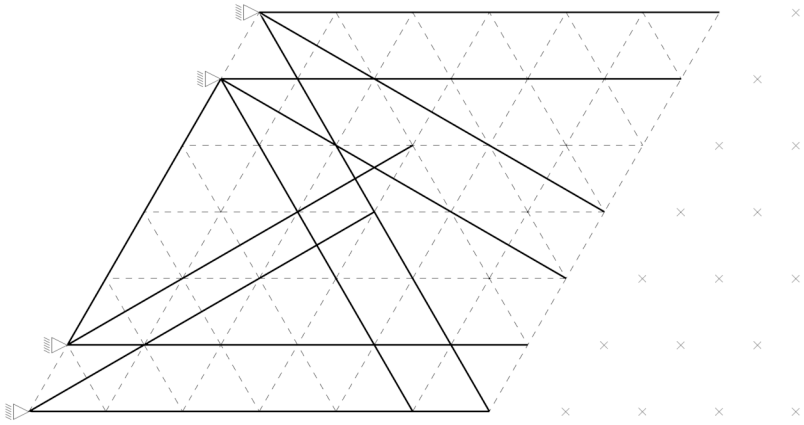


Diagr.2.3.5-2 Einzugsbereich Zylinder

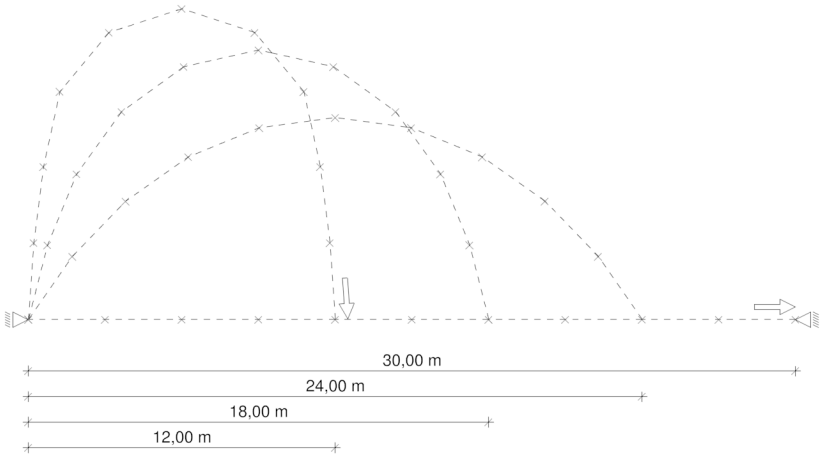


Diagr.2.3.5-3 Einzugsbereich Knoten

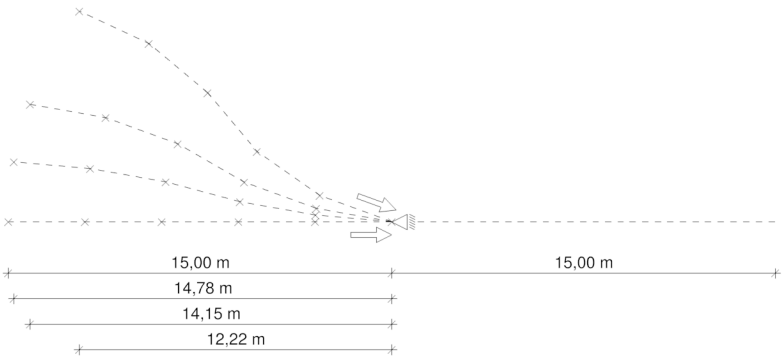
Statische Annäherung
Worst-Case-Szenario
in Form von maximaler Spannweite und Auskragung



Zchng.2.3.5-1 maximale Längen innerhalb der Konstruktion



Zchng.2.3.5-2 Worst-Case Spannweite

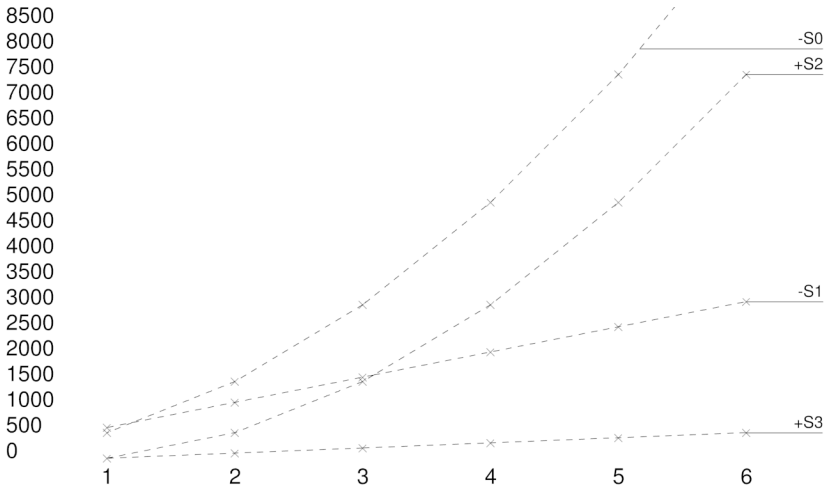


Zchng.2.3.5-3 Worst-Case Auskragung

Die vorausgegangene Abschnitt behandeln die Konstruktion lediglich als Linie im Schnitt. Dieser Abschnitt involviert die Höhe der Konstruktion. Gerechnet wurde mit dem Stabwerks-Programm Star2, das von der Universität Stuttgart zur Verfügung gestellt wird. Die Ergebnisse wurden anschließend auf den konkreten Fall in entsprechende Formeln gefasst.

Verweis auf:
Star2, Seite 275

Klar erkennbar ist, dass die Belastung der Stäbe S1 und S3 linear ansteigt, wenn die Struktur weiter überspannt. Die Belastung der Stäbe S0 und S2 hingegen steigt weitaus dramatischer an. Eine entsprechende siutationsbedinge Dimensionierung der Motoren oder der Stäbe ist daher erforderlich.



Diagr.2.3.5-4 Belastung

$$S1 = \frac{A(A+1)}{2(\frac{l}{h}G)}$$

Frl.2.3.5-1 S1

$$S2 = A\sqrt{\frac{g(h^2+l^2)}{-h}}$$

Frl.2.3.5-0 S2

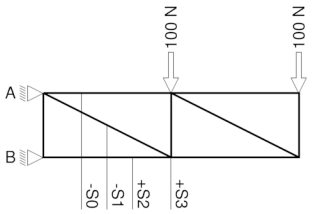
$$S3 = A\frac{(A+1)\cdot l\cdot g}{-2h}$$

Frl.2.3.5-0 S3

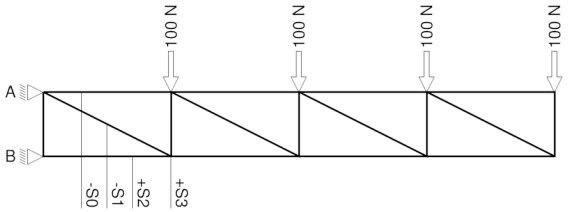
$$S4 = G\cdot A - G$$

Frl.2.3.5-0 S4

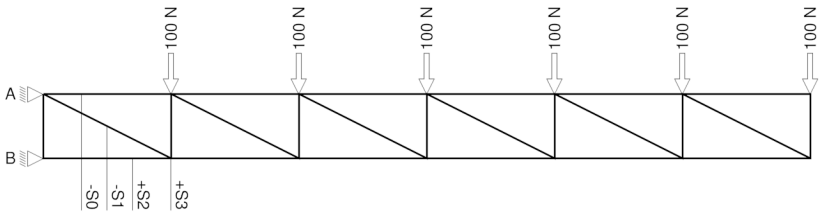
Statische Annäherung
Worst-Case-Szenario
Einbeziehen der Tragwerkshöhe



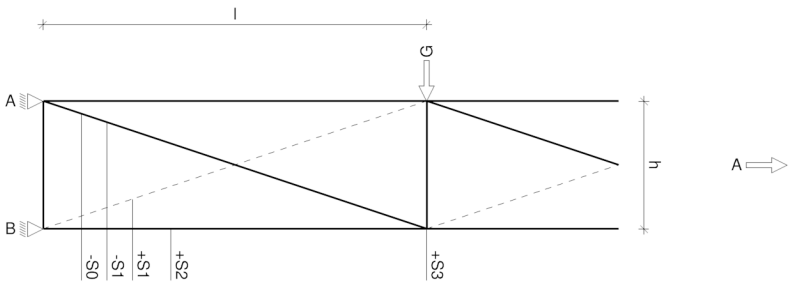
Zchng.2.3.5-4 Fachwerk mit 2 Feldern



Zchng.2.3.5-5 Fachwerk mit 4 Feldern



Zchng.2.3.5-6 Fachwerk mit 6 Feldern



Zchng.2.3.5-7 Grundlage zu den Formeln

Die Studie beschäftigt sich mit der Dimensionierung von Linearaktuatoren anhand von Hydraulikzylinder. Bei welchem Durchmesser bieten die Zylinder das beste Verhältnis von Gewicht zu Kraft?

Verwendet werden Zylinder mit einer Länge von 1,15 m, sowie einem maximalen Hub von 1,00 m. Die maximale Knicklänge ist bereits durch den angegebenen maximalen Hub und die Länge des Zylinders von Seiten des Herstellers einkalkuliert.

Parker Hannifin GmbH, „Lightraulics C-Series Cylinders“, 2016

Erkennbar ist, dass die Zylinder mit einem Durchmesser von 160 mm am besten abschneiden. Eine höhere Dimensionierung ist ungünstiger, da das Gewicht im höheren Maße ansteigt als die Kraft.

	80 / 36	100 / 36	125 / 56	160 / 63	180 / 90	200 / 125
105,00	-	-	-	-	-	-
70,00	-	-	-	-	-	-
52,50	-	-	-	+	-	-
42,00	-	-	+	+ -	+	-
35,00	-	+	+ -	+ -	+ -	+
30,00	+	+ -	+ -	+ -	+ -	+ -
26,25	+	+ -	+ -	+ -	+ -	+ -
23,33	+ -	+ -	+ -	+ -	+ -	+ -
	⇩	⇩	⇩	⇩	⇩	⇩
Kraft	142,11 kN	236,44 kN	337,92 kN	594,90 kN	685,46 kN	683,71 kN
Druck	191,00 kN	298,00 kN	466,00 kN	764,00 kN	967,00 kN	1194,00 kN
Zug	152,00 kN	260,00 kN	373,00 kN	646,00 kN	725,00 kN	727,00 kN

Diagr.2.3.5-5 Kraft/Widerstand Gegenüberstellung

Statische Annäherung
Worst-Case-Szenario
Annahme des Widerstandes der Tragstruktur

Zylinder	80 / 36	100 / 36	125 / 56	160 / 63	180 / 90	200 / 125
Zylinder						
Länge Zylinder Minimum	0,48 m	0,58 m	0,64 m	0,77 m	0,83 m	0,79 m
zusätzlich plus Hub	0,86 m	0,81 m	0,78 m	0,72 m	0,69 m	0,71 m
Hub	0,86 m	0,81 m	0,78 m	0,72 m	0,69 m	0,71 m
Länge Knoten	0,80 m	0,80 m	0,80 m	0,80 m	0,80 m	0,80 m
Gesamt	3,00 m	3,00 m	3,00 m	3,00 m	3,00 m	3,00 m
Abmaße und Fläche						
Weite	x 6	x 6	x 6	x 6	x 6	x 6
Tiefe	x12	x 12	x 12	x 12	x 12	x 12
Fläche eines Dreieck	3,90 qm	3,90 qm	3,90 qm	3,90 qm	3,90 qm	3,90 qm
Gesamt	561,18 qm	561,18 qm	561,18 qm	561,18 qm	561,18 qm	561,18 qm
Einzelgewicht						
Fläche unten	10,00 kg	10,00 kg	10,00 kg	10,00 kg	10,00 kg	10,00 kg
Fläche oben	20,00 kg	20,00 kg	20,00 kg	20,00 kg	20,00 kg	20,00 kg
Knoten	30,00 kg	30,00 kg	30,00 kg	30,00 kg	30,00 kg	30,00 kg
Zylinder	15,20 kg	35,10 kg	44,90 kg	90,00 kg	133,00 kg	156,00 kg
zusätzlich plus Hub	1,20 kg	1,32 kg	2,69 kg	3,77 kg	6,62 kg	8,32 kg
zusätzlich plus Hub	10,32 kg	10,69 kg	20,98 kg	26,96 kg	45,35 kg	58,66 kg
Dachlast	85,00 kg	85,00 kg	85,00 kg	85,00 kg	85,00 kg	85,00 kg
Anzahl						
Flächen unten	x 144	x 144	x 144	x 144	x 144	x 144
Flächen oben	x 144	x 144	x 144	x 144	x 144	x 144
Knoten	x 91	x 91	x 91	x 91	x 91	x 91
Zylinder	x 702	x 702	x 702	x 702	x 702	x 702
Gewicht gesamt						
Fläche unten	1,440 t	1,440 t	1,440 t	1,440 t	1,440 t	1,440 t
Fläche oben	2,880 t	2,880 t	2,880 t	2,880 t	2,880 t	2,880 t
Knoten	2,730 t	2,730 t	2,730 t	2,730 t	2,730 t	2,730 t
Zylinder	17,915 t	32,146 t	46,249 t	82,103 t	125,200 t	150,689 t
Summe Eigengewicht	24,965 t	39,196 t	53,299 t	89,153 t	132,250 t	157,739 t
Dachlast	47,701 t	47,701 t	47,701 t	47,701 t	47,701 t	47,701 t
Gesamt	72,666 t	86,897 t	101,000 t	136,853 t	179,950 t	205,439 t
Kraft zu Gewicht						
Fläche unten	10 kg	10 kg	10 kg	10 kg	10 kg	10 kg
Fläche oben	20 kg	20 kg	20 kg	20 kg	20 kg	20 kg
Knoten	30 kg	30 kg	30 kg	30 kg	30 kg	30 kg
Zylinder x 9	230 kg	412 kg	593 kg	1053 kg	1605 kg	1932 kg
Dachlast x Fläche	331 kg	331 kg	331 kg	331 kg	331 kg	331 kg
Gesamt	621 kg	803 kg	984 kg	1444 kg	1996 kg	2323 kg
1 G	6,09 kN	7,88 kN	9,65 kN	14,16 kN	19,58 kN	22,79 kN
Faktor	x 23	x 30	x 35	x 42	x 35	x 30
Gesamt	142,11 kN	236,44 kN	337,92 kN	594,90 kN	685,46 kN	683,71 kN

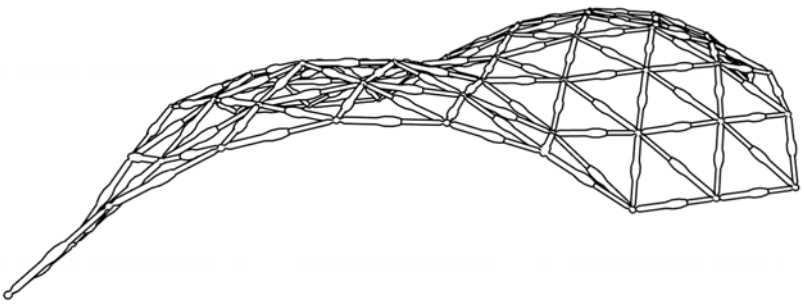
Tab.2.3.5-1 Annahme des Widerstandes

2.3.6 OPTIMIERUNG

Um den idealen Kräfteverlauf für die Form zu finden, den die Tragstruktur gerade einnimmt, dienen die Hängemodelle von Antonio Gaudi und die Gitterschalen von Frei Otto als analoge Vorbilder für digitale Simulation. In der Abbildung wird eine Fläche gezeigt, die an nutzerdefinierten Punkten fixiert und dann mittels Cloth-Simulation fallen gelassen wurde. Durch Umkehren der Schwerkraft von -9,81m/s2 in +9,81m/s2 wird die Tragstruktur wie bei Antonio Gaudi von Zug-Belastung in Druckbeanspruchung umgekehrt. Durch diese Optimierung kann der Kraftaufwand, den die Motoren für die Größenänderung der Dreiecke aufwenden müssen, reduziert werden. Ein kostengünstigerer und umweltschonender Einsatz wird dadurch möglich.

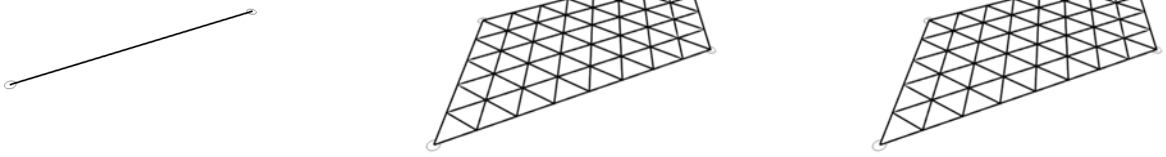
Im weiteren Verlauf der Arbeit wird dieser Zugang noch genauer erklärt und zu einem zentralen Thema werden.

Verweis auf:
Kettenlinie, Seite 184

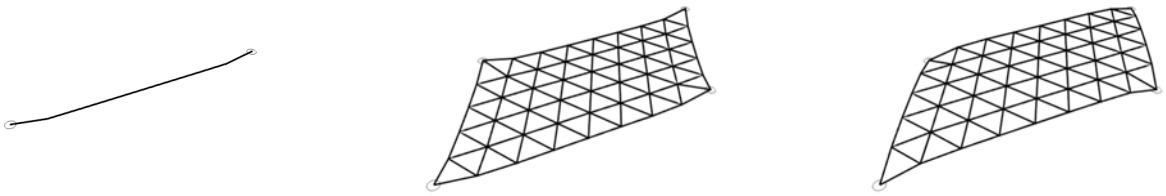


Zchng.2.3.6-1 Cloth-Simulation in Anwendung

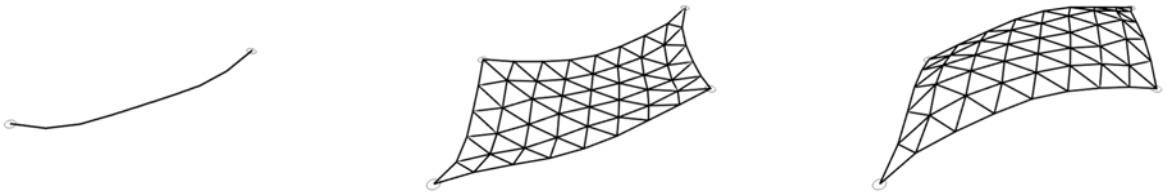
Optimierung
Cloth-Simulation
Kette, fallendes Tuch, Umkehrung



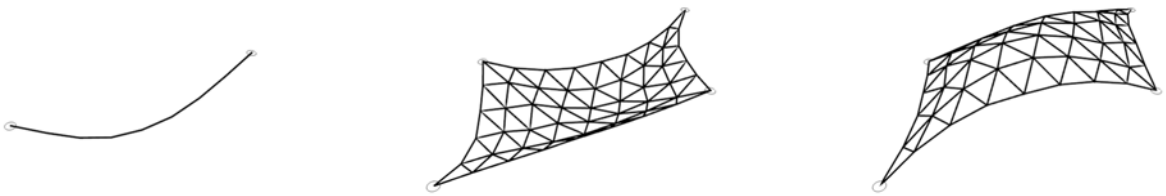
Zchng.2.3.6-2 a,b,c Ausgangsposition



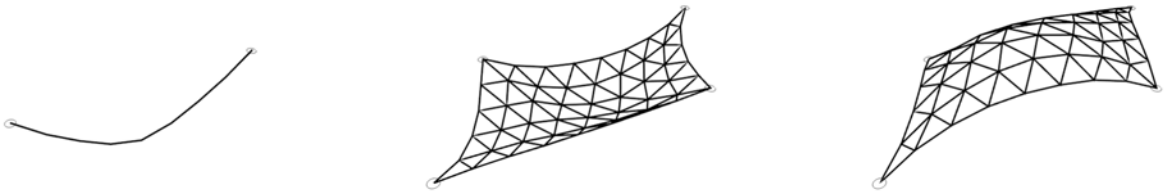
Zchng.2.3.6-3 a,b,c 20 %



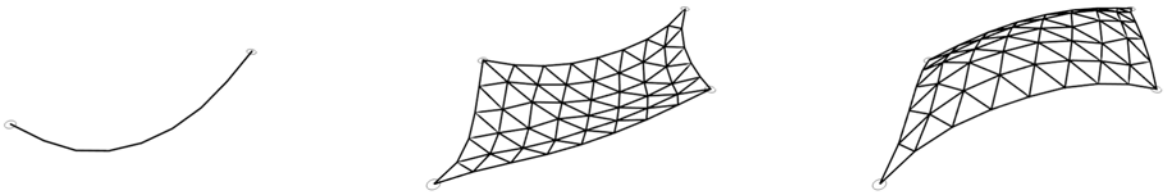
Zchng.2.3.6-4 a,b,c 40 %



Zchng.2.3.6-5 a,b,c 60 %



Zchng.2.3.6-6 a,b,c 80 %



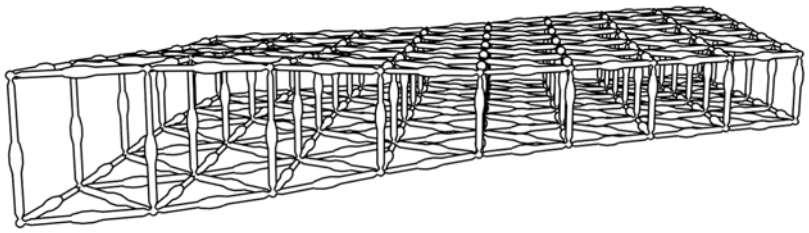
Zchng.2.3.6-7 a,b,c Endposition

Grundsätzlich gibt es einen Zielkonflikt in Bezug auf die Tragwerkshöhe. Einerseits führt eine Maximierung der Höhe zu einer potentiell höheren Spannweite, anderseits erhöht das den Weg, den einzelne Motoren abfahren müssen.

Ein weiterer Versuch, den Energieaufwand und maximale Spannweite zu optimieren, ist das Aufdicken der Struktur an gezielten Punkten. Dies geschieht, wie im Abschnitt Steuerung erklärt, mittels Proportional-Editing.

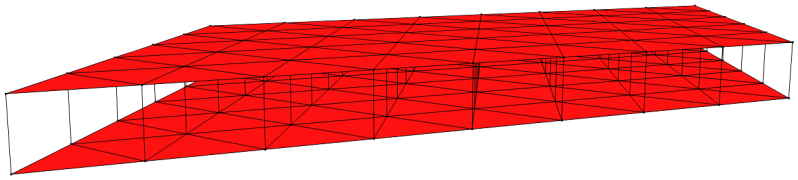
Verweis auf:
Steuerung, Seite 60

Das Ergebnis der veränderten Höhe ist im Diagramm unten rechts zu sehen. Rote Bereiche stellen dabei den niedrigsten und blaue Bereiche den maximalen Abstand zur gegenüberliegenden Fläche dar.

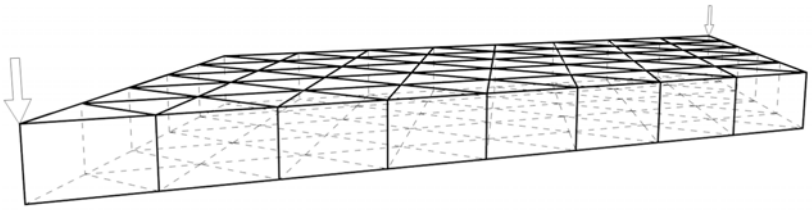


Zchng.2.3.6-8 Höhenvariation in Anwendung

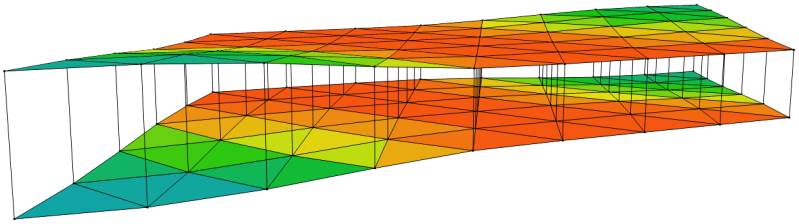
Optimierung
Dicke
gleichförmig, erwartete Krafteinwirkung, Reaktion



Diagr.2.3.6-1 Dicke skaliert auf 1,40m bis 1,80m



Diagr.2.3.6-2 erwartete Krafteinwirkung



Diagr.2.3.6-3 Dicke skaliert auf 1,40m bis 1,80m

2.3.7 PROTOTYP 1

Die folgenden Fotos zeigen den ersten gebauten Prototyp. Er bestand aus 18 Motoren, die jeweils unabhängig voneinander angesteuert werden konnten. Der Screenshot unten rechts zeigt ein exaktes Abbild des Modells. Große Schwächen zeigte das Modell in erster Linie in den Knotenpunkten. Daraus entstand in Folge ein eigenes Kapitel, dass sich speziell diesem Problem widmet.

Verweis auf:
Knotenpunkte, Seite 110

Aus Kostengründen wurde dieses Modell schließlich demontiert und zu einem verbesserten Prototyp verfeinert.

Verweis auf:
Prototyp 2, Seite 124

Folgeseite:
Abb.2.3.7-1 Modellfoto Prototyp 1

Prototyp 1
18 Servomotoren, Polystyrol, Acryl, Metall
Verbindungen geschraubt oder mittels Kabelbindern

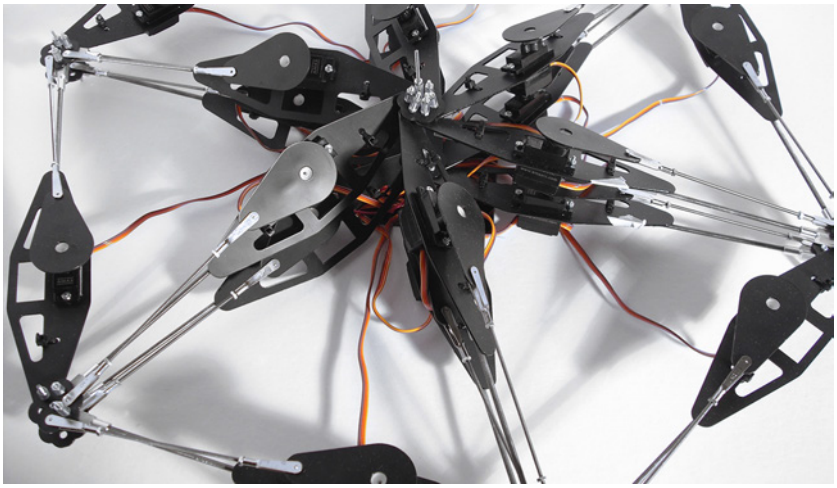


Abb.2.3.7-2 Foto gesamt

Abb.2.3.7-3 Foto Detail

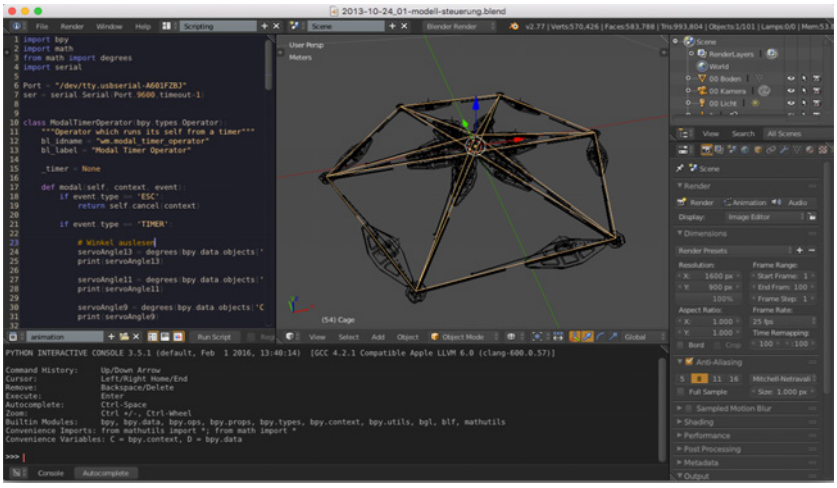


Abb.2.3.7-4 Screenshot des 3D-Modells


M2
M2


AMAX

2.4 BRANCHING

In diesem Abschnitt wird versucht, eine möglichst effiziente Verbindung von den einzelnen Motoren zu einer zentralen Steuerung zu finden. Der Script funktioniert einfach: Er iteriert durch jeden einzelnen Punkt der Fläche und nutzt das blender-interne Shortest-Path, um den kürzesten Weg zum ausgewählten Punkt zu finden. Anschließend werden die gefundenen Pfade extrahiert.

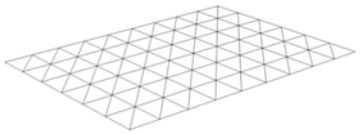
Festzuhalten ist die unterschiedliche Erscheinung, die die Fläche durch diese Bündelung annimmt. Das Festlegen eines Mittelpunktes und der Verlauf aller Linien zu diesem ist eine architektonische Entscheidung, die festgelegt werden muss.

Folgeseite:
Abb.2.4.0-1 Rendering

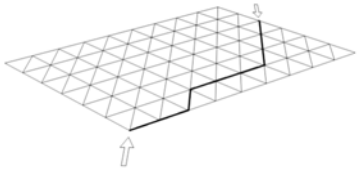
Erzeugen der Verbindungen
Scriptsprache: Python
DCC: Blender

```python
import bpy

# Ausgangspunkt im Gitternetz markieren

# Anzahl der Punkte evaluieren
Zaehler = len (bpy.context.active_object.data.vertices)
Zaehler_original = Zaehler
Zaehler_geloescht = Zaehler -1
Zaehler = Zaehler -1
Iterationen = Zaehler

# Punkte sortieren
bpy.context.area.type='VIEW_3D'
bpy.ops.view3D.snap_cursor_to_selected()
bpy.context.area.type='TEXT_EDITOR'
bpy.context.area.type='VIEW_3D'
bpy.ops.mesh.sort_elements(type='SELECTED', elements={'VERT'})
bpy.context.area.type='TEXT_EDITOR'
bpy.ops.mesh.select_all(action='DESELECT')
bpy.ops.object.editmode_toggle()

for i in range(Iterationen):
    # Kürzesten Pfad finden
    bpy.context.active_object.data.vertices[0].select = True
    bpy.context.active_object.data.vertices[Zaehler].select = True
    bpy.ops.object.editmode_toggle()
    bpy.ops.mesh.shortest_path_select()

    # Volumen erzeugen
    bpy.ops.mesh.duplicate_move(MESH_OT_duplicate={"mode":1}, TRANSFORM_OT_translate={"value":(0, 0, 0), "constraint_axis":(False, False, False), "constraint_orientation":'GLOBAL', "mirror":False, "proportional":'DISABLED', "proportional_edit_falloff":'SMOOTH', "proportional_size":1, "snap":False, "snap_target":'CLOSEST', "snap_point":(0, 0, 0), "snap_align":False, "snap_normal":(0, 0, 0), "texture_space":False, "remove_on_cancel":False, "release_confirm":False})

    # Loop vervollständigen
    bpy.ops.mesh.select_all(action='DESELECT')
    bpy.ops.object.editmode_toggle()
    Zaehler = Zaehler -1
```

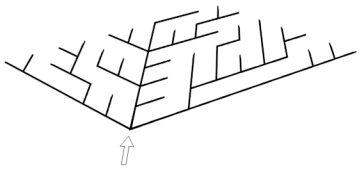
Scp.2.4.0-1 Aufteilen



Zchng.2.4.0-1 Ausgangsgeometrie

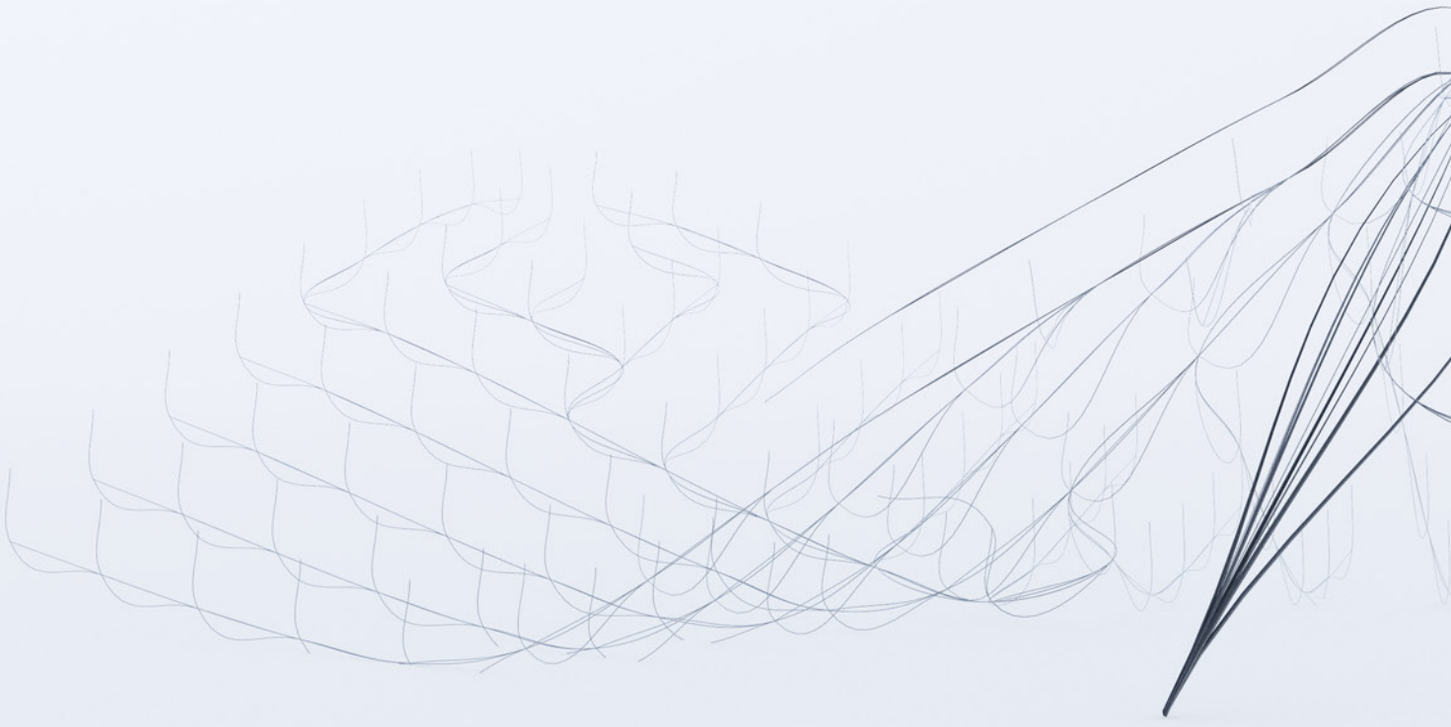


Zchng.2.4.0-2 einzelner Pfad



Zchng.2.4.0-3 alle Pfade

Zchng.2.4.0-4 Differenzierung



2.5 HÜLLE

Welche Hüllen können der Transformation der Tragstruktur folgen? Es handelt sich nicht nur um eine technische Frage nach möglicher Verformbarkeit, sondern besonders um deren architektonische Wirkung. Die Hülle funktioniert schließlich als Fassade der Struktur und kann die Verformung des Tragwerks ausdrücken beziehungsweise noch verstärken. Dabei geht es überwiegend darum, welche Transformation die Hülle selbst vollziehen muss, wie beispielsweise Falten, Dehnen, Stauchen, Schuppen, Raffen oder Auflösen. Das gilt besonders für Extremsituation wie Faltenwurf, Ausdünnen, Aufblähen und Weiteren. Die Ornamentik dieser Prozesse gilt es zu untersuchen.

Zudem liegt der Fokus auf dem entstehenden Dialog zwischen Hülle und Tragwerk. Das Tragwerk ist polygonal und manche Strategien erlauben, diesem Polygonzug zu folgen. Andere Hüllen ermöglichen eine Idealisierung der kantigen Struktur zu weichen Kurven, oder eine Loslösung von Tragwerk und Hülle. Die einzelnen Strategien sind in Bezug auf deren Funktion als thermische Trennung, Witterungsschutz beziehungsweise Lichtdurchlässigkeit unterschiedlich leistungsstark. Hüllen können kombiniert werden, um sich im entsprechenden Einsatz gegenseitig zu ergänzen.

2.5.1 AUFLÖSEN

Unter Auflösen wird verstanden, dass das Tragwerk an manchen Stellen lediglich auf seine statische Funktion reduziert wird.

Eine weitere Option wäre das Auflösen der Hülle nach dem Vorbild von John Lautners Sheats-Goldstein-Residenz in Form eines Luftschleiers. Bei Pneumatik als Antriebsart kann möglicherweise die Druckluft zur Raumbildung mitbenutzt werden. Dieses System kann technisch mit einem Windfang von Kaufhäusern verglichen werden. Wobei die Leistungsstärke dieser Form von thermischer Trennung hinterfragt, beziehungsweise gesteigert werden muss. Insbesondere ist der Energiebedarf zu überprüfen.

Folgeseite:
Abb.2.5.1-1 Rendering

optisch	nicht vorhanden
thermisch	nicht vorhanden
akustisch	nicht vorhanden
mechanisch	nicht vorhanden

Tab.2.5.1-1 Potenzial

Auflösen
Größenänderung
einzeln, addiert, umgekehrt

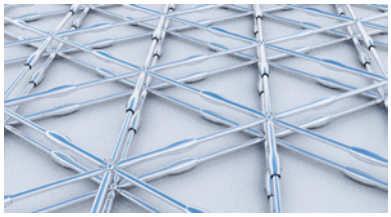
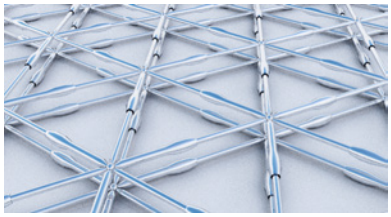




Abb.2.5.1-2 a,b,c Ausgangsposition

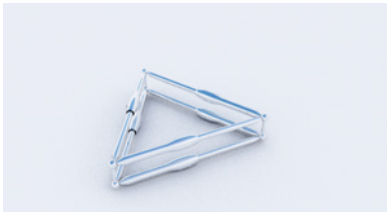
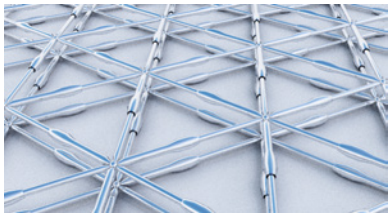
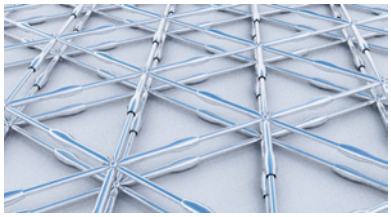




Abb.2.5.1-3 a,b,c 20 %

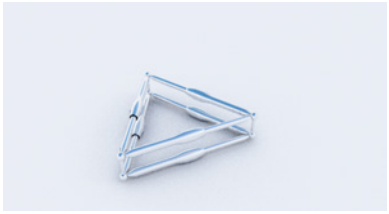
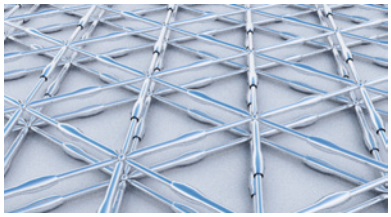
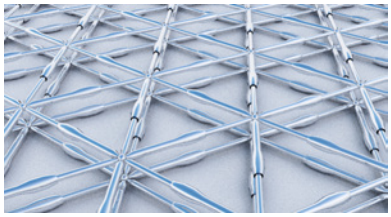




Abb.2.5.1-4 a,b,c 40 %

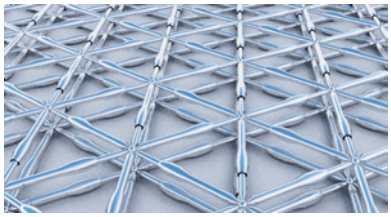
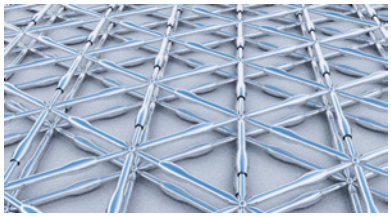




Abb.2.5.1-5 a,b,c 60 %

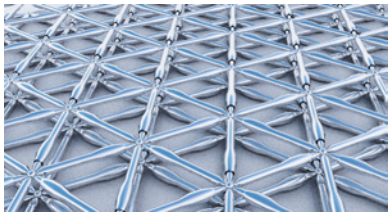
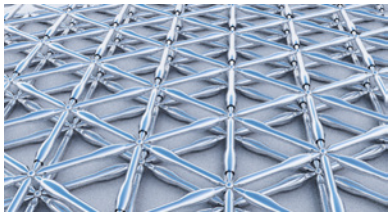




Abb.2.5.1-6 a,b,c 80 %

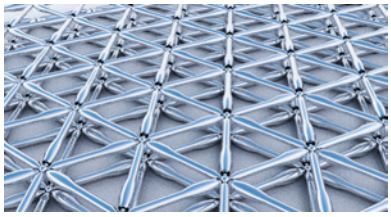
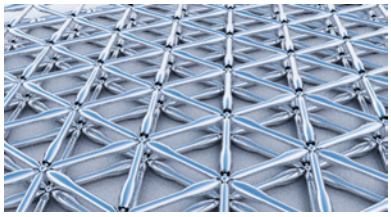
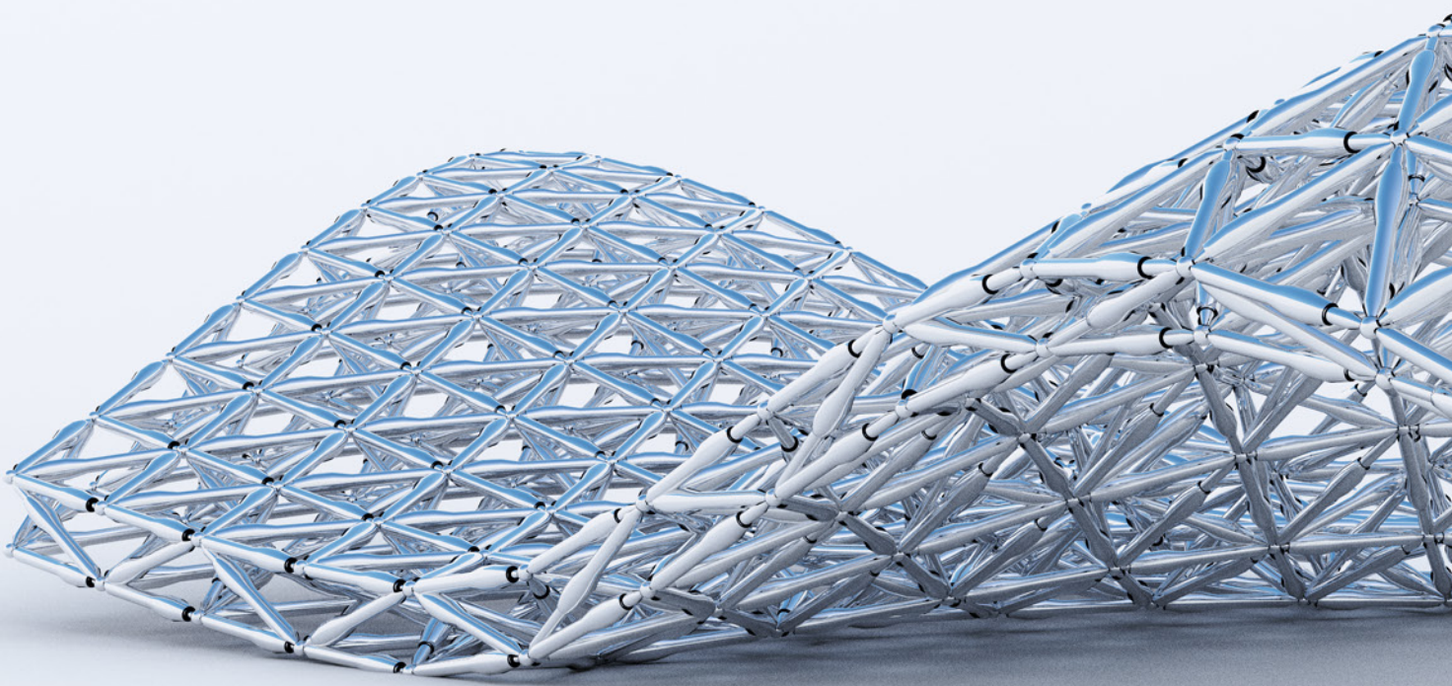
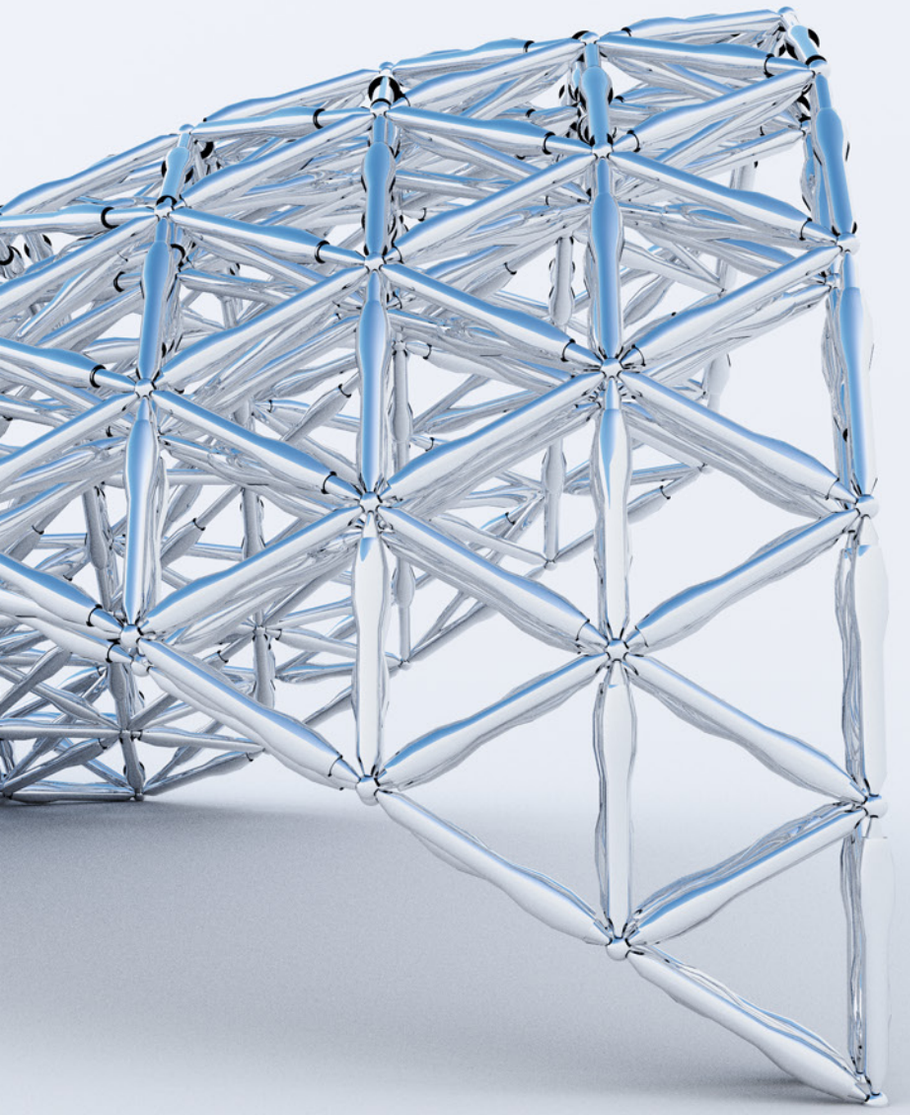




Abb.2.5.1-7 a,b,c Endposition





2.5.2 F A L T E N

Ein spannender Weblink mit Infos über Ron Resch:
www.ronresch.org

Bei dem verwendeten Falt-Muster handelt es sich um das Resch-Pattern, das der Mathematiker Ron Resch entwickelt hat. Es erlaubt eine Größenänderung der Hülle um rund 173 %. Je kleiner das Pattern zusammengefahren wird, desto mehr steigt die Höhe der Hülle.

Die Hülle könnte aus einzelnen, mit Scharnieren verbunden Flächen erzeugt werden. Das würde den Vorteil bieten, dass das Material nicht durch langfristiges Knicken geschwächt wird. Jedoch bietet das tatsächliche Falten aus größtmöglichen Flächen den Vorteil, dass die Dichtheit der Hülle gewährleistet ist. Zudem sind keine mechanischen Verbindungen zu sehen. Darüber hinaus entsteht beim Biegen der Flächen, die sich wieder zurück in ihre Knickpositionen verformen wollen, eine Durchlaufwirkung. Diese Durchlaufwirkung erlaubt, das polygonale Tragwerk mit einer idealisierten, doppelt gekrümmten Fläche zu überziehen. Das Pattern selbst kann auch kleiner skaliert werden, um diesen Effekt zu verstärken.

Die Größenänderung dieser Hülle hat einen Überraschungseffekt, da sie in dieser Art und Weise nicht vorhersehbar ist.

Folgeseite:
Abb.2.5.2-1 Rendering

optisch	transparent und transluzent möglich
thermisch	möglich
akustisch	wenig Masse, Reflexion prüfen
mechanisch	möglich

Tab.2.5.2-1 Potenzial

Falten
Größenänderung
einzeln, addiert, umgekehrt

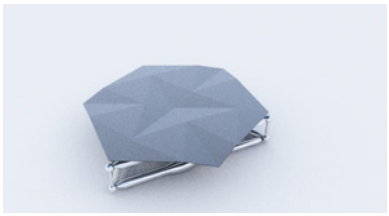
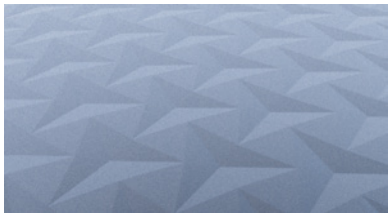
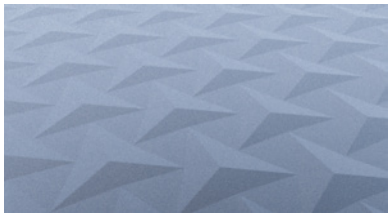




Abb.2.5.2-2 a,b,c Ausgangsposition

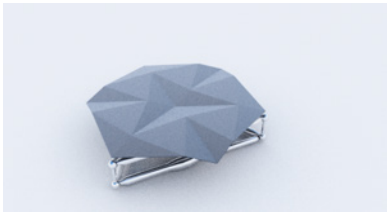
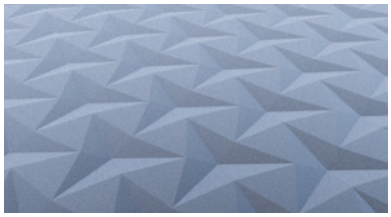
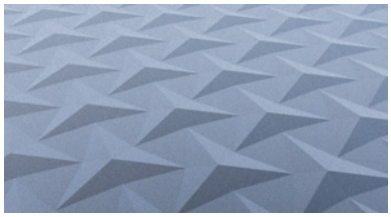




Abb.2.5.2-3 a,b,c 20 %

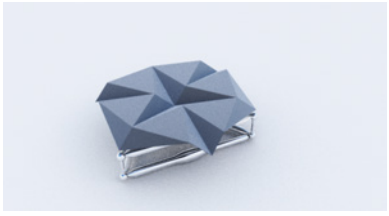
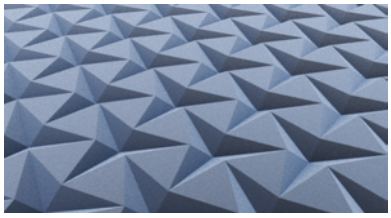
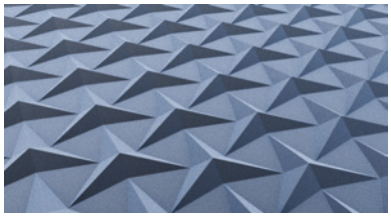




Abb.2.5.2-4 a,b,c 40 %

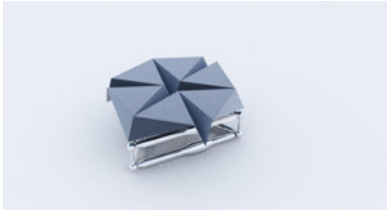
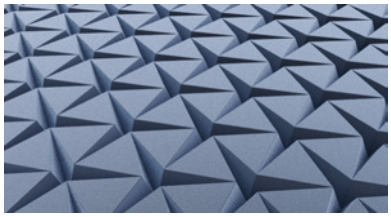
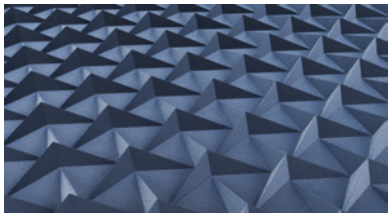




Abb.2.5.2-5 a,b,c 60 %

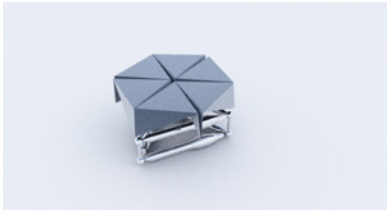
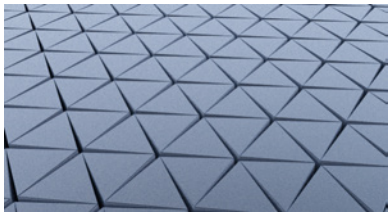
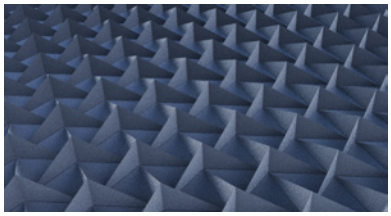




Abb.2.5.2-6 a,b,c 80 %



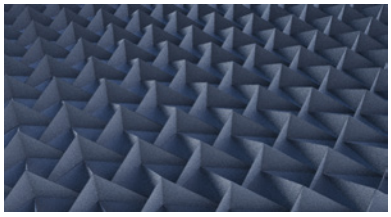
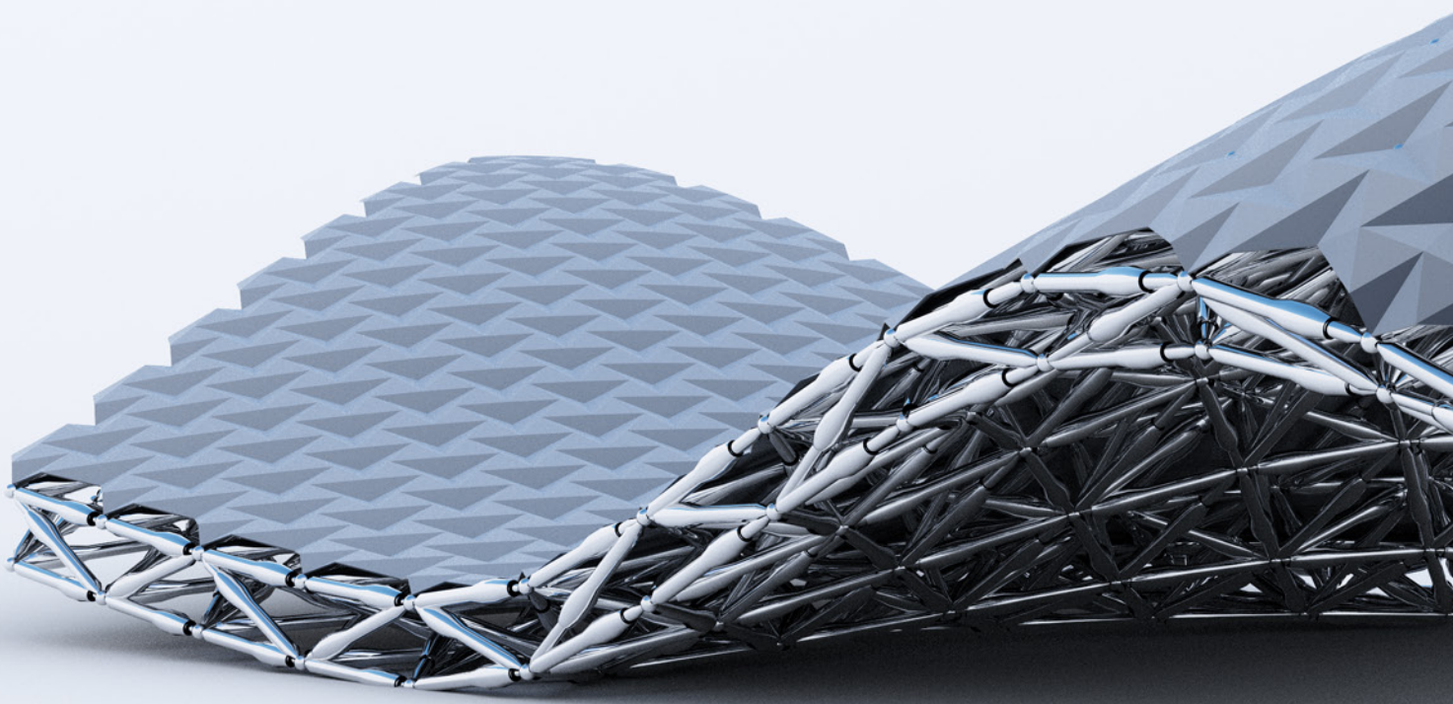
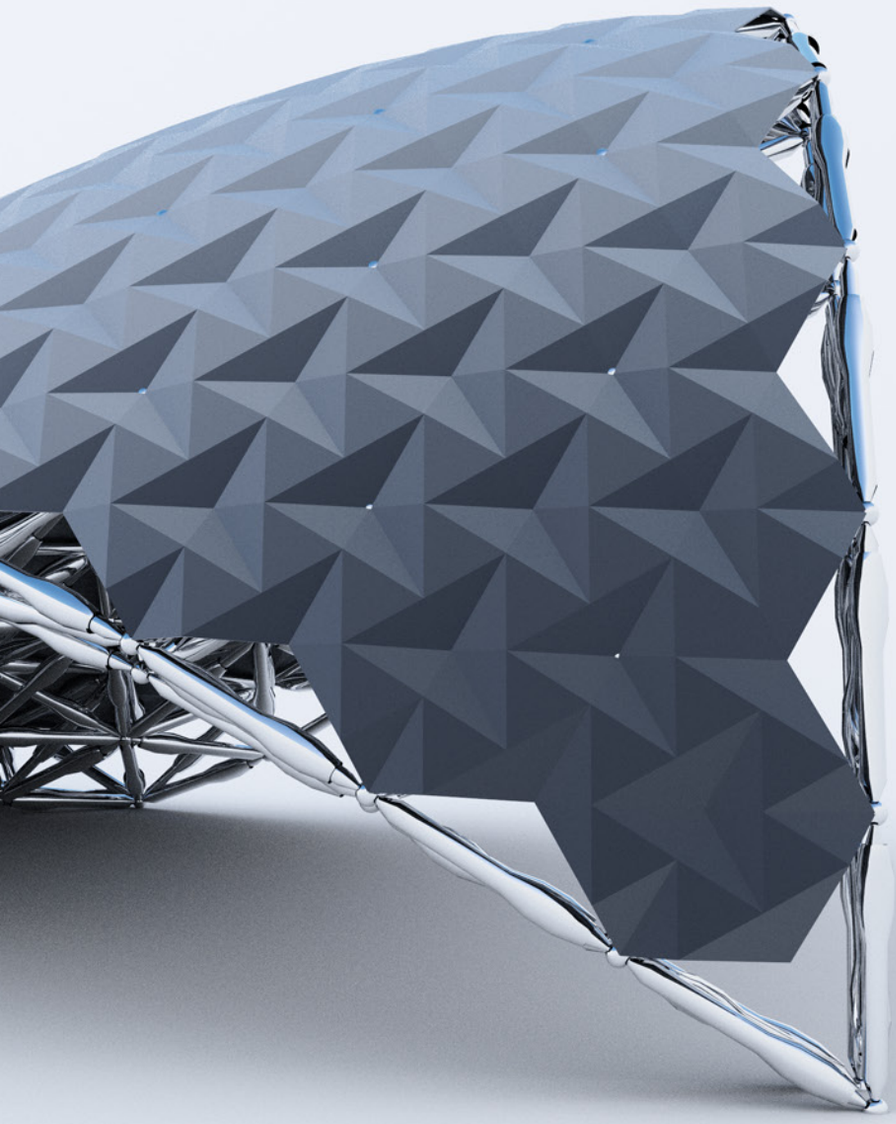


Abb.2.5.2-7 a,b,c Endposition





2.5.3 D E H N E N

Das Prinzip Dehnen basiert auf dem Bespannen der Tragstruktur mit Folien oder Stoffen. Dabei sind zwei Prinzipien möglich. Dehnbare Materialien resultieren in einer polygonalen Hülle. Der Effekt der Größenänderung ist insofern besonders, da die Hülle scheinbar skaliert wird. Bei nicht dehnbaren Materialien müssen diese vorab überdimensioniert produziert werden, um nicht im ausgefahrenem Zustand zu reissen. Im zusammengefahrenen Zustand hängen diese schließlich durch.

„Die Decke ist das älteste Architekturdetail. Ursprünglich waren es Felle oder Erzeugnisse der Textilkunst, welche Bedeutung sie in den germanischen Sprachen heute noch besitzt. Diese Decke musste irgendwo angebracht werden, sollte sie genügend Schutz für eine Familie bieten! Bald kamen die Wände dazu, um auch seitlichen Schutz zu bieten. Und in dieser Reihenfolge entwickelte sich der bauliche Gedanke sowohl in der Menschheit als auch im Individuum."

Adolf Loos, „Das Princip der Bekleidung", 1898

So schreibt Adolf Loos im Jahre 1898. Zu ergänzen ist: Bekleidung war von jeher flexibel und nicht starr. Der Begriff „Wand", findet sich im Begriff „Gewand" wieder. Beduinen-Architektur und andere nomadische Zeltkonstruktionen sind beweglich und nach Abbau auch bewegbar.

Folgeseite:
Abb.2.5.3-1 Rendering

optisch	transparent und transluzent möglich
thermisch	zu wenig Masse
akustisch	zu wenig Masse
mechanisch	zu wenig Widerstand

Tab.2.5.3-1 Potenzial

Dehnen
Größenänderung
einzeln, addiert, umgekehrt



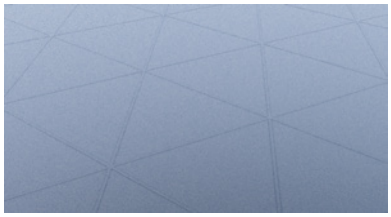


Abb.2.5.3-2 a.b.c Ausgangsposition

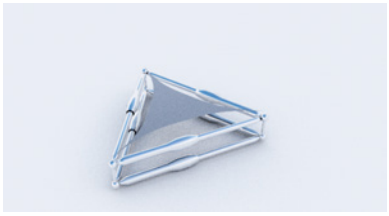


Abb.2.5.3-3 a,b,c 20 %

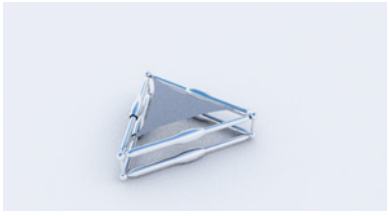
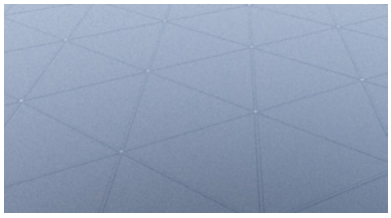




Abb.2.5.3-4 a,b,c 40 %

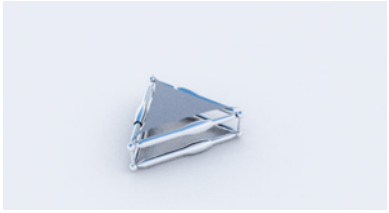
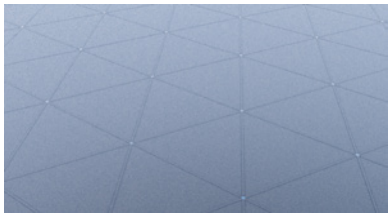




Abb.2.5.3-5 a,b,c 60 %

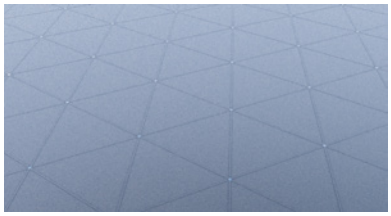




Abb.2.5.3-6 a,b,c 80 %

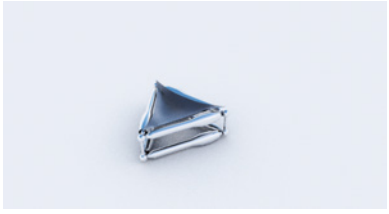
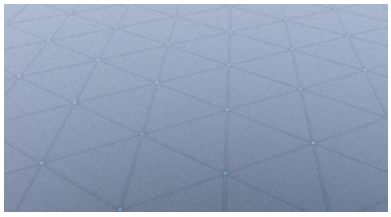
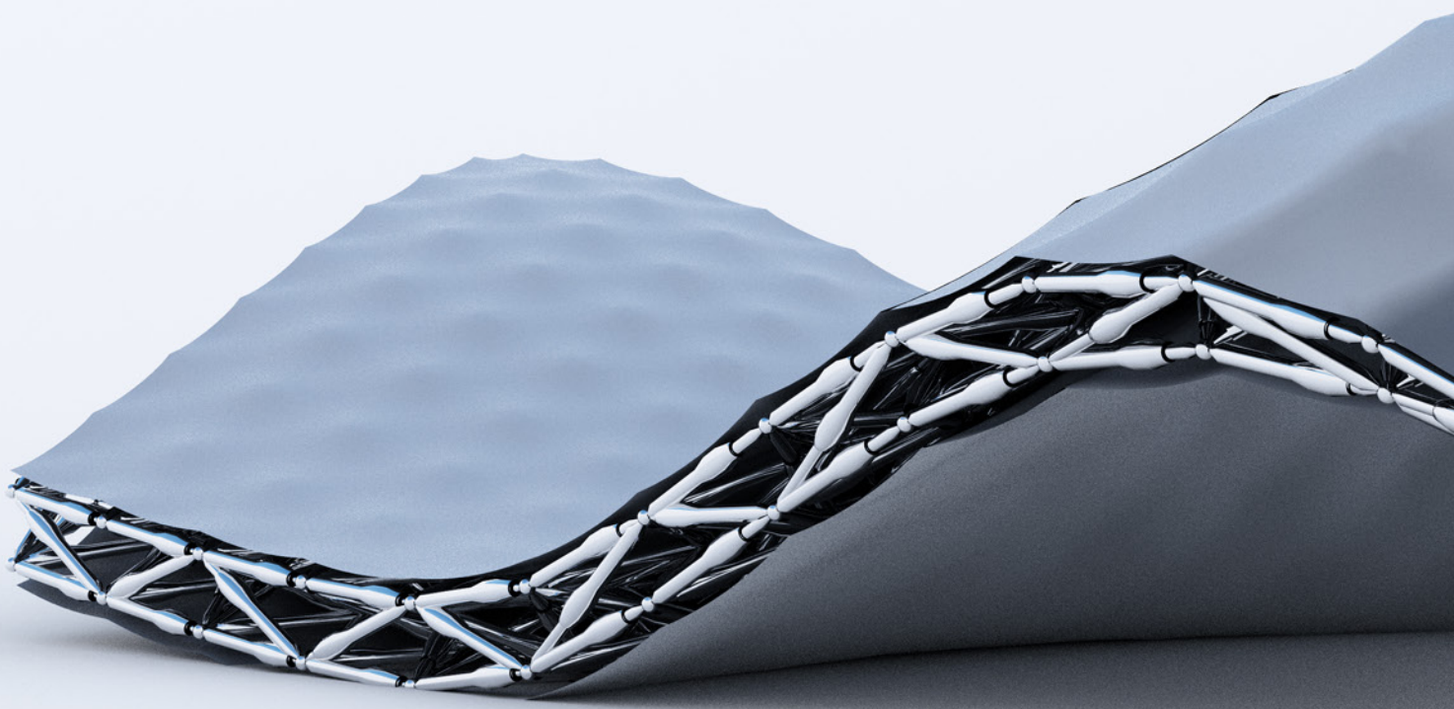
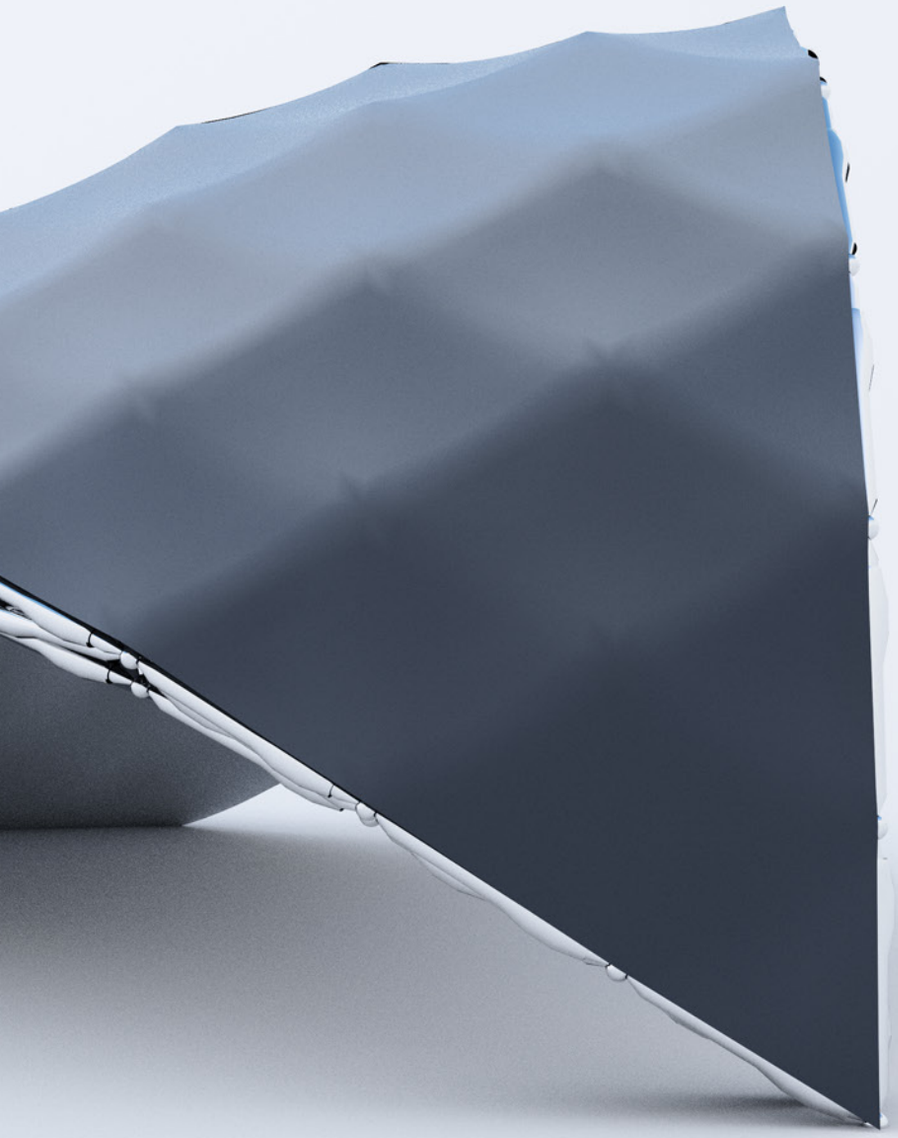




Abb.2.5.3-7 a,b,c Endposition





2.5.4 S C H U P P E N

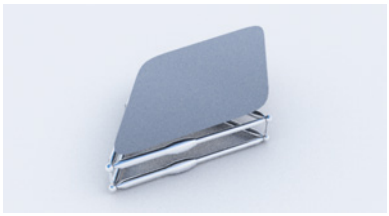
Das Prinzip Schuppen funktioniert analog zu Dachziegeln. Es gelten dabei die gleichen Einschränkungen wie beispielsweise die minimale Steigung. Grundsätzlich kann zwischen flexiblen und nicht verformbaren Schuppen unterschieden werden. Flexibel verformbare Schuppen würden sich bei konkaven Flächen idealisiert verformen und bei konvexen Flächen eine Art Sheddach erzeugen. Dieses kann für unterschiedliche Belichtungsszenarios sinnvoll eingesetzt werden. Starre Schuppen bilden immer eine Form von Sheddach.

Folgeseite:
Abb.2.5.4-1 Rendering

optisch	transparent und transluzent möglich
thermisch	schwierig, da nicht vollständig geschlossen
akustisch	möglich, Reflexion prüfen
mechanisch	möglich

Tab.2.5.4-1 Potenzial

Schuppen
Größenänderung
einzeln, addiert, umgekehrt



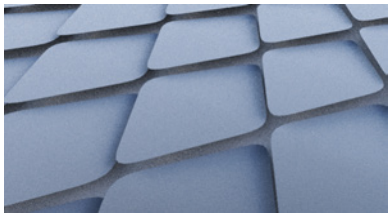
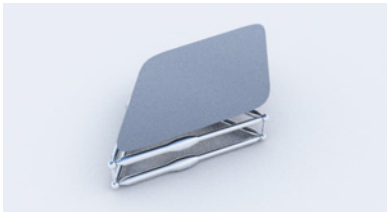


Abb.2.5.4-2 a,b,c Ausgangsposition



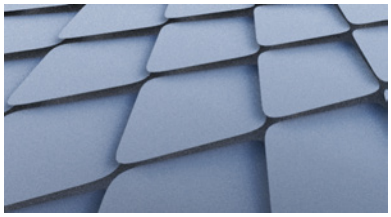


Abb.2.5.4-3 a,b,c a,b,c a,b,c 20 %

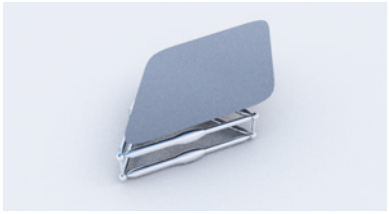
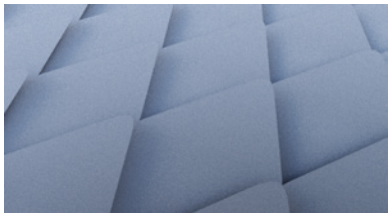
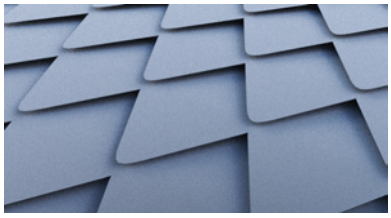




Abb.2.5.4-4 a,b,c a,b,c a,b,c 40 %

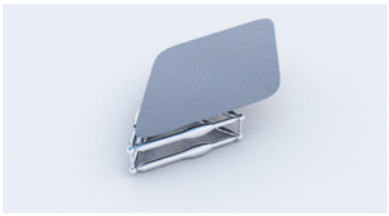
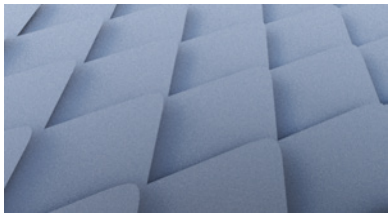




Abb.2.5.4-5 a,b,c a,b,c a,b,c 60 %

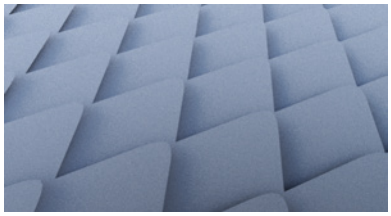




Abb.2.5.4-6 a,b,c a,b,c a,b,c 80 %

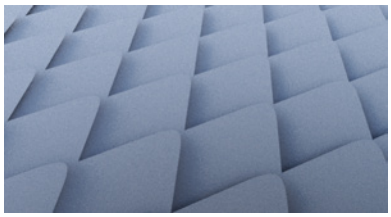
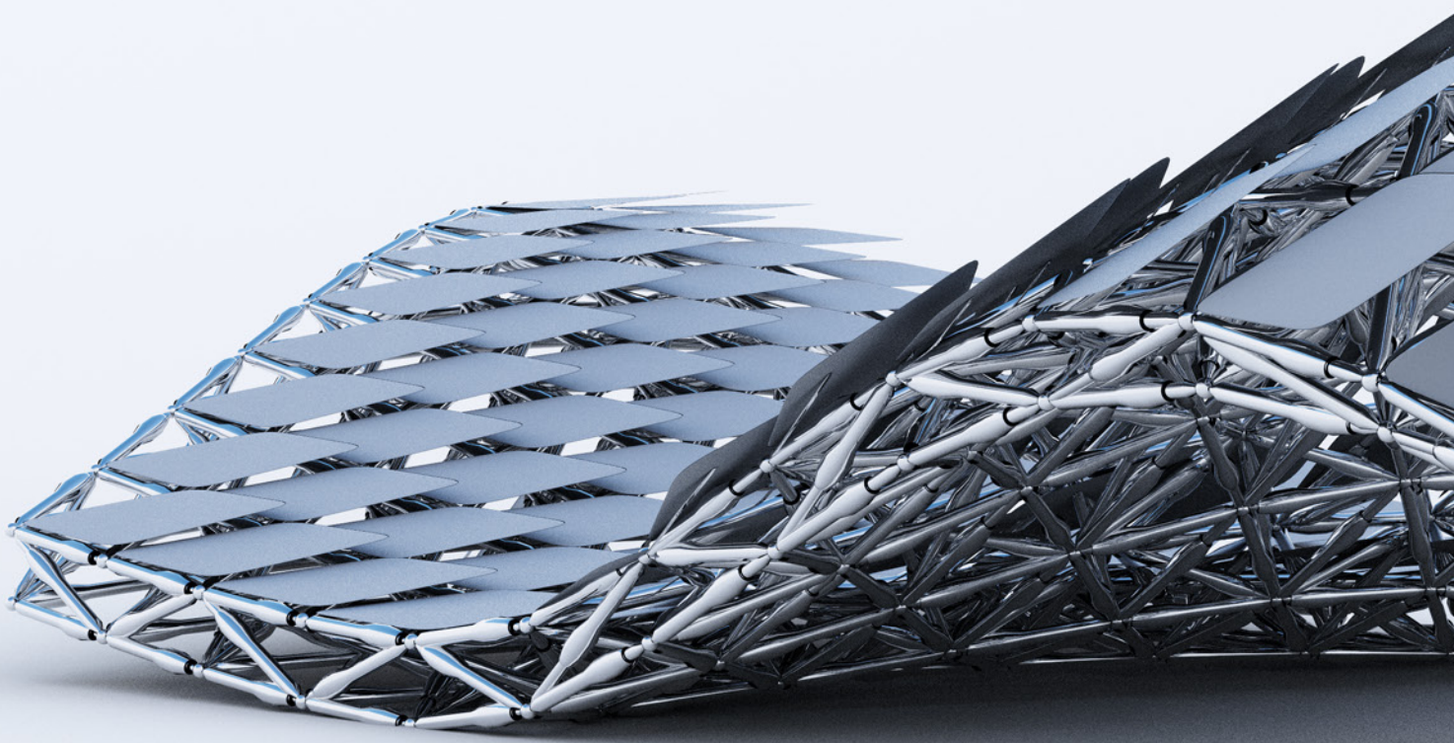
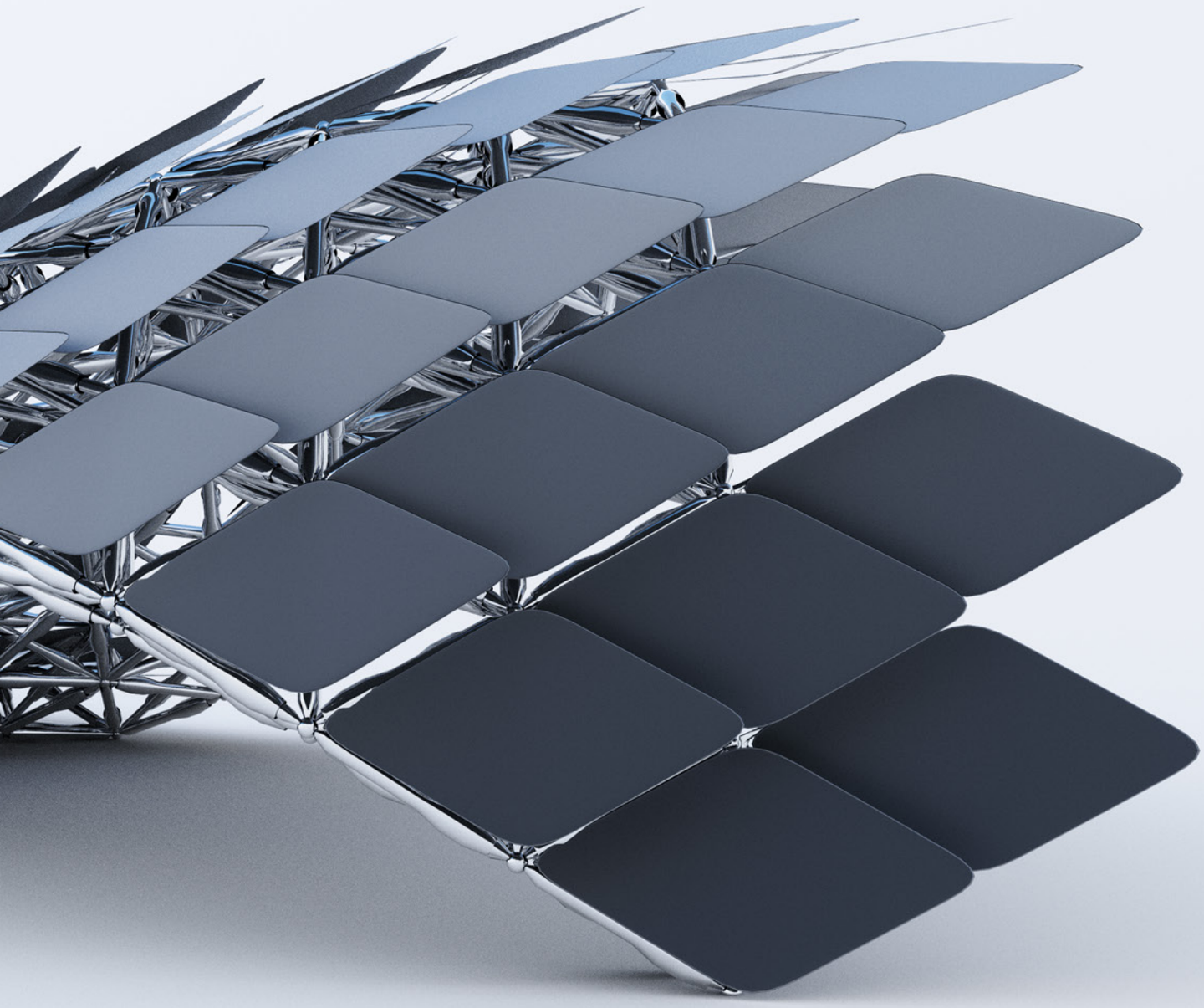




Abb.2.5.4-7 a,b,c a,b,c a,b,c Endposition





2.5.5 S T A U C H E N

Die Strategie Stauchen wäre möglicherweise mittels tiefgezogener Teile zu realisieren. Um eine Transformation zu ermöglichen, sind diese in alle drei Richtungen vorgeformt. Materialien wie GFK sind anfangs mit weniger Kraftaufwand zu verbiegen als gegen Ende der Verformung. Sie folgen daher dem Weg des geringsten Widerstands und resultieren in einer idealisierten Verformung.

Folgeseite:
Abb.2.5.5-1 Rendering

optisch	transparent und transluzent möglich
thermisch	schwierig, da wenig Masse
akustisch	möglich, Reflexion prüfen
mechanisch	möglich

Tab.2.5.5-1 Potenzial

Stauchen
Größenänderung
einzeln, addiert, umgekehrt



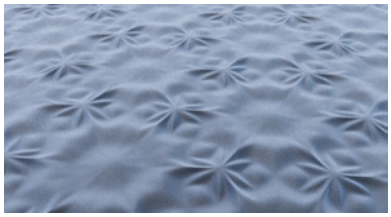


Abb.2.5.5-2 a,b,c Ausgangsposition

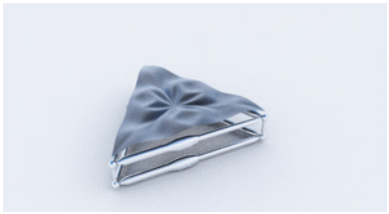
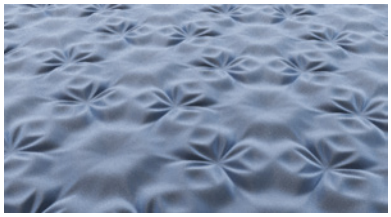
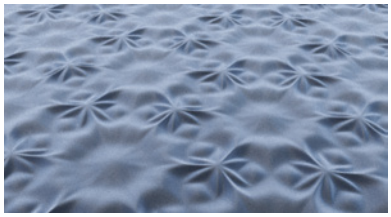




Abb.2.5.5-3 a,b,c 20 %

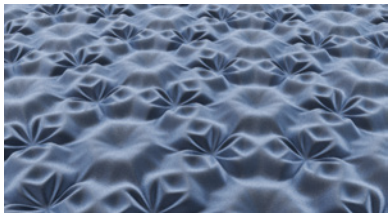
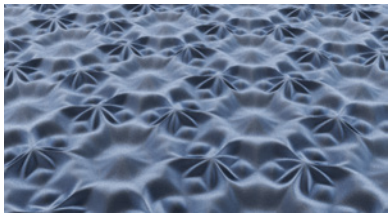




Abb.2.5.5-4 a,b,c 40 %

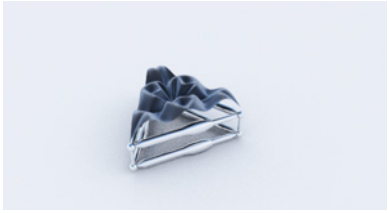
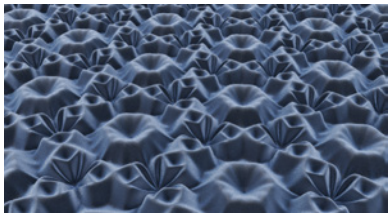
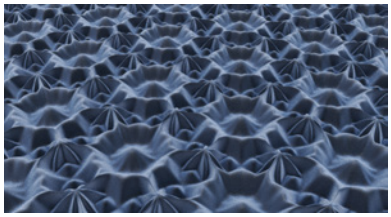




Abb.2.5.5-5 a,b,c 60 %

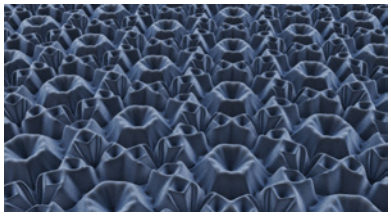




Abb.2.5.5-6 a,b,c 80 %

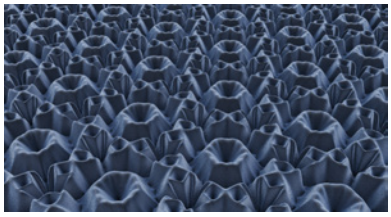
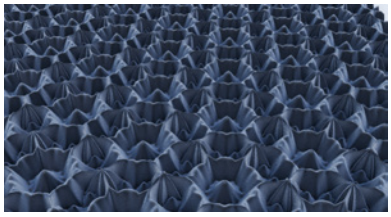
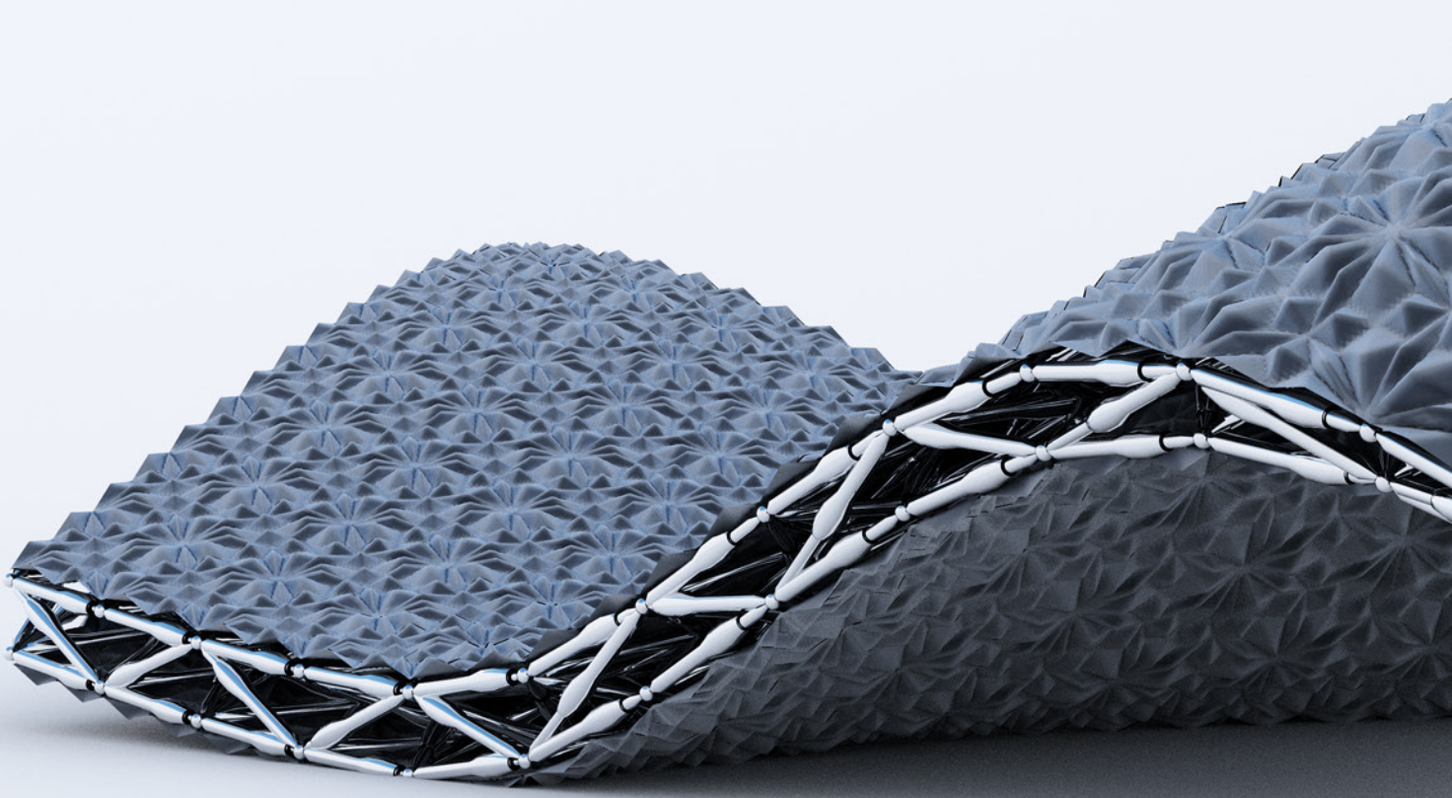
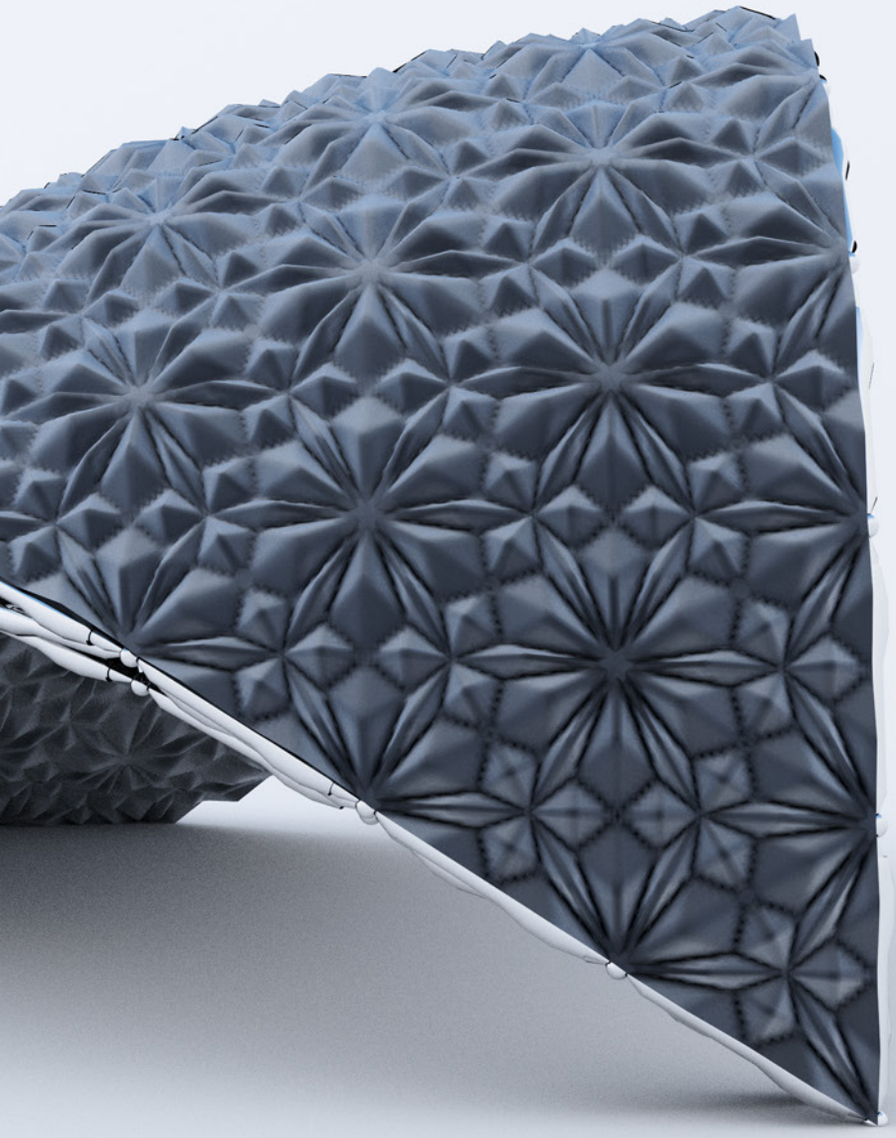




Abb.2.5.5-7 a,b,c Endposition





2.5.6 R A F F E N

Entgegen dem Prinzip Dehnen basiert das Prinzip Raffen auf befüllten Pneus beziehungsweise nicht dehnbaren Materialien. Bei maximaler Größe sind diese annähernd gespannt. Beim Zusammenfahren hängen sie durch und raffen sich. Da das Luftvolumen bei Größenänderung gleich bleibt, die Grundfläche der Membran schrumpft, verformt sich die Membran in die Höhe. Diese Strategie besticht besonders durch die komplexe Extremsituation beim Raffen. Besonders das Zusammenfahren mit Faltenwurf ist ausdrucksstark und macht die Verformung lesbar. Ein Gefühl von befüllt und schlaff wird assoziiert.

Folgeseite:
Abb.2.5.6-1 Rendering

optisch	transparent und transluzent möglich
thermisch	möglich
akustisch	zu wenig Masse
mechanisch	zu wenig Widerstand

Tab.2.5.6-1 Potenzial

Raffen
Größenänderung
einzeln, addiert, umgekehrt

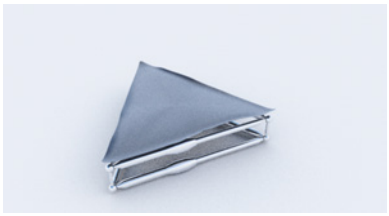
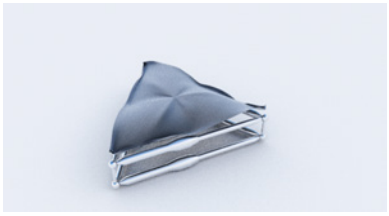


Abb.2.5.6-2 a,b,c Ausgangsposition



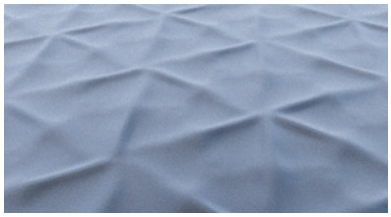


Abb.2.5.6-3 a,b,c 20 %



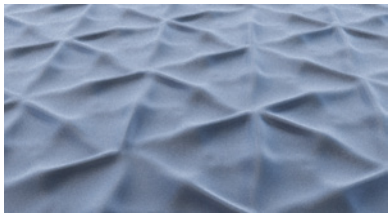


Abb.2.5.6-4 a,b,c 40 %



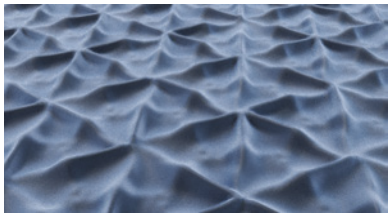


Abb.2.5.6-5 a,b,c 60 %



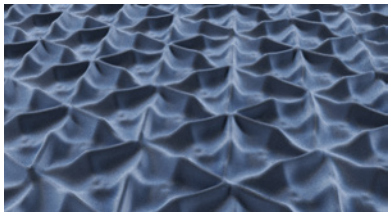


Abb.2.5.6-6 a,b,c 80 %



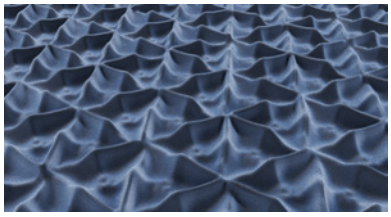
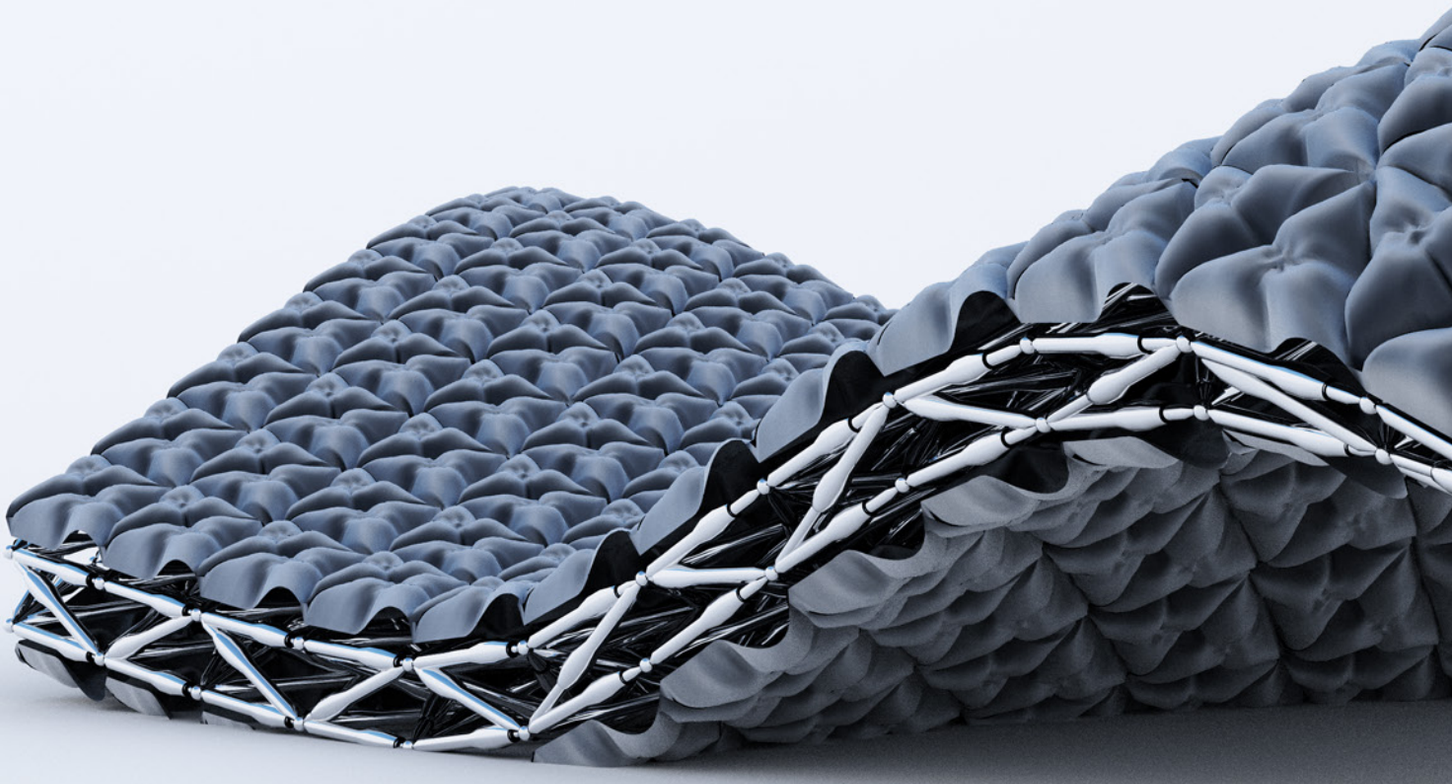
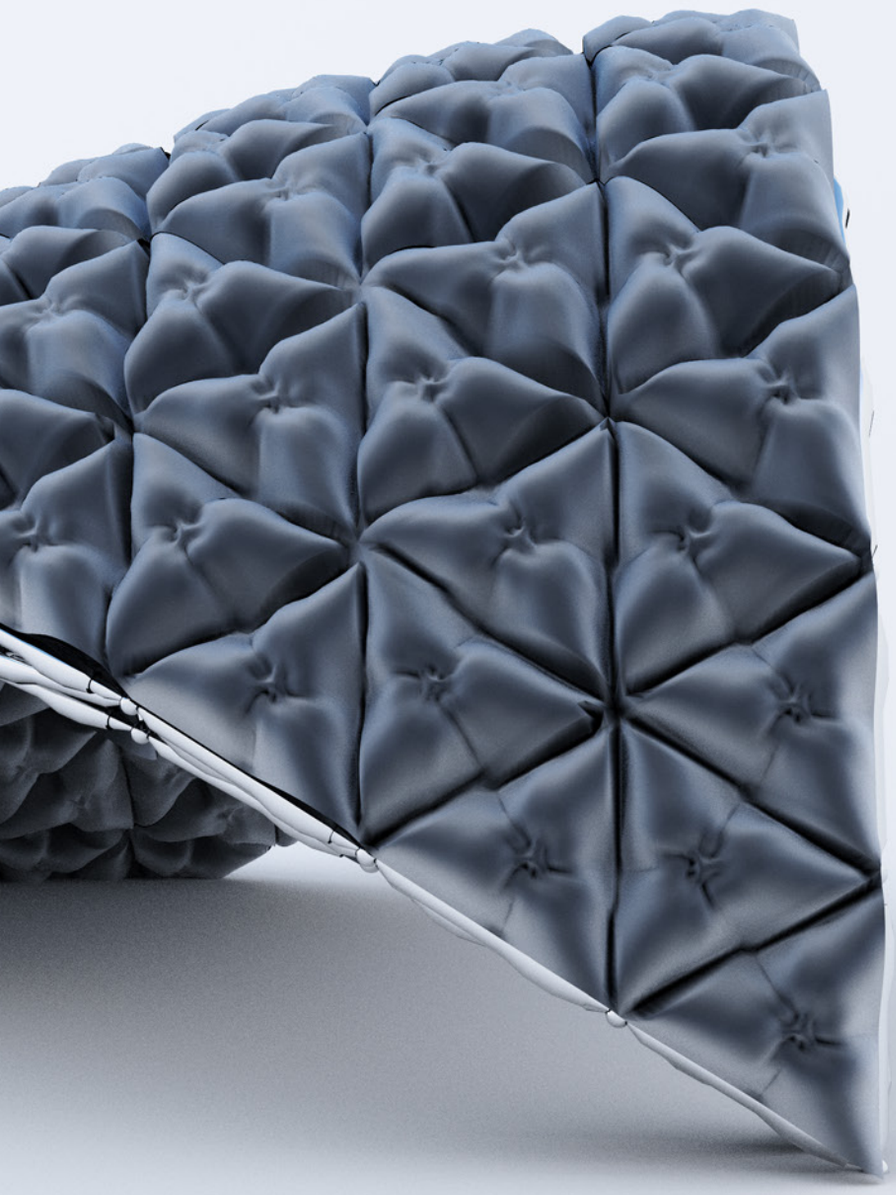


Abb.2.5.6-7 a,b,c Endposition





2.5.7 V E R D I C H T E N

Die Variante Verdichten sieht vor, eine so dichtes Netz aus Einzelelementen zusammen zu setzen, dass dieses raumbildend wird. Partikel bieten eine geeignete Methode, dieses Konzept zu simulieren.

Durch die Assoziation mit Haaren entsteht noch mehr der Bezug zu der Struktur als lebender Organismus.

Folgeseite:
Abb.2.5.7-1 Rendering

optisch	transluzent möglich
thermisch	möglich, bei ausreichender Anzahl
akustisch	möglich, bei ausreichender Anzahl
mechanisch	möglich, bei ausreichender Anzahl

Tab.2.5.7-1 Potenzial

Verdichten
Größenänderung
einzeln, addiert, umgekehrt

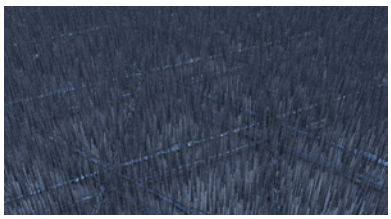
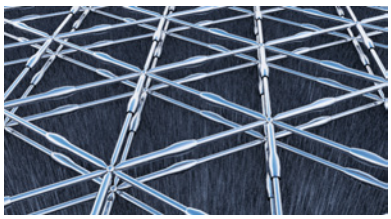




Abb.2.5.7-2 a,b,c Ausgangsposition



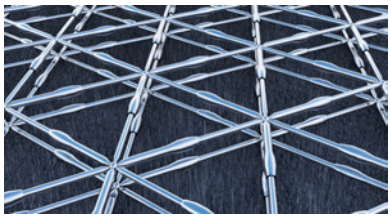


Abb.2.5.7-3 a,b,c 20 %

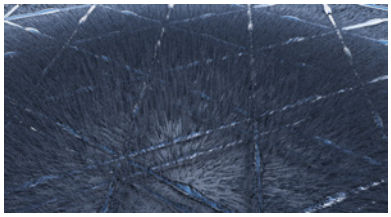
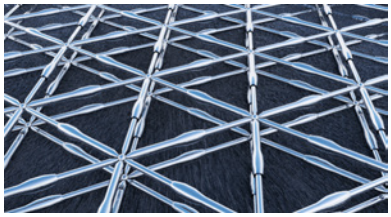




Abb.2.5.7-4 a,b,c 40 %

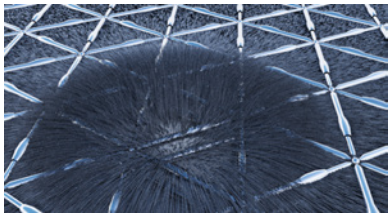
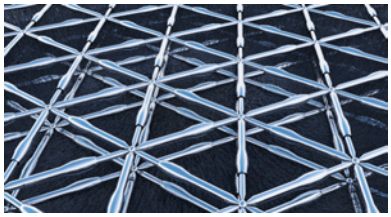




Abb.2.5.7-5 a,b,c 60 %

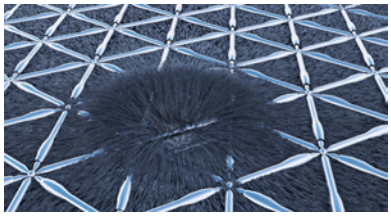
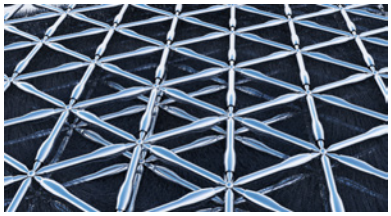




Abb.2.5.7-6 a,b,c 80 %

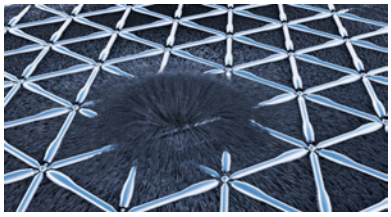
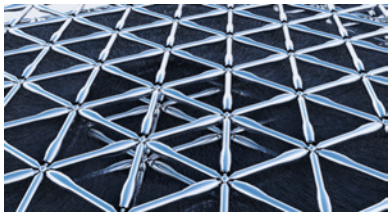
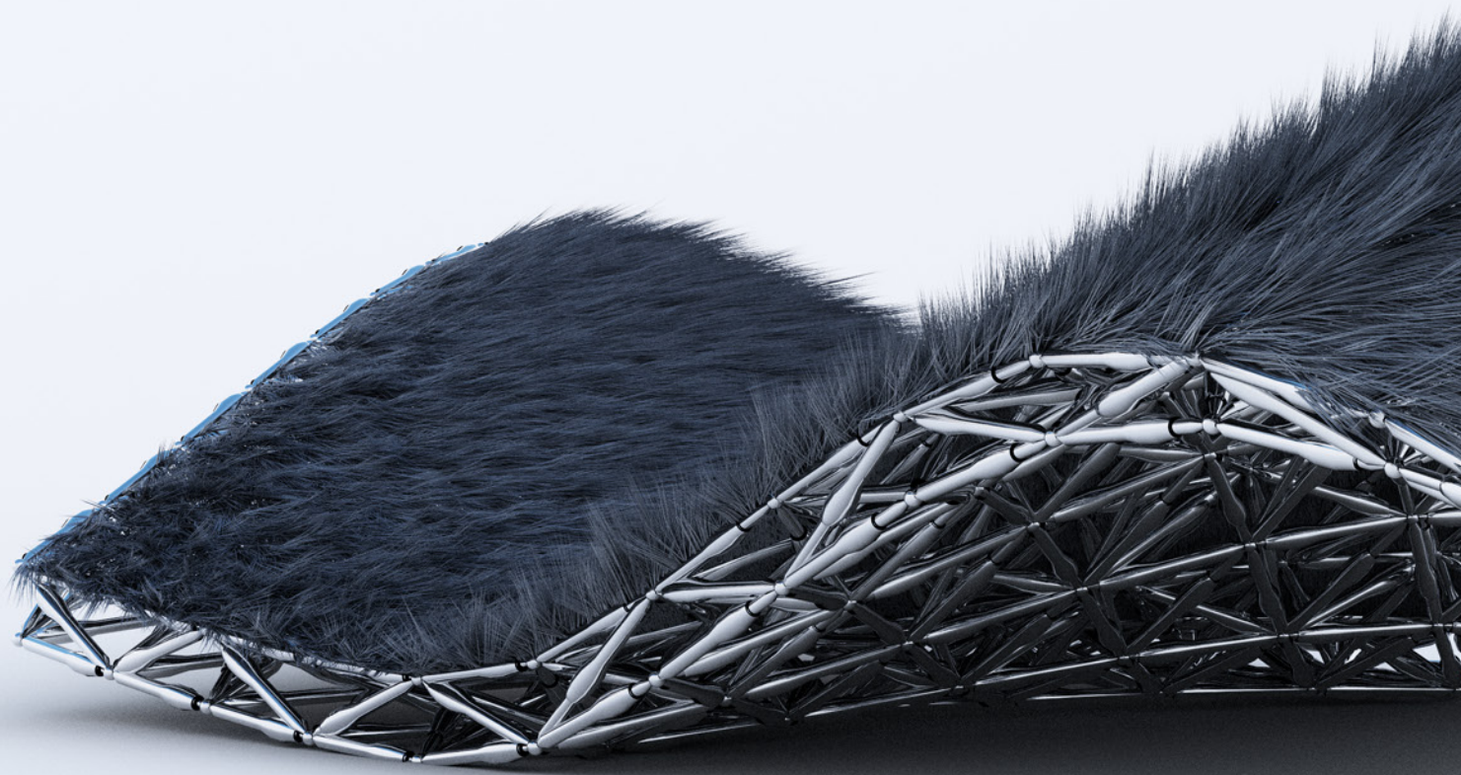
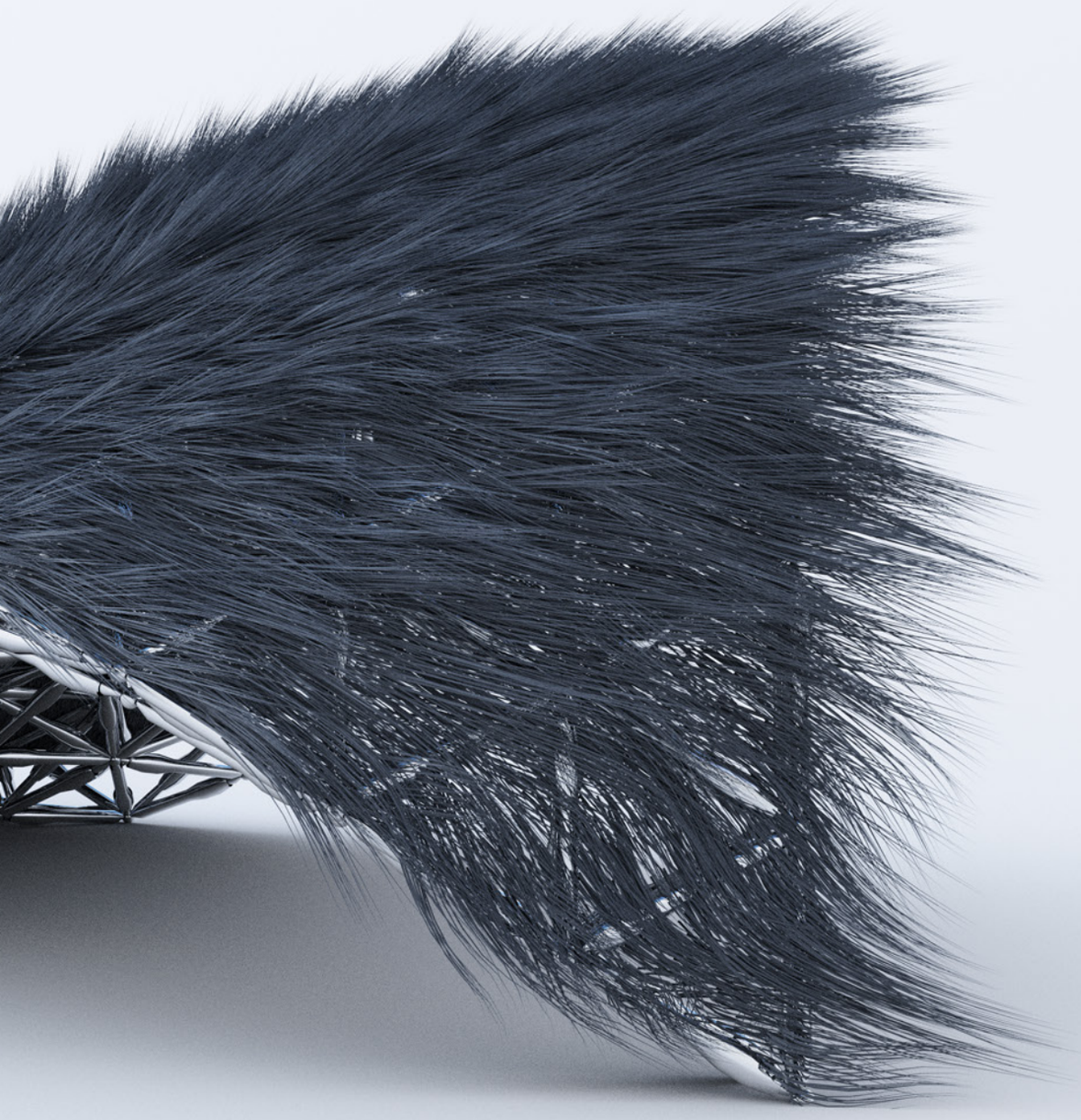




Abb.2.5.7-7 a,b,c Endposition





2.6 KNOTENPUNKTE UND KERN

Wesentliches Kriterium in der Entwicklung von Stabwerken ist die Verbindung der einzelnen Stäben miteinander. Die Frage, ob die Knotenpunkte biegesteif oder beweglich ausgeführt werden, ist von zentraler Bedeutung. Bei gelenkigen Knoten ist zudem eine Einschränkung der Freiheitsgrade zu erörtern.

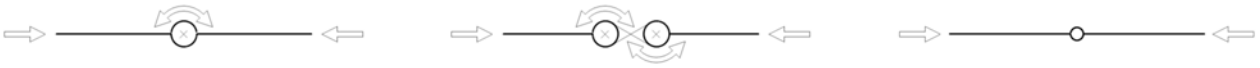
In der Regel ist es nicht mölgich, Kräfte auf einen Punkt zu konzentrieren. Bei biegesteifen Gelenken führt das zu Momenten innerhalb des Knotens. Bei gelenkigen Varianten führt das zum Verdrehen der einzelnen Gelenke, wie in der ersten Spalte der Visualisierung dargestellt.

Die Zweite Spalte zeigt einen Lösungsansatz durch Umlenken der Kräfte mittels Zahnräder. Die Verbindung der beiden Stäbe mittels Zahnräder erlaubt, dass diese sich jeweils nur in die gleiche Richtung verformen können. Der Winkel, den diese dabei einnehmen, ergibt sich aus der vorausgegangen Diskussion zur Krümmung im Abschnitt Tragwerkshöhe. Diese Variante wird auf der Folgeseite noch genauer beschrieben.

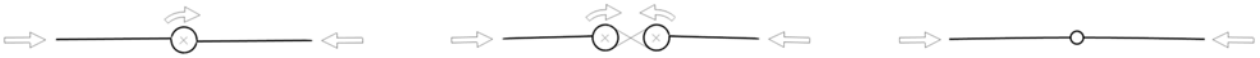
Verweis auf:
Krümmung, Seite 56

Denkbar wäre, elastische Stäbe einzusetzen, wie das bereits bei zahlreichen Gitterschalen angewendet wurde und in der dritten Spalte schematisch angedeutet ist. Bei beweglichen Konstruktionen ist jedoch zu berücksichtigen, dass die Kräfte für deren Verformung zusätzlichen Energie-Aufwand erfordern.

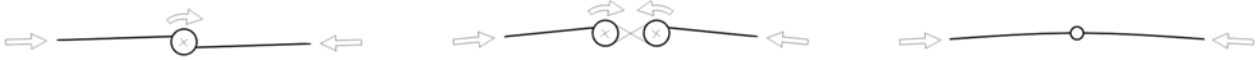
Analyse Knotenpunkt
Problemstellung durch Verdrehen bei Krafteinwirkung
Lösung durch Umkehrung und Durchlaufwirkung



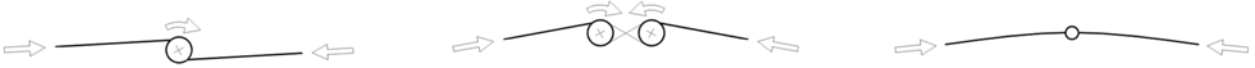
Diagr.2.6.0-1 a,b,c Ausgangsposition



Diagr.2.6.0-2 a,b,c 20 %



Diagr.2.6.0-3 a,b,c 40 %



Diagr.2.6.0-4 a,b,c 60 %

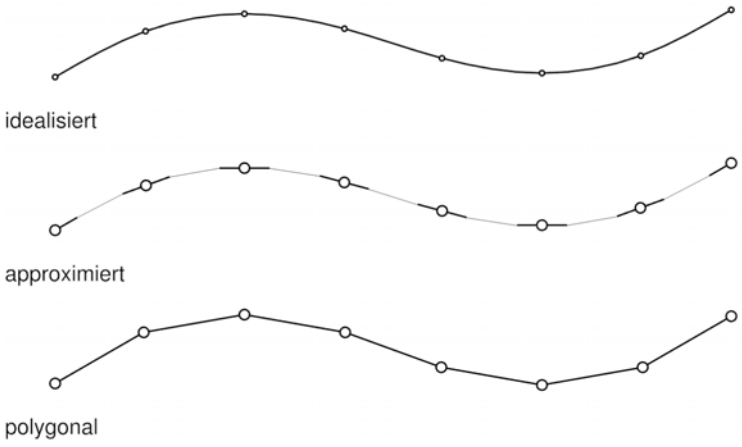
Diagr.2.6.0-5 a,b,c 80 %

Diagr.2.6.0-6 a,b,c Endposition

2.6.1 LÖSUNG

Die Varianten zeigen zwei Möglichkeiten, Kräfte innerhalb eines Knotenpunktes mittels Zahnräder umzulenken. Architektonisches Potenzial dieser Varianten ist eine Idealisierung der Geometrie. Gesetz den Fall, dass eine möglichst glatte Oberfläche gewünscht ist, führt die Ausrichtung der Knoten als Mittelwert der Gelenke einen kleinteiligeren Polygonzug aus, wie im Diagramm unten beschrieben.

Eine mechanisch anmutende Erscheinung entsteht. Diese Steam-Punk-Optik kann durchaus ein gewollter Weg sein, ein Architekturprojekt zu artikulieren.



Diagr.2.6.1-1 Annäherung an die Idealform

Gegenüberstellung Knotenpunkte
Lösungsansatz Keilriemen
Vergleich von einfach und doppelt

Abb.2.6.1-1 Variante 1

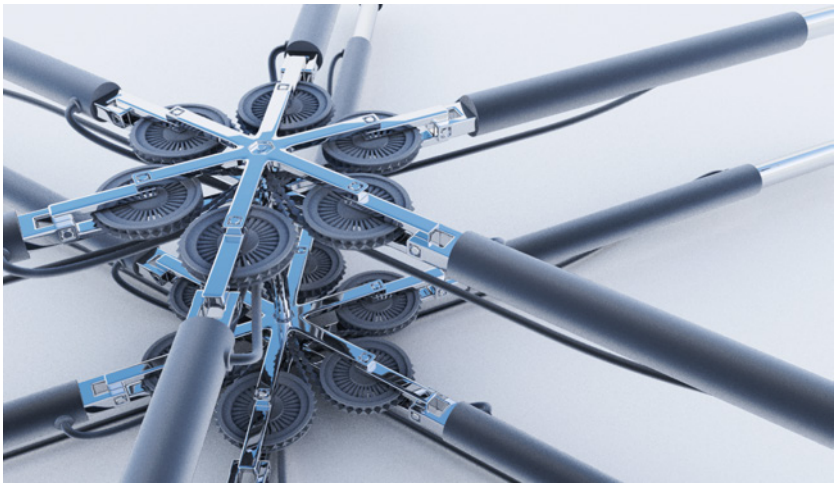


Abb.2.6.1-2 Variante 2

2.6.2 OPTIMIERUNG

Ein klassischer Ansatz im Bezug auf die Formfindung ist, eine Geometrie zu entwickeln, mittels Computersimulation Kräfte anzusetzen und die Tragfähig zu überprüfen. Methoden der Topologie-Optimierung erlauben es, diesen Prozess umzudrehen. Gegeben sind dabei lediglich die maximalen Ausmaße der Geometrie. Es werden Kräfte angesetzt. Bei genauer Betrachtung der Ergebnisse ist zu erkennen, dass die Geometrie in kleine Zellen zerlegt wird. Für jede Zelle wird die Belastung ermittelt. Jene mit höherer Tragfähigkeit bleiben erhalten. Jene mit geringer Tragfähigkeit werden entfernt. Dieser Formfindungsprozess, weist gewisse Parallelen zu Kettenmodellen und hängenden Gitterschalen auf. Auch hier wird der Weg sprichwörtlich umgekehrt. Analog zu diesen Methoden dient die Topologie-Optimierung vorerst der Inspiration für neue Bauteilformen. Selbstverständlich muss die ausgegebenen Geometrie auch in Bezug zu anderen Entwurfsparametern gesetzt werden.

Verwendung fand das Programm BESO2D von Zhihao Zuo. Der Link findet sich im Abschnitt Zusatz. Nähere Informationen sind zudem im links genannten Buch zu finden.

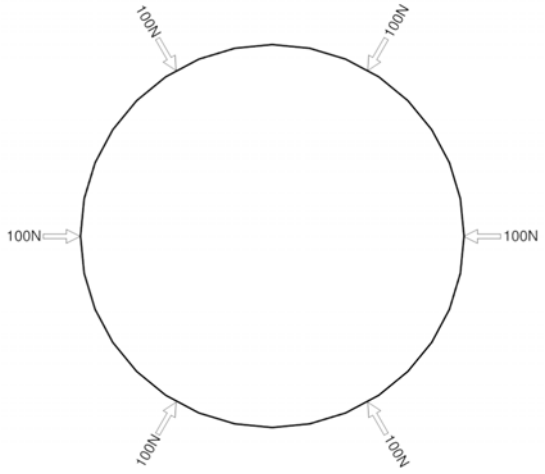
Verweis auf:
BESO2D, Seite 275

Xiaodong Huang und Mike Xie, „Evolutionary Topology Optimization of Continuum Structures: Methods and Applications“, 2010

Auf der Folgeseite wird eine Auswahl von Entwürfen für eine mögliche Ausformulierung des Kerns aufgezeigt. Angesetzt wurden stellvertretend 100 N. Da es sich um eine Topologie-Optimierung handelt, ist die Skalierung der Kräfte nicht ausschlaggebend. Von zentraler Rolle ist die Reduktion in Prozent, die erfolgen soll. Im Beispiel sind das 15 %. Mit anderen Worten formuliert: Es ist selbstverständlich ein Unterschied, ob auf einer Seite des Bauraums 100 N und auf der anderen 50 N angesetzt werden. Es wäre allerdings kein Unterschied, wenn auf beiden Seiten 1000 N, in einem weiteren Beispiel 5.000 N angegeben werden würden.

Folgeseite:
Abb.2.6.2-1 Rendering

Folgeseite danach:
Abb.2.6.2-2 Rendering



Diagr.2.6.2-1 Krafteinwirkung

Topologie-Optimierung eines Knoten mittels BESO2D
Angewendet an einem abstrahierten Knotenpunkt
in Grundriss mit Zug und Druck, sowie Schnitt

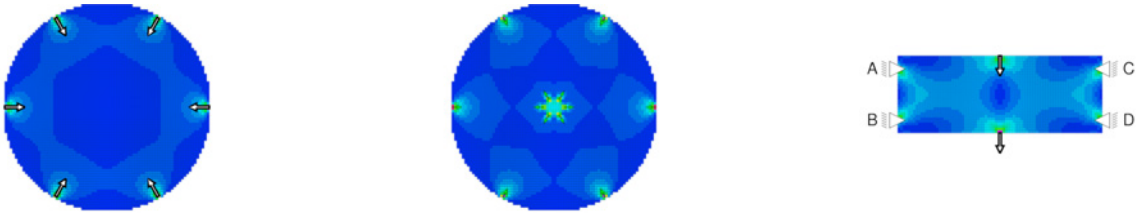


Abb.2.6.2-3 a,b,c Fea

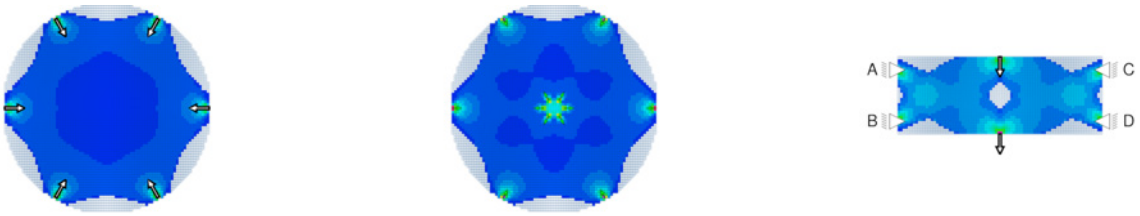


Abb.2.6.2-4 a,b,c 85 %

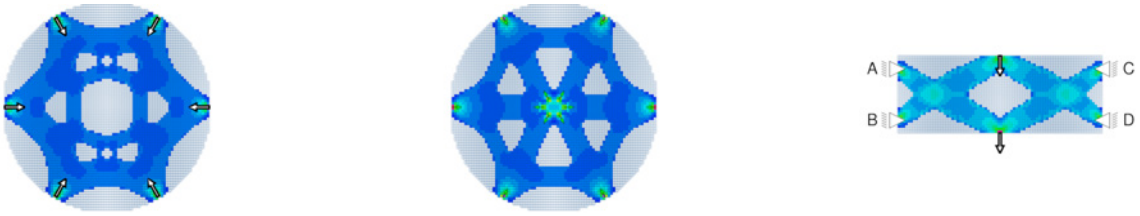


Abb.2.6.2-5 a,b,c 60 %

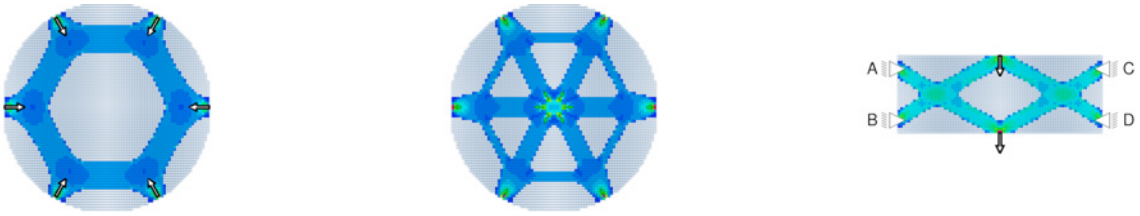


Abb.2.6.2-6 a,b,c 45 %

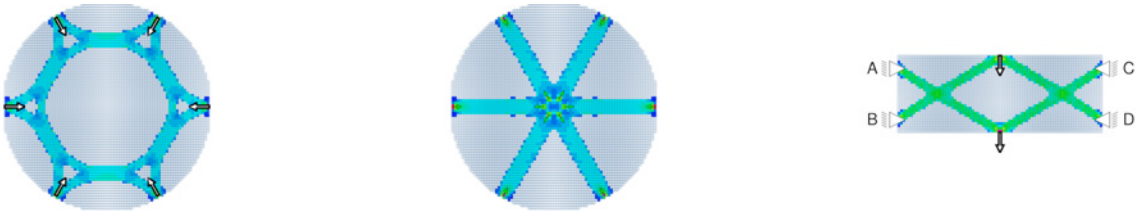


Abb.2.6.2-7 a,b,c 30 %

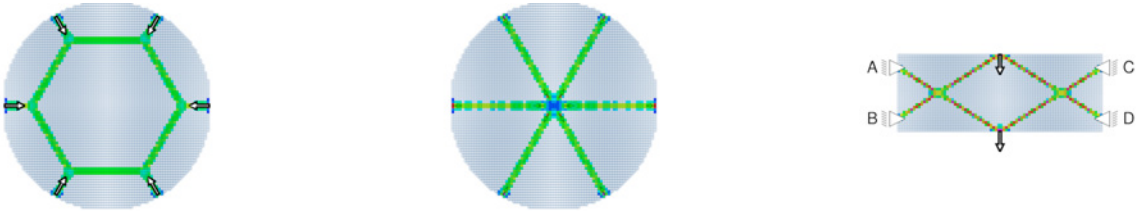
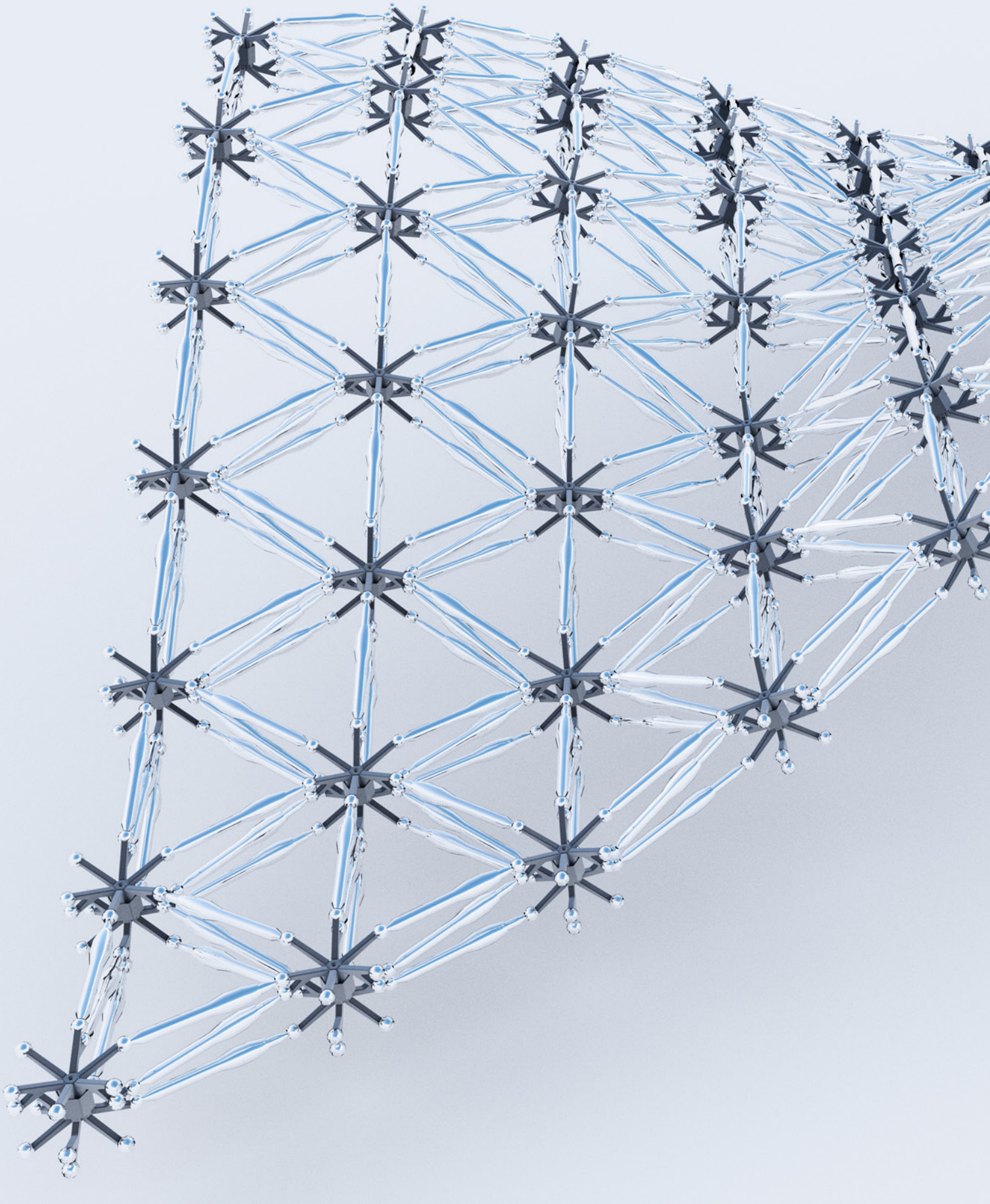
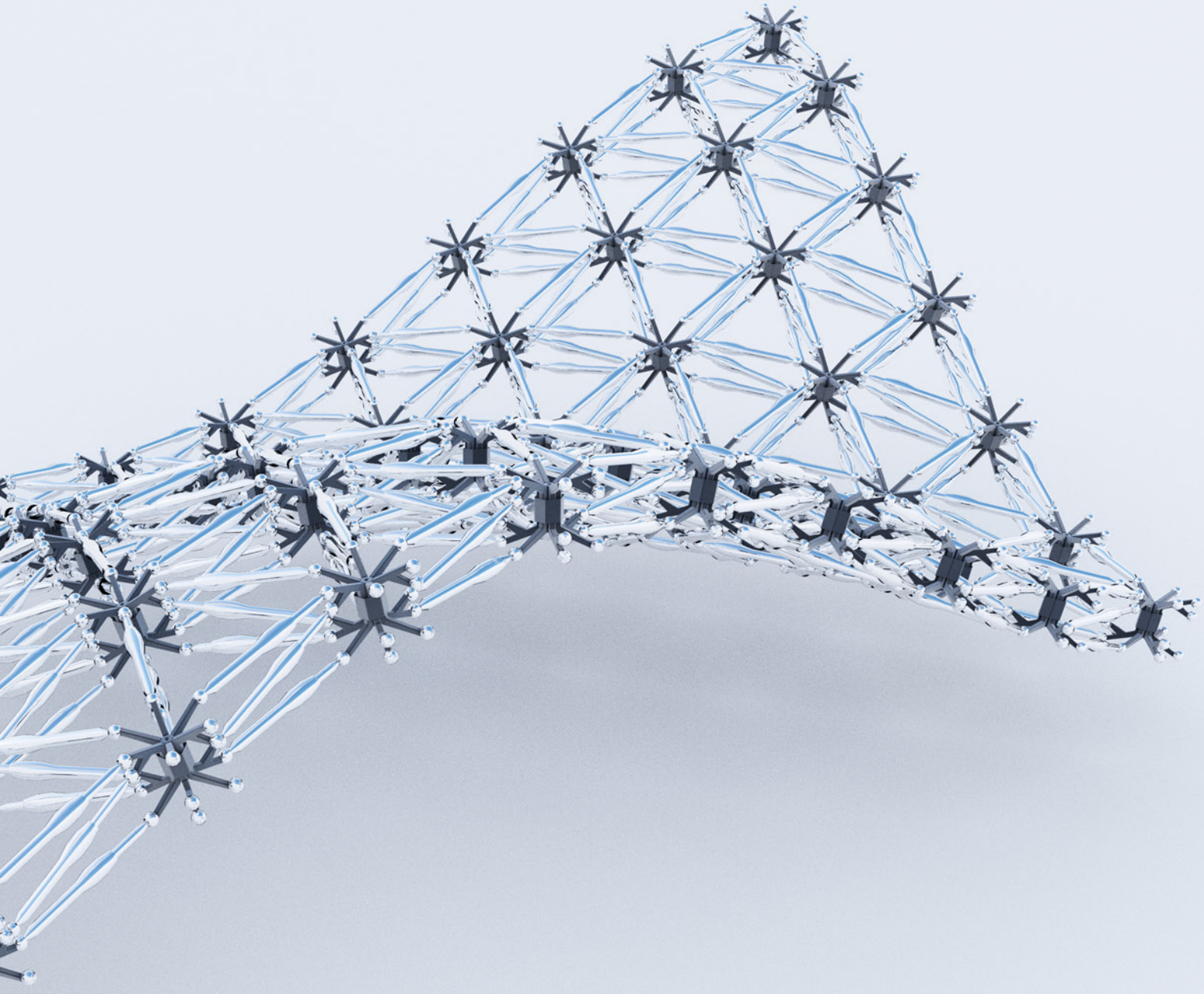


Abb.2.6.2-8 a,b,c 15 %





William Hunter, „Predominantly solid-void three-dimensional topology optimization using open source software“, 2009

Verweis auf:
ToPy, Seite 275

Verweis auf:
ParaView, Seite 275

Folgeseite:
Abb.2.6.2-9 Rendering

Selbstverständlich, ist diese Methode auch im dreidimensionalen Raum anwendbar. Verwendet wurde dazu die kostenlose Bibliothek ToPy für Python von William Hunter.

Die Ergebnisse zeigen die Optimierung eines Stabes. Die Anzahl der Zellen beträgt bei diesem 625.000 Zellen. Das ergibt insgesamt 5.000.000 Punkte. Wie zu erwarten ist die Zeit zur Berechnung entsprechend hoch: 1.303 Minuten in Summe für 50 Iterationen. Es ergibt sich ein Mittel von 26 Minuten pro Iteration.

Topologie-Optimierung eines Trägers
Angewendet an einem abstrahierten Stab
in drei Entwicklungsschritten

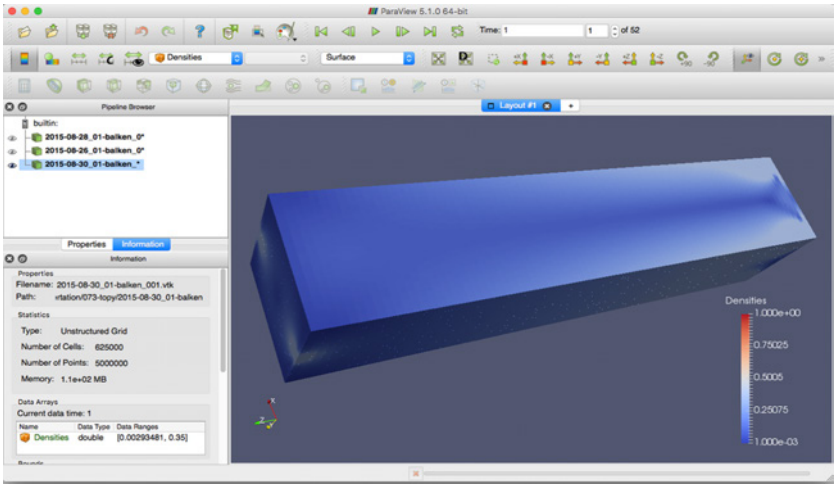


Abb.2.6.2-10 Ausgangsgeometrie

Anzahl der Zellen	625.000
Anzahl der Punkte	5.000.000
Arbeitsspeicher	100 MB

Tab.2.6.2-1 Ausgangsgeometrie

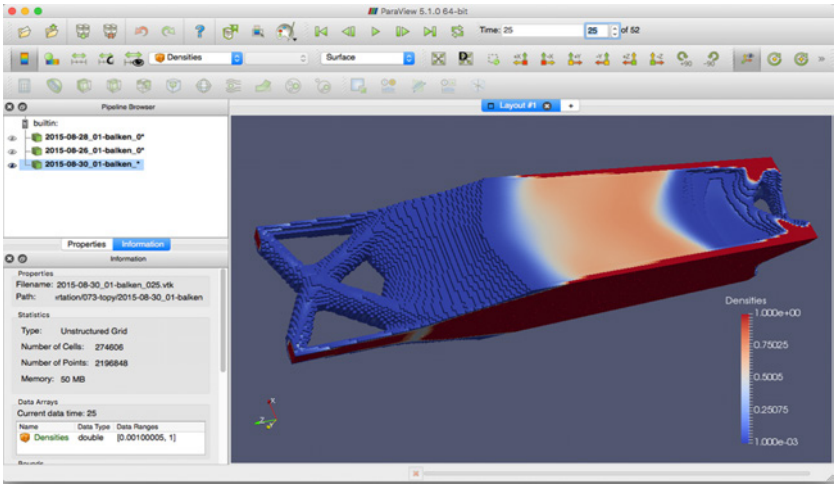


Abb.2.6.2-11 Zwischenschritt

Anzahl der Zellen	274.606
Anzahl der Punkte	2.196.848
Arbeitsspeicher	50 MB

Tab.2.6.2-2 Zwischenschritt

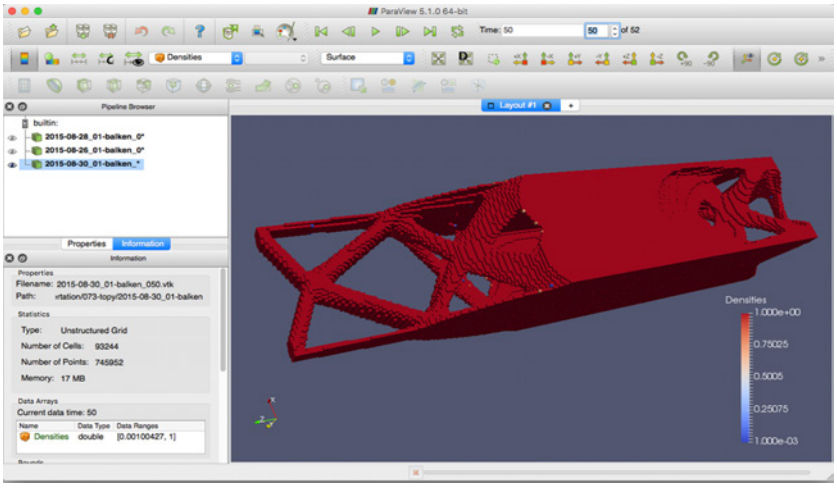
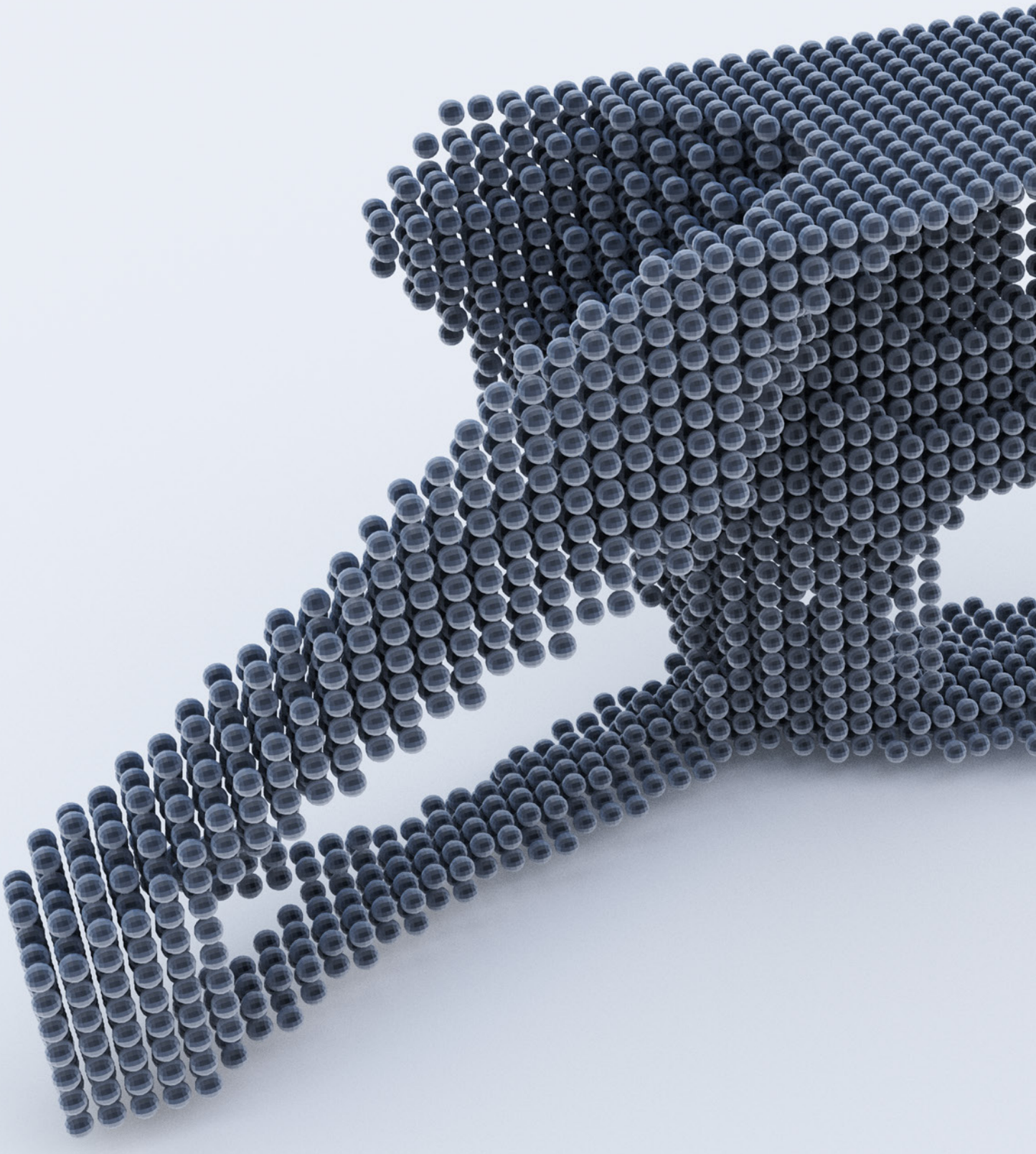
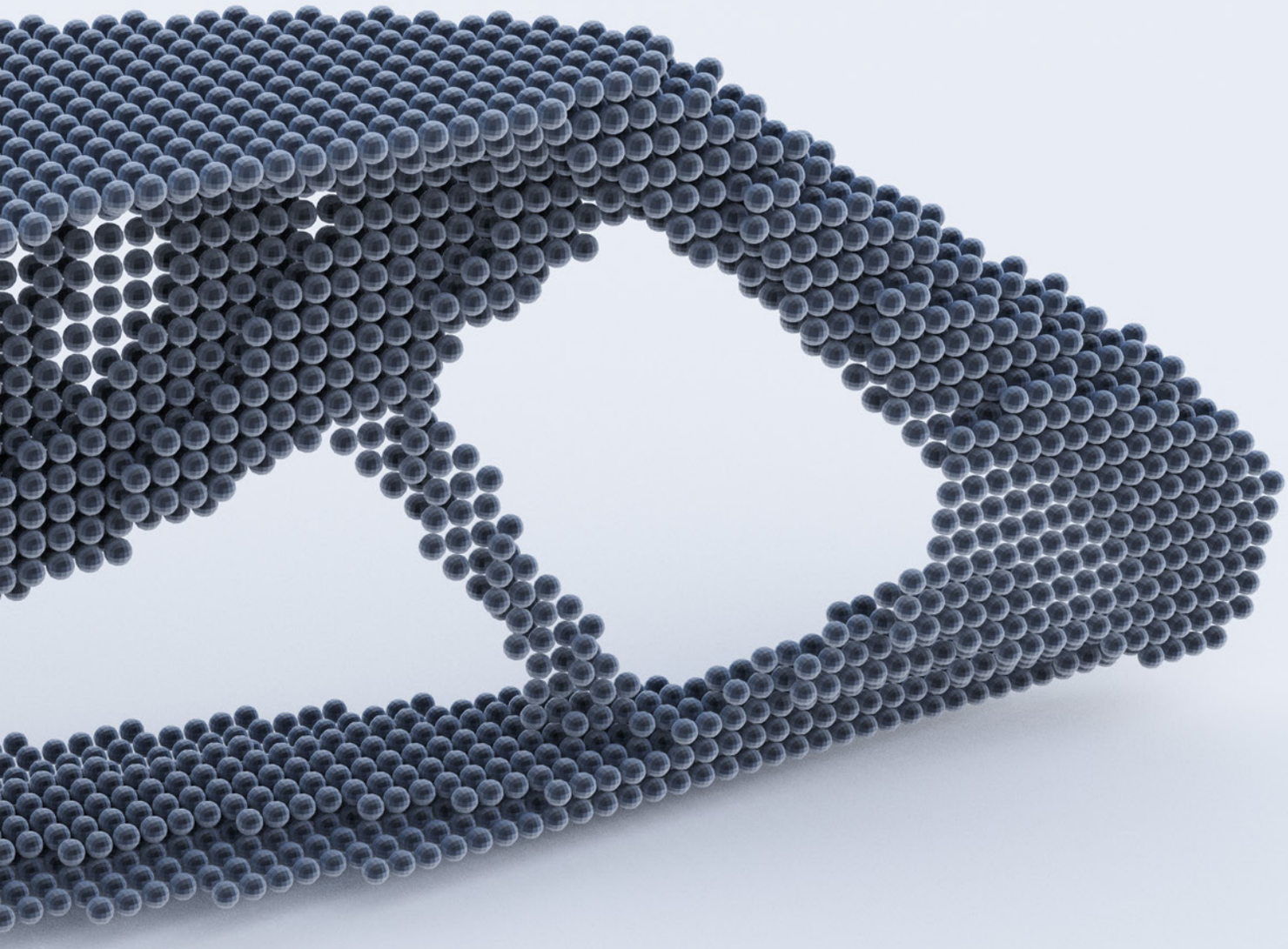


Abb.2.6.2-12 Finale Geometrie

Anzahl der Zellen	93.244
Anzahl der Punkte	745.952
Arbeitsspeicher	17 MB

Tab.2.6.2-3 Finale Geometrie





2.6.3 PROTOTYP 2

Die Erkenntnisse aus den analogen Modellen führen zu einem weiteren Ansatz neben den Keilriemen: das Zerlegen der Bewegung in kleinere Teilschritte. Wie in der Simulation zu sehen, werden die Knoten in mehreren Schritten in ihren Freiheitsgraden eingeschränkt. Die Aufhängung der Motoren selbst erlaubt in diesem Beispiel lediglich eine Bewegung auf der Y-Achse. Die Verbindung zwischen Motor und Knoten erlaubt wiederum nur eine Verdrehung auf der Z-Achse. Die Führung der Motoren und die Schubstange erlaubt ein Verdrehen auf der X-Achse.

Das gesamte Element ist mittels einer Fläche mit vier Knotenpunkten steuerbar. Auf der Folgeseite zu sehen ist die Aggregation von 9 dieser Flächen zu Prototyp 2, einem kinematischen Dreieck aus insgesamt 27 Motoren.

Folgeseite:
Abb.2.6.3-1 Modellfoto Prototyp 2

Bewegungssimulation
Kinematische Fläche
zerlegt in separate Freiheitsgrade

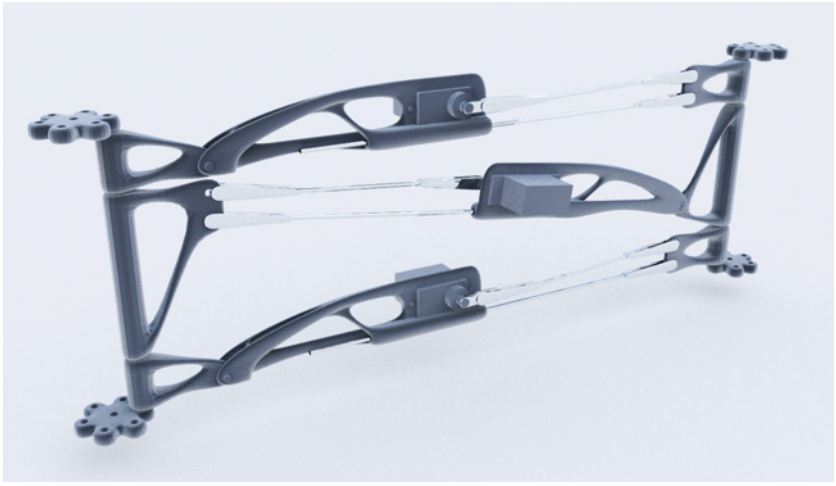


Abb.2.6.3-2 Ausgangsposition

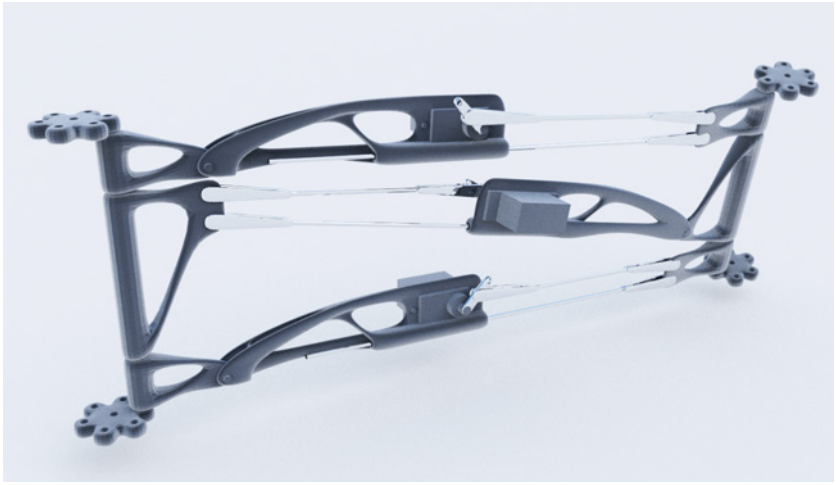


Abb.2.6.3-3 Zwischenschritt

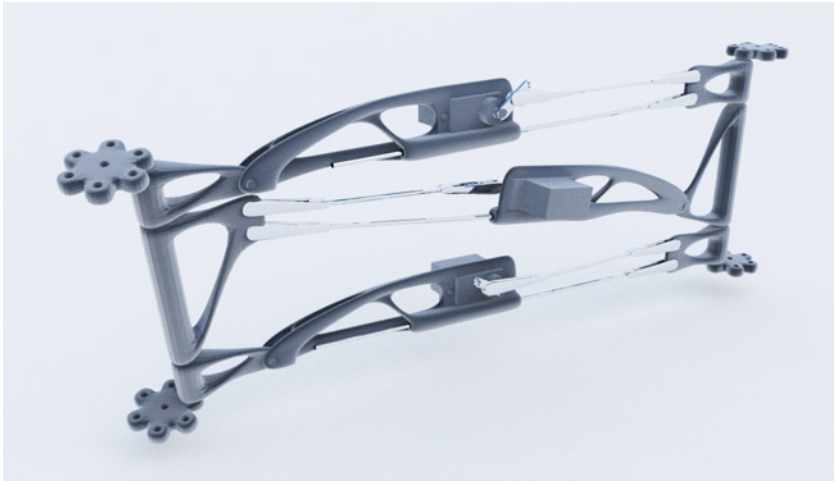
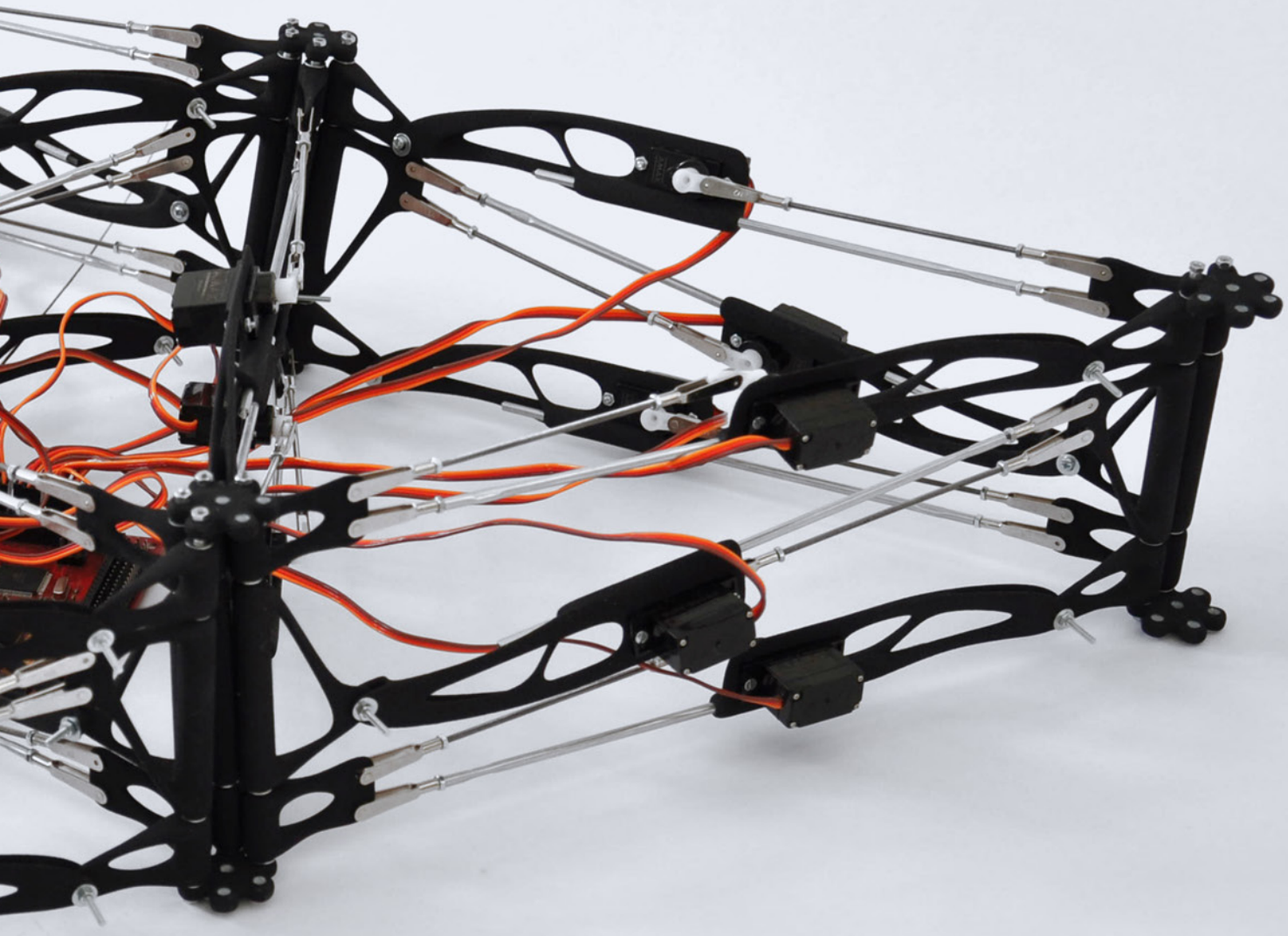


Abb.2.6.3-4 Endposition



2.7 AUTOMATION

Mark Elling Rosheim, „Leonardo's Lost Robots“, 2006

Bereits um 1495 entwickelte Leonardo da Vinci einen automatisierten Ritter aus Metall. Er war in der Lage, eigenständig zu gehen und weitere menschenähnliche Bewegungen auszuführen, so schreibt Mark Elling Rosheim. Was war die Motivation für den Erbauer Leonardo da Vinci? Was übt eine derartige Magie auf den Menschen aus, autonom funktionierende Maschinen zu konzipieren?

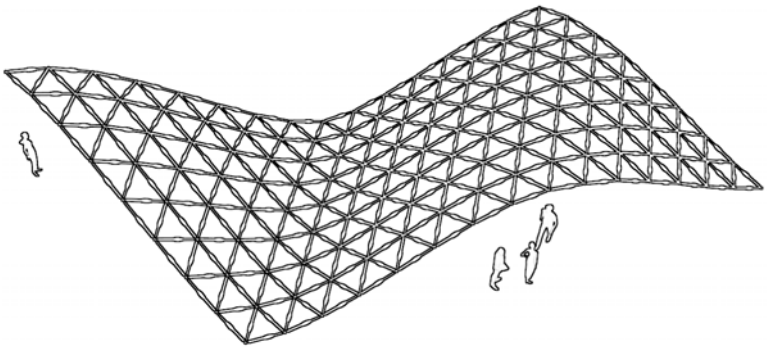
2.7.1 ÜBERGEORDNET

Dieser Abschnitt beschäftigt sich mit der Reaktion der Struktur auf seine Umgebung. Die drei Einflussfaktoren Mensch, Tragfähigkeit des Bodens und Schallquellen sind dabei stellvertretend. Von zentraler Bedeutung ist, wie die Geometrie auf Zielkonflikte zwischen den Anforderungen antworten kann.

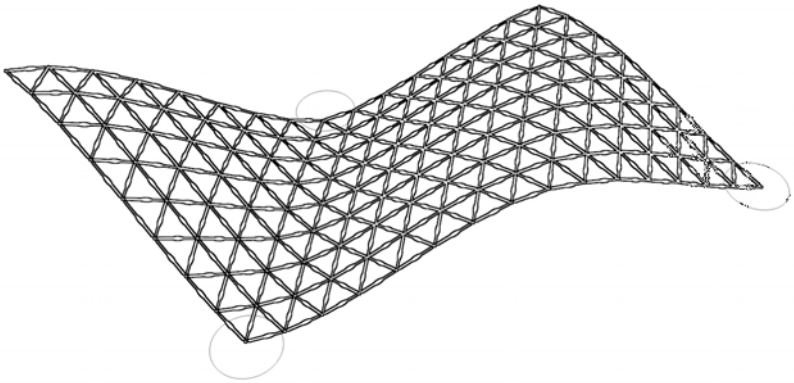
Bekannte Themenstellungen sind auch Windrichtung und Sonneneinfall. Diese sind unmittelbar mit der Qualität des erzeugten Raumes gekoppelt. Der Abschnitt Reaktion im Kapitel Referenzmodell wird sich ausgiebiger mit diesen Themen beschäftigen.

Verweis auf:
Reaktion, Seite 174

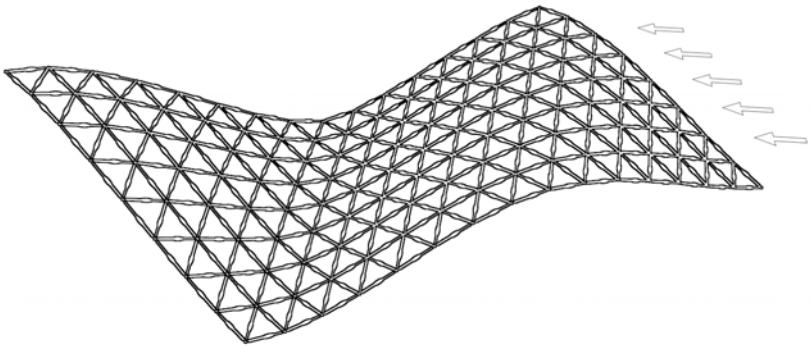
Reaktion der Fläche
auf gegebene Umstände
Kompromiss aus drei gewählten Einflussfaktoren



Diagr.2.7.1-1 Reaktion auf Menschenmengen



Diagr.2.7.1-2 Reaktion auf Tragfähigkeit des Bodens



Diagr.2.7.1-3 Reaktion auf Schallwellen

2.7.2 D E T A I L

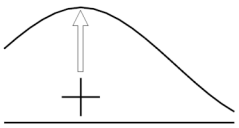
„Die Kapsel ist eine Cyborg-Architektur. Der Mensch, die Maschine und der Raum überwinden ihre Inkongruenz und schaffen zusammen einen neuen Organismus. Wie die Menschen mit künstlichen Organen eine neue Ordnung herstellen, die weder menschlich noch maschinell ist, wird auch die Kapsel, eine Architektur, die die Menschen sowie den Mechanismus übertrifft, zukünftig immer mehr mechanisiert werden. Dieser präzise Mechanismus ist kein Werkzeug, sondern ein Teil des Lebenssystems, der selbst den Zweck des Daseins darstellt.“

Noriaki Kurokawa, „Capsule Declaration“, 1969

Ein besonderer Fokus liegt auf der Interaktion der Struktur mit Menschen. Bei Abwesenheit von Nutzern existiert kein Raum. Erst wenn sich eine Person auf die Struktur zubewegt, spannt sich Raum auf. Eine einzigartige Wechselwirkung zwischen Mensch und Architektur entsteht.

Vorausgesetzt ist, dass die Struktur in der Lage ist, den Menschen als solchen zu registrieren. Mittels Dynamic-Paint wird es möglich, dass sich die Geometrie an dieser Stelle, in Bezug auf einen gegebenen Radius, in die Höhe verformt.

Diagr.2.7.2-1 Aktion



Diagr.2.7.2-2 Reaktion

Reaktion auf Einzelperson
Aktion, Messung, Reaktion
Reaktion in Form von Bewegung in Z-Richtung

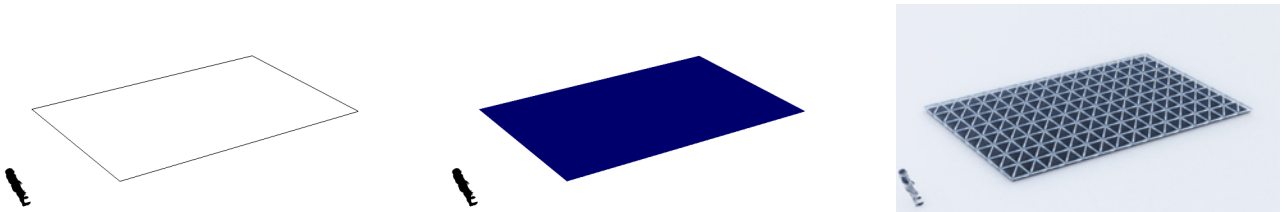


Abb.2.7.2-1 a,b,c Ausgangsposition

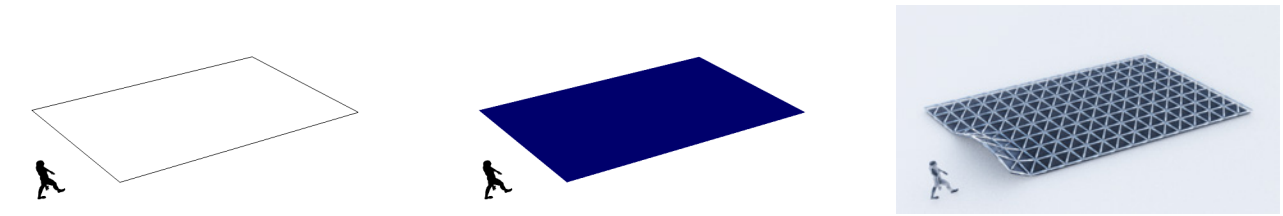


Abb.2.7.2-2 a,b,c 20 %

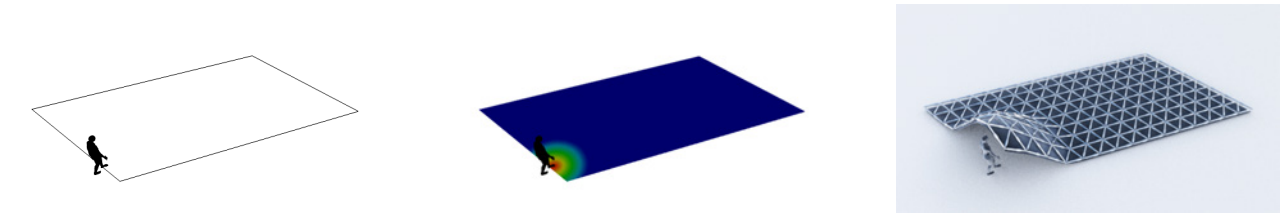


Abb.2.7.2-3 a,b,c 40 %

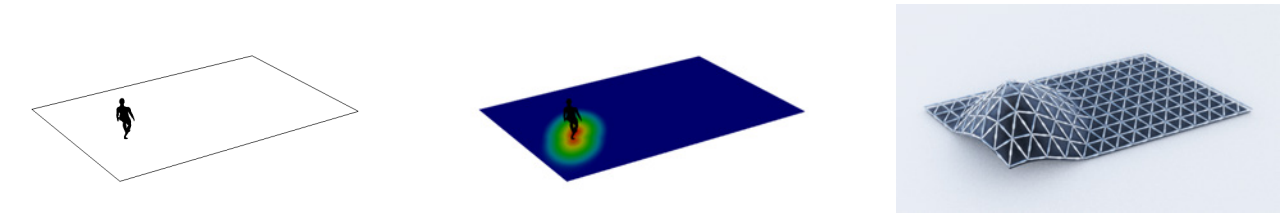


Abb.2.7.2-4 a,b,c 60 %

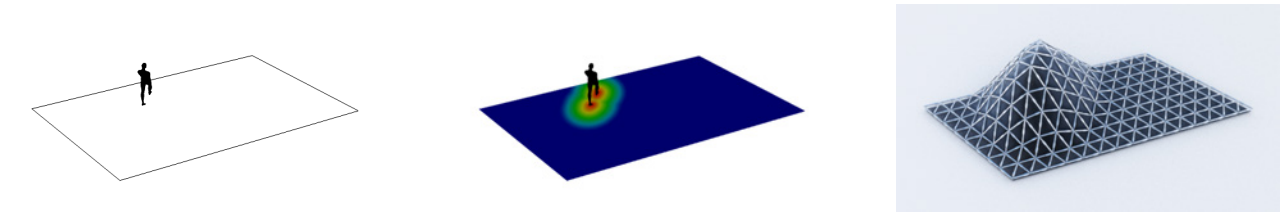


Abb.2.7.2-5 a,b,c 80 %

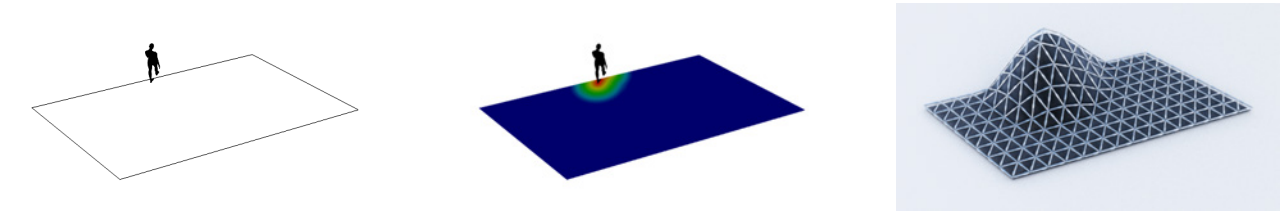


Abb.2.7.2-6 a,b,c Endposition

Das folgende Beispiel zeigt die Reaktion auf eine hohe Anzahl von Menschen. Repräsentiert werden diese in Form von Punkten. Stellvertretend wird hier ein Partikelsystem verwendet, das eine mögliche Bewegung simuliert. Die Fläche reagiert automatisch darauf.

Bereits eine geringe Anzahl von Punkten führt dabei zu einer komplexen Struktur. Die Überlagerung der Einflussbereiche führt zu einer bizarren Ornamentik, die auf abstrakte Art die Interaktion der Punkte reflektiert.

Reaktion auf Personengruppe
Aktion, Messung, Reaktion
Reaktion in Form von Bewegung in Z-Richtung

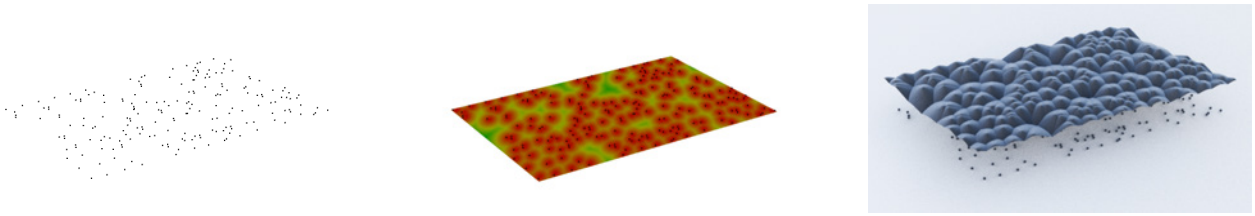


Abb.2.7.2-7 a,b,c Ausgangsposition

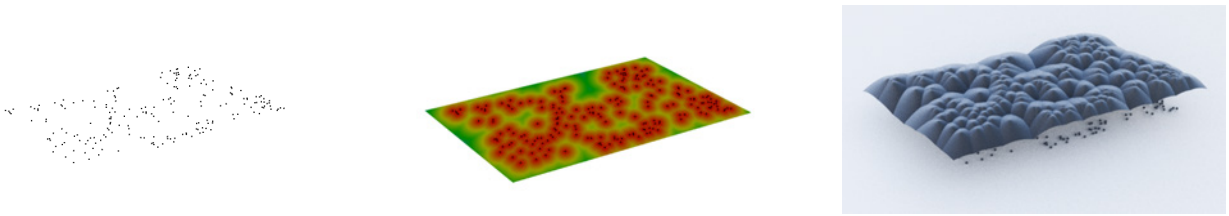


Abb.2.7.2-8 a,b,c 20 %

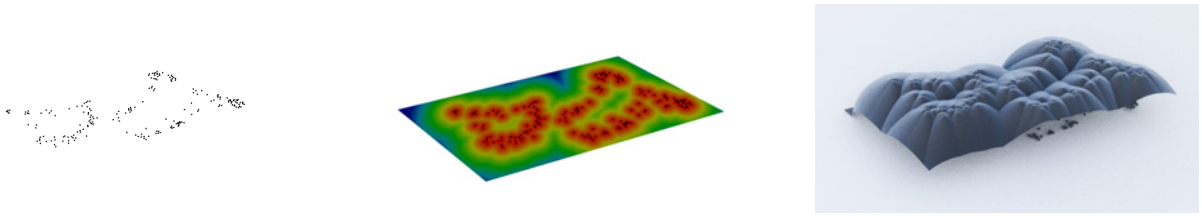


Abb.2.7.2-9 a,b,c 40 %

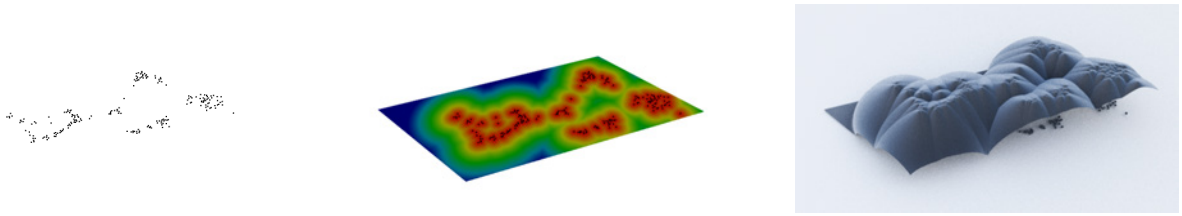


Abb.2.7.2-10 a,b,c 60 %

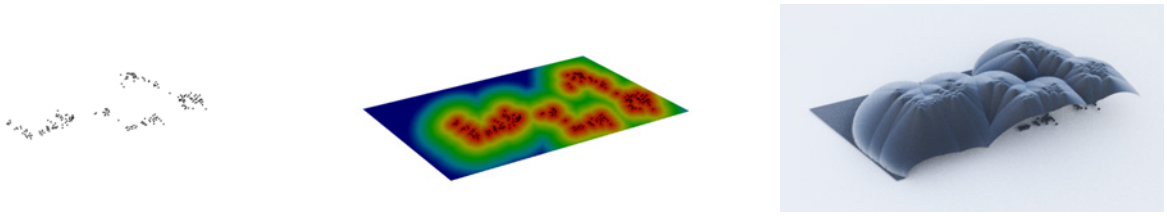


Abb.2.7.2-11 a,b,c 80 %

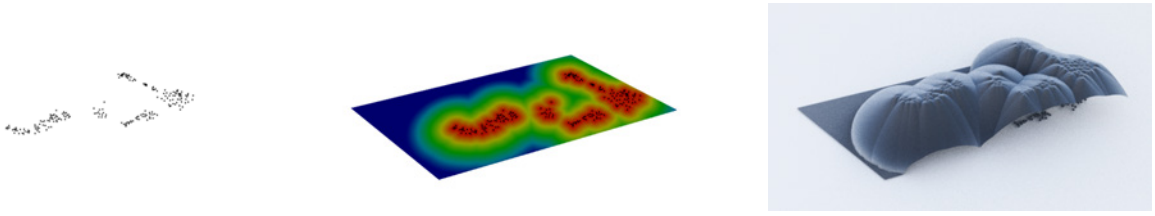
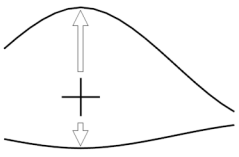


Abb.2.7.2-12 a,b,c Endposition

Ebenso kann auch die Bodenfläche als bewegliches Bauteil ausformuliert werden. Möglicherweise erlauben Sensoren, oder die Verformung der Bodenfläche als Auslöser, eine Erweiterung der Raumhöhe. Der Eindruck, dass der Raum sich um eine Person herum aufspannt, wird verstärkt, wenn auch der Boden eine Reaktion zeigt.

Folgeseite:
Abb.2.7.2-13 Modellfoto

Diagr.2.7.2-3 Aktion



Diagr.2.7.2-4 Reaktion

Reaktion auf Einzelperson
Boden, mit Decke, mit Tragstruktur
Reaktion in Form von Bewegung in Z-Richtung

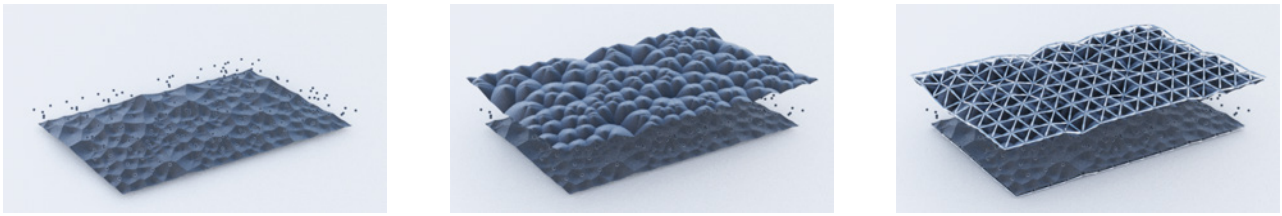


Abb.2.7.2-14 a,b,c Ausgangsposition

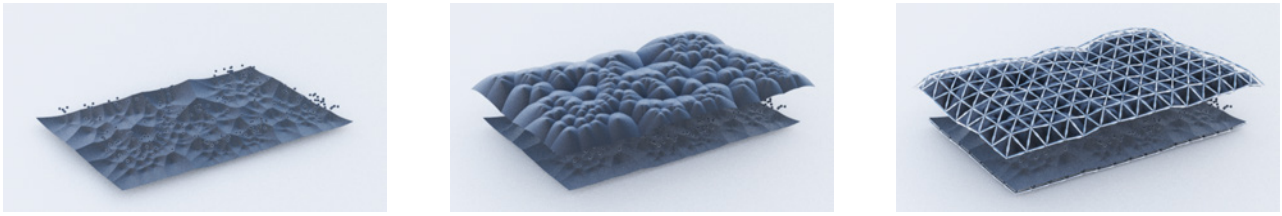


Abb.2.7.2-15 a,b,c 20 %

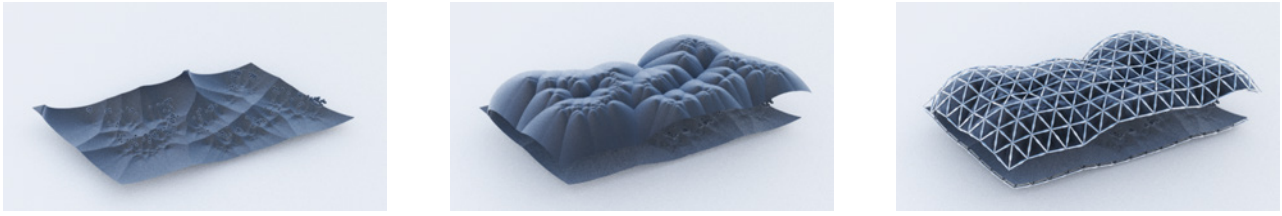


Abb.2.7.2-16 a,b,c 40 %

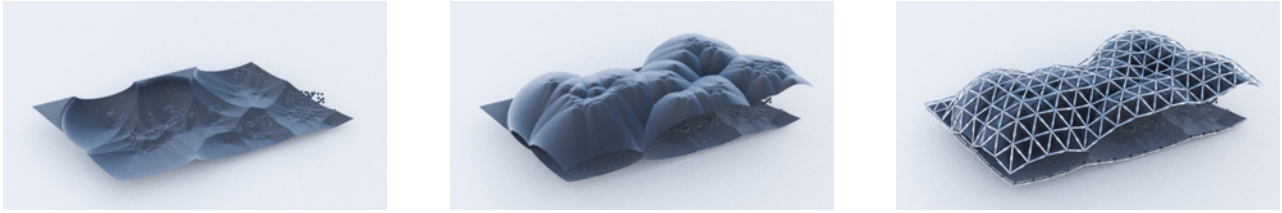


Abb.2.7.2-17 a,b,c 60 %

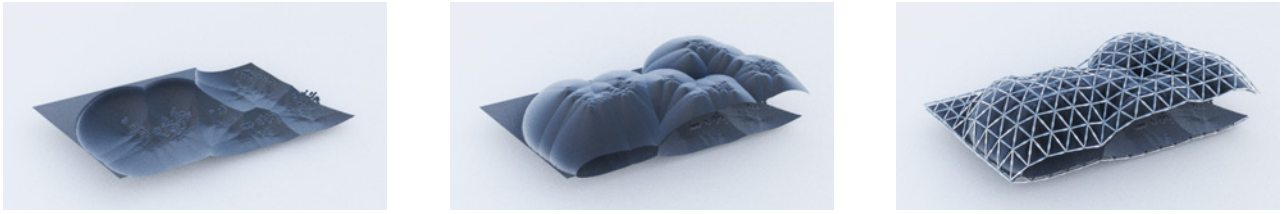


Abb.2.7.2-18 a,b,c 80 %

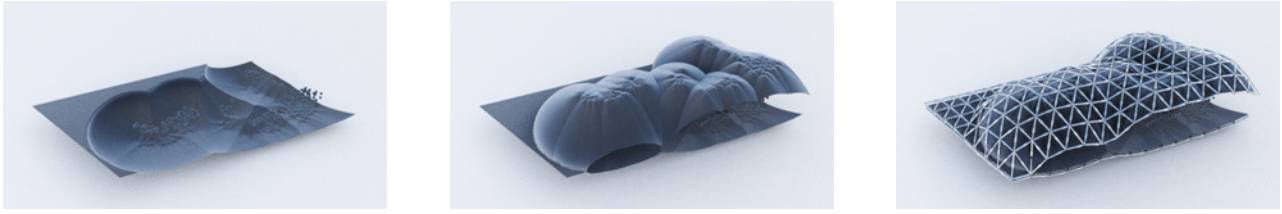


Abb.2.7.2-19 a,b,c Endposition

Dieser Abschnitt zeigt die Studie in den klassischen Architekturdarstellungen Grundriss, Schnitt und Perspektive. Klar erkennbar ist, wie die Struktur auf die variierende Dichte der Menschengruppen reagiert.

Folgeseite:
Abb.2.7.2-20 Rendering

Reaktion auf Einzelperson
Schnitt, Grundriss, Perspektive
Reaktion in Form von Bewegung in Z-Richtung

Abb.2.7.2-21 a,b,c Ausgangsposition

Abb.2.7.2-22 a,b,c 20 %

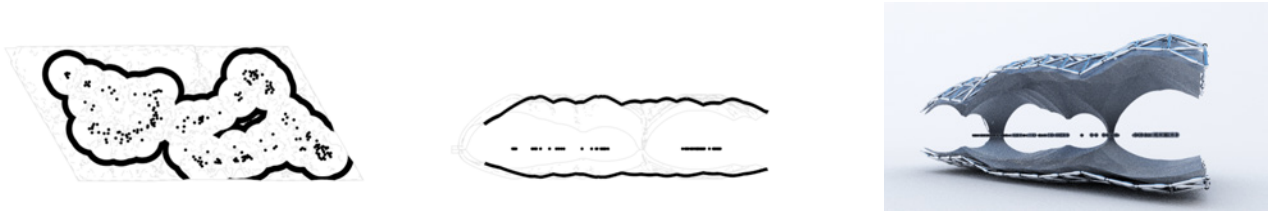


Abb.2.7.2-23 a,b,c 40 %

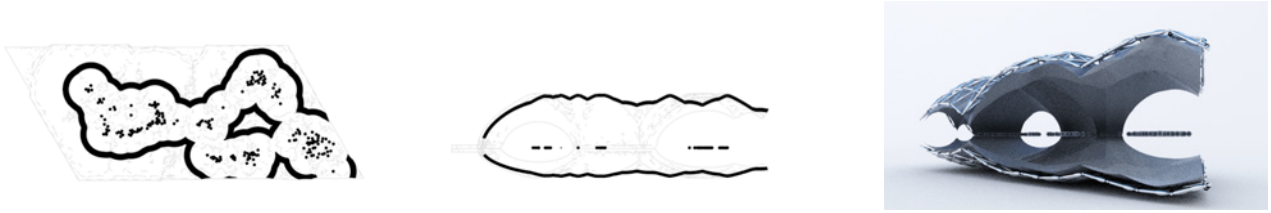


Abb.2.7.2-24 a,b,c 60 %

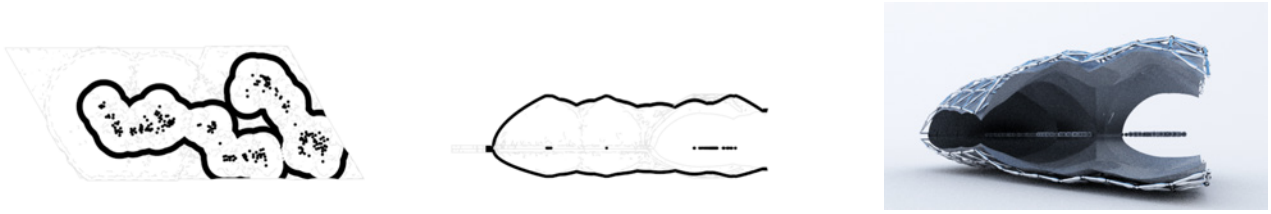


Abb.2.7.2-25 a,b,c 80 %

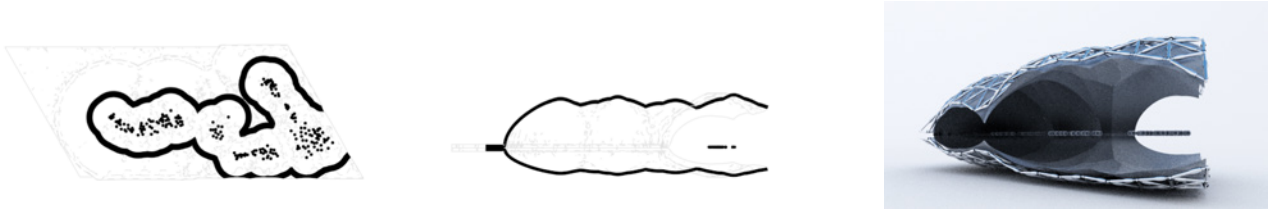
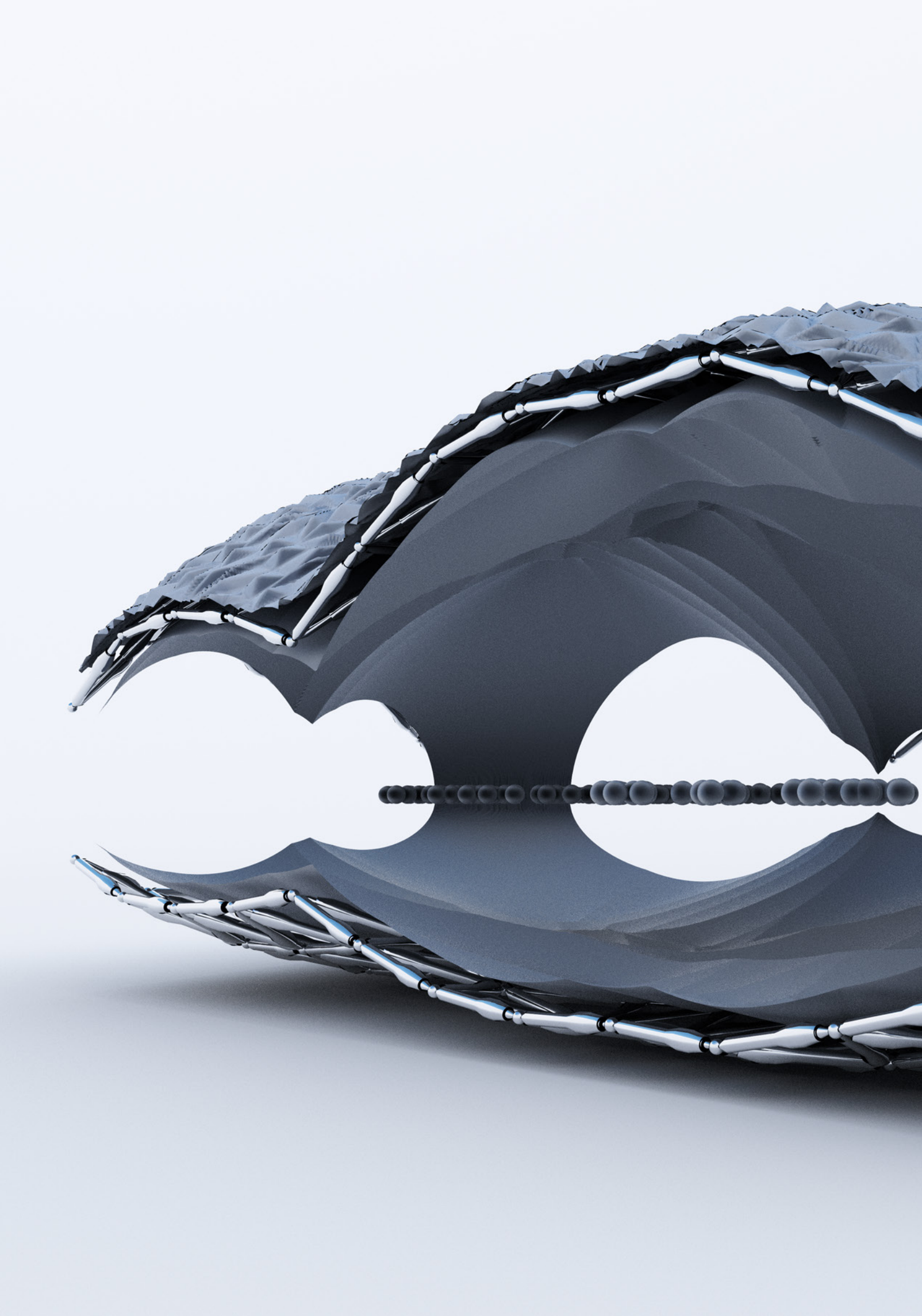
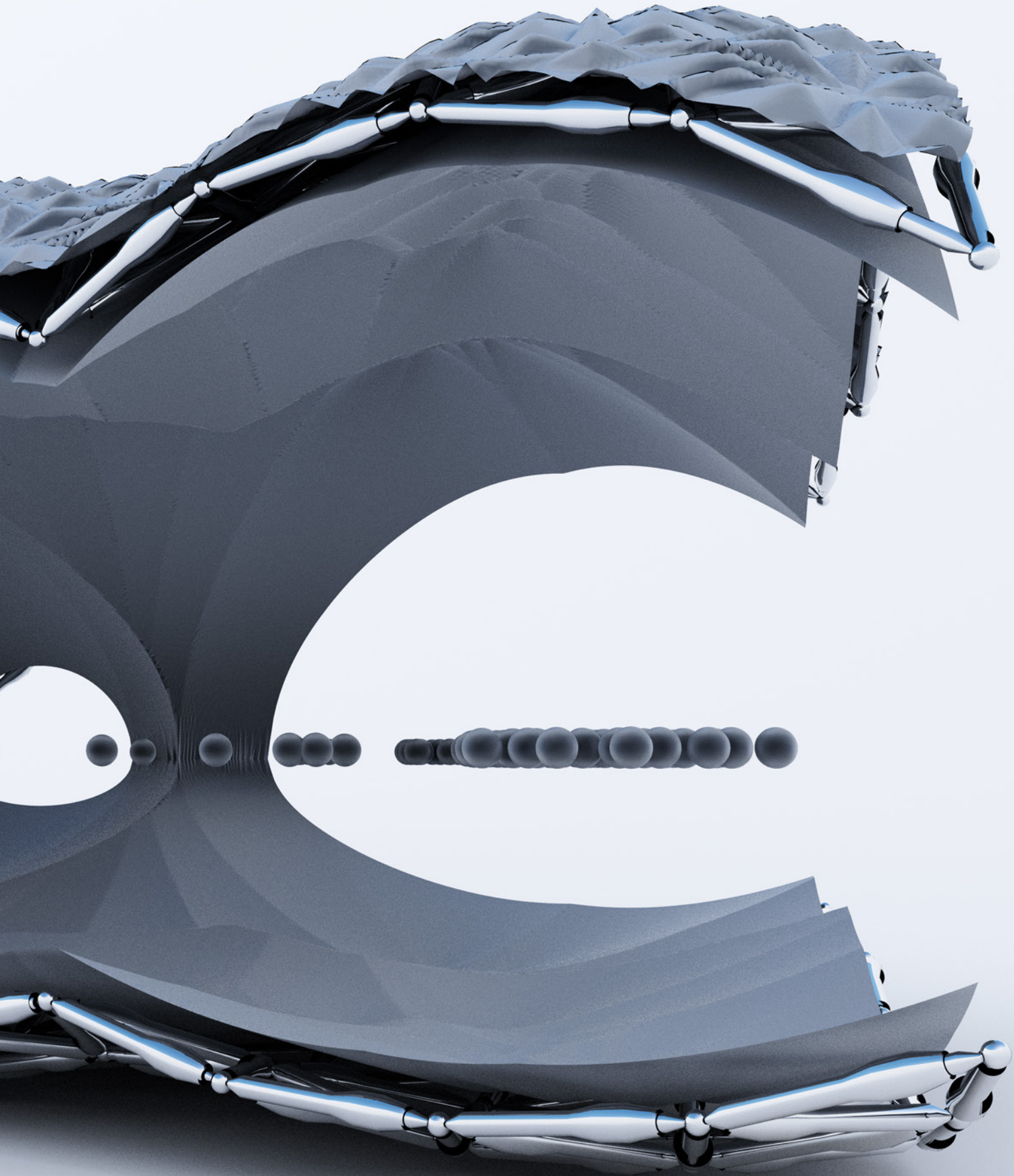


Abb.2.7.2-26 a,b,c Endposition





3 REFERENZMODELL

Das vorgestellte Referenzmodell führt die gewonnen Erkenntnisse zusammen. Darüber hinaus versucht es einen Lösungsansatz für das Schaffen von interaktiver, selbstorganisierender Architektur aufzuzeigen.

„Standing at the crossroads, trying to read the signs, to tell me which way I should go to find the answer“

Eric Clapton, „Let it grow“, 1974

3.1 SELBSTORGANISATION

3.1.1 BYOB

Die Organisation einer Veranstaltung kann hohen Planungsaufwand darstellen. Wie viele Personen werden wann erscheinen? Sind Veganer oder Nichtalkoholiker unter den Gästen? Gibt es Stehtische, oder ist ein Sitzplan vorgesehen? Die Gastronomie hält für die Organisation von Partys eine aufwändige und kostspielige Logistik bereit. Fallen die Begriffe Logistik, aufwändig und kostspielig, könnte geschlussfolgert werden, dass eine Personengruppe sicherlich keine Partys organisiert: Studenten.

Doch Studenten organisieren Partys und das nicht selten. Sogar an Orten, die dafür nicht prädestiniert wären. Wie ist das möglich? Häufig liegt, ausgesprochen oder nicht, ein einfaches Organisationsprinzip zu Grunde: BYOB, bring your own beer. Wie der Name sagt, organisiert dabei jeder Gast seine eigenen Getränke, aber auch Knabberei, Grillkohle, den Stuhl zum Sitzen und oft noch mehr. Im Gegensatz zur zentralen Organisation eines Gastgebers, entsteht eine dezentrale Organisation der Veranstaltung durch die Gäste selbst. Es reicht aus, wenn nur wenige übergeordnete Parameter, wie Zeitpunkt und Ort, von Initiatoren vorgegeben werden.

Sicherlich ist es bequem, sich bedienen zu lassen, aber das Verteilen von Organisation auf eine Vielzahl einzelner Personen bietet Potenziale. Im Bezug auf die Nutzeranalyse und Prognose: Global ist schwer abzuschätzen, wie viele Leute insgesamt vertreten sein werden. Lokal ist jedem einzelnen klar, wann und ob er erscheinen wird. Im Bezug auf die Logistik: Global ist schwer abzuschätzen, wie viel und was konsumiert werden wird. Lokal weiß jeder Gast hoffentlich genauer, was und wie viel er verträgt oder eben nicht. Getreu dem Motto: Wenn jeder an sich denkt, ist an alle gedacht.

Beide Organisationstypen bieten Vorteile und Nachteile, die situationsbedingt abgewogen werden müssen. Auch Mischformen sind möglich und meist gegeben.

Bei den bAm, die im Rahmen dieser Arbeit vorgestellt werden, liegt der Fokus auf sich selbstorganisierenden dezentral gesteuerten Modulen.

3.1.2 SCHWARM

„Engineering problems related to Swarm Intelligence are mentioned in relation to Cellular Robotic Systems which consist of collections of autonomous, non-synchronized, non-intelligent robots cooperating to achieve global tasks.“

Gerardo Beni und Jing Wang, „Swarm Intelligence in Cellular Robotic Systems“, 1993

Der Begriff Schwarmintelligenz geht weit über das Thema der Selbstorganisation hinaus, oder zumindest parallel daran vorbei. Unter Schwarmintelligenz wird nicht nur verstanden, dass sich eine Gruppe selbst organisieren kann, oder vier Augen mehr sehen als zwei. Schwarmintelligenz beschreibt das emergente Phänomen, wenn eine Gruppe Funktionen erhält, die einzelne Akteure nicht haben. Möglicherweise kennen diese die erweiterte Funktion des Schwarmes gar nicht. Diese Idee entspringt ursprünglich der Robotik.

3.1.3 B O I D S

Diese Form von Schwarmintelligenz, die sich aus dem Handeln einzelner emergent ergibt, findet sich auch im Tierreich wieder: bei Ameisen, Fischschwärmen und Vögeln. Bereits 1987 präsentierte Craig Reynolds auf der Siggraph eine Methode, um das Flugverhalten von Vogelschwärmen zu simulieren. Die einzelnen Akteure, in diesem Fall auch Boids genannt, kennen drei grundlegende Regeln: Separation, Angleichung und Zusammenhalt. Jeder Boid evaluiert auf Grundlage seiner aktuellen Position, in Bezug auf jeweils nächste Nachbarn, seine weitere Bewegung. Zusätzliche Regeln, wie das Distanzieren von Hindernisse, oder das Erreichen von Zielen können ergänzt werden. Diese künstliche Intelligenz resultiert in einem bewegten, wechselwirkenden Schwarm. Durch Anstieg der Rechenleistung in den letzten Jahren ist die Einarbeitung weiterer Parameter und die Simulation komplexerer Agenten möglich geworden.

Es ist festzuhalten, dass ein Boid-System unvorhersehbar scheint, jedoch deterministisch ist. Es funktioniert nach vordefinierten Regeln. Erfährt ein System nach einem Neustart die gleichen Einflussfaktoren, wird es gleich reagieren. Unabhängig von der Zeit, sowie dem Computer, der es berechnet. Erst eine Abweichung von den Einflussfaktoren, egal wie klein diese ist, resultiert in einem unvorhersehbaren Verlauf von einzelnen Agenten. Jedoch ist das neue Ergebnis in Bezug auf die gesamte Aggregation selbstähnlich zum ersten. Die Parameter für Separation, Angleichung und Zusammenhalt, spiegeln sich in den Resultaten durch Eigenschaften wie Dichte und Richtung wieder. Die Kontrolle der Form tritt hinter die Kontrolle des gesamten Verhaltens.

Craig Reynolds, „Flocks, Herds, and Schools: A Distributed Behavioral Model“, 1987

3.1.4 B A M

Hier ist ein einfaches Boid-System zu sehen, das in abgewandelter Form in den Processing-Beispielen zu finden ist. Vom architektonischen Standpunkt aus könnte diskutiert werden: Die vereinzelten Boids bilden in bestimmten Bereichen bereits Flächen. Dieses Phänomen gilt es lediglich zu verstärken. Die im Rahmen dieser Forschungsarbeit entwickelten bAm basieren auf Boids. Dieses Verhalten wird bei bAm nicht in Echtzeit ausgeführt, sondern je nach Rechenleistung in mehreren Schritten im Voraus berechnet. Dadurch bietet sich die Möglichkeit, dass Agenten verschiedene Optionen kalkulieren und die zielführendste Variante wählen. Diese Berechnung erfolgt so schnell, dass der Nutzer sie als Echtzeit wahrnimmt.

Das Verhalten gliedert sich in drei Teile: die interne Logik, die Reaktion auf externe Einflussfaktoren und die übergeordneten Regeln des Schwarms. Die interne Logik besteht aus den drei grundlegenden Gesetzen von Reynolds. Erweitert werden sie durch die Regel, dass ein Knoten sich mit anderen kraftschlüssig verbinden kann.

Verweis auf:
Boids, Seite 145

Die Reaktion auf externe Einflussfaktoren, wie beispielsweise Nutzer, kann vereinzelte Regeln der internen Logik überschreiben oder addieren. Oberstes Ziel für das Erzeugen von Raum ist die Regel, im Bereich von Attraktoren in Form von Nutzern, die Position in Z-Richtung zu erhöhen.

Das führt zum dritten Punkt, den Verhaltensregeln des gesamten Schwarms. Dazu gehört die Evaluierung der Reichweite, der statischen Bestimmtheit, das Auflagern aller Aggregationen auf jeweils mindestens drei Punkten und die Kontrolle des Gleichgewichts. Wie diese statische Anforderung erfüllt wird, wurde zu einem wesentlichen Teil der Arbeit. Zu finden ist der Lösungsansatz im Abschnitt Kettenlinie.

Verweis auf:
Kettenlinie, Seite 184

Boidsystem
Processing Python
in drei Momentaufnahmen

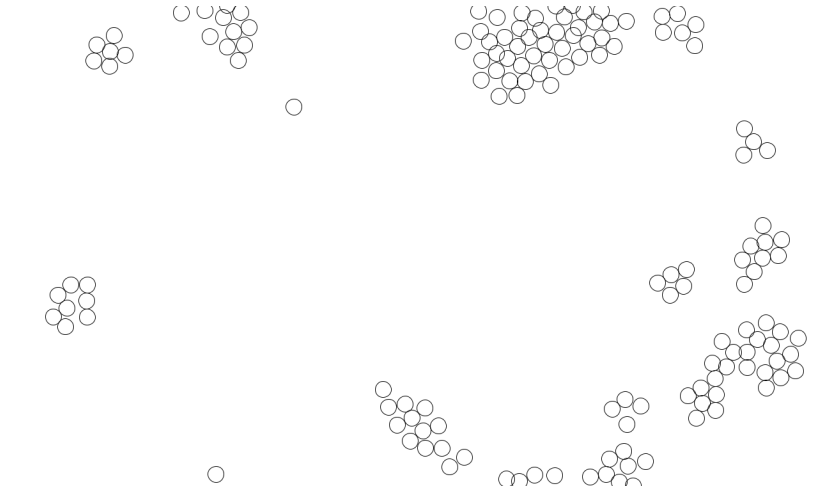


Abb.3.1.4-1 Momentaufnahme 1

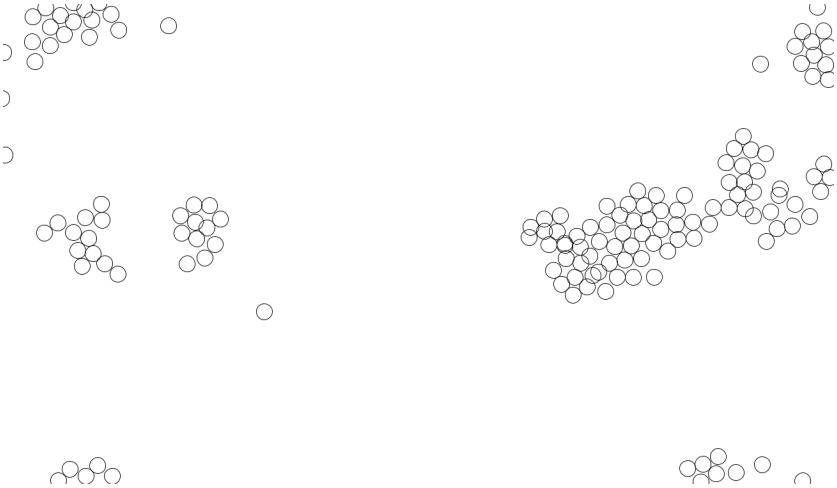


Abb.3.1.4-2 Momentaufnahme 2

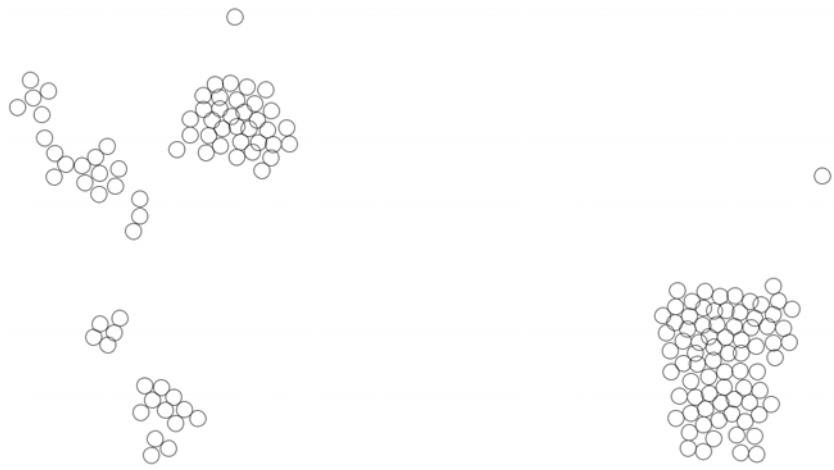


Abb.3.1.4-3 Momentaufnahme 3

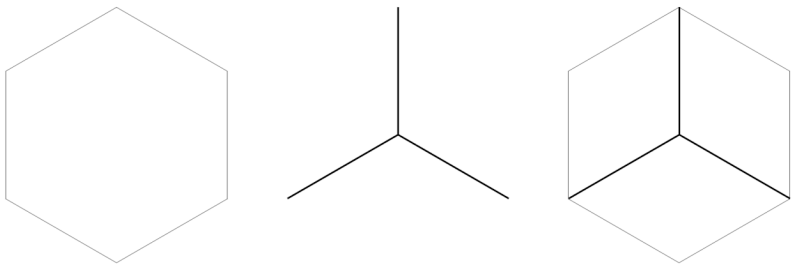
3.2 STRUKTUR

Verweis auf:
Gitterform, Seite 48

Wie bereits diskutiert, wäre das Dreieck die einfachere Lösung, aber nicht die effizienteste. Besonders bei den Kosten der Motoren spiegelt sich das wieder.

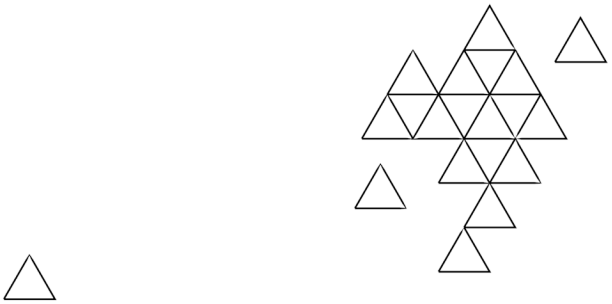
Bei der Aggregation zu einer Tragstruktur ist zudem zu beachten, dass die Kanten an den Enden nicht doppelt sind. Das Dreieck lässt sich versetzt zu einer Struktur aus Dreiecken zusammensetzten. Das Kreuz erlaubt, eine Struktur aus Vierecken zu generieren. Das Hexagon erlaubt, ein Fläche aus Hexagons zu erzeugen. Zu beachten aber auch: Ein Dreieck kann ebenso eine Fläche aus Hexagons formen, wenn diese entsprechend zusammen gesetzt werden, wie im Diagramm unten beschrieben ist.

Für die bAm wurde schließlich das Dreieck als Grundform gewählt. Das entstehende Hexagon verspricht eine hohe Effizienz in Bezug auf den einzusetzenden Materialaufwand. Es ergibt sich ein niedrigeres Gewicht, sowie niedrigere Energiekosten. Statische Bestimmtheit wird erzeugt, indem die einzelnen Gelenke zum Kern hin zwar mittels Motoren beweglich, aber so auch biegesteif ausgeführt sind.

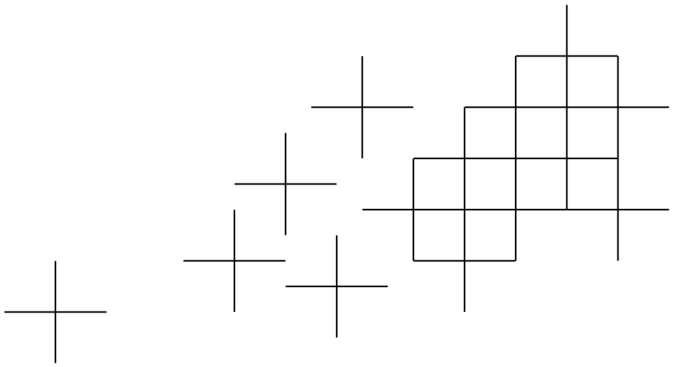


Zchng.3.2.0-1 Grundform

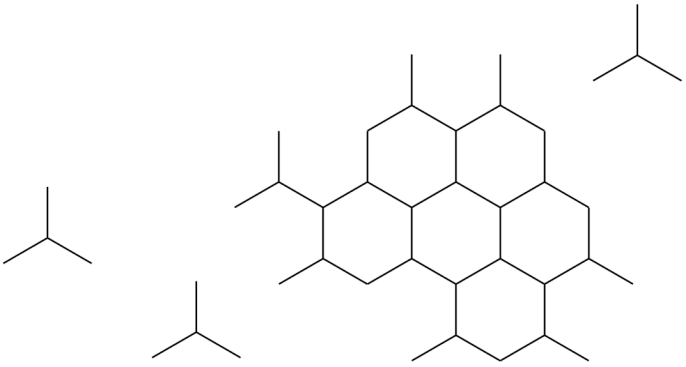
Studie zur Gitterform
Vergleich von Dreieck, Viereck, Sechseck
alle Varianten basieren auf gleicher Kantenlänge



Zchng.3.2.0-2 Dreiecke



Zchng.3.2.0-3 Vierecke



Zchng.3.2.0-4 Sechsecke

3.2.1 VOLLSTÄNDIG

Eine erste Ausformulierung dieses Dreiecks ist hier dargestellt. Diese Variante mit neuen Motoren folgt dem Konzept: Das gesamte Tragwerk besteht aus Motoren. Lediglich der Kern ist fest. Auflager und Knotenpunkte sind auf den folgenden Seiten schematisch dargestellt und werden in Folge weiter ausgearbeitet. Vorerst geht es lediglich um die zugrundeliegende Grundform der bAm.

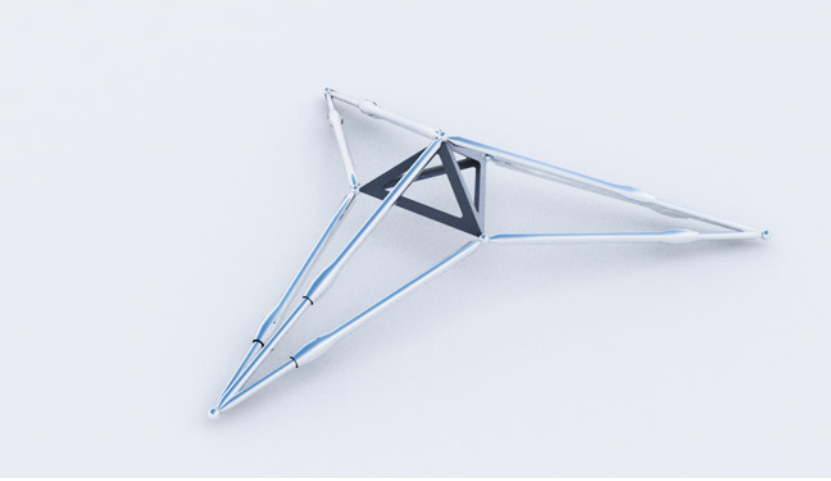
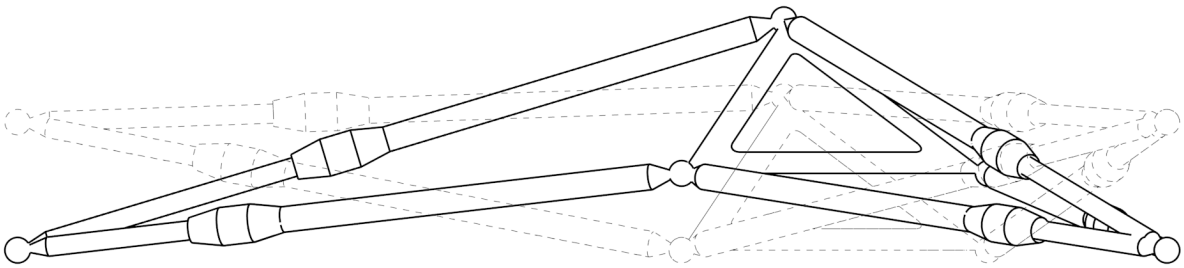
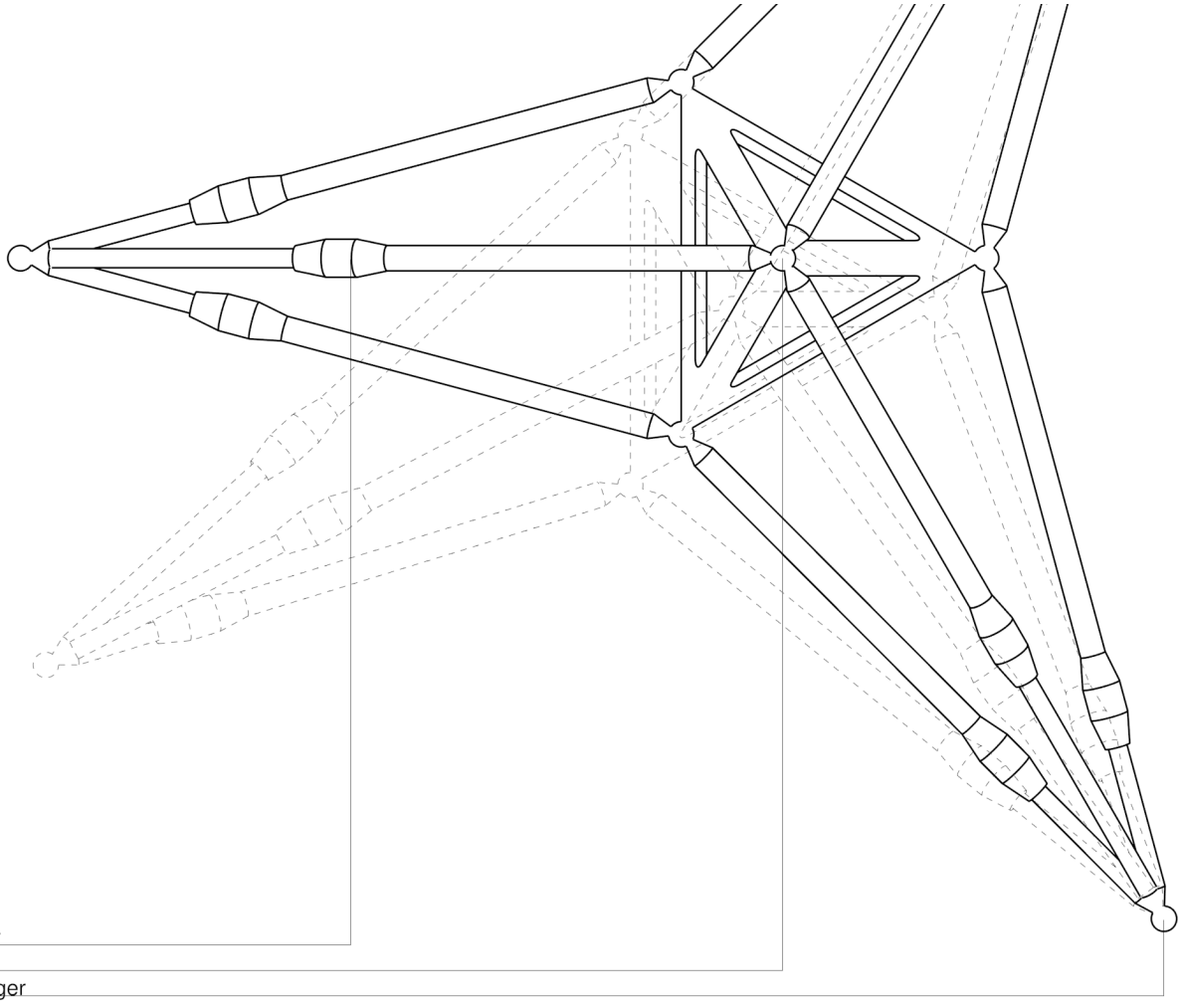


Abb.3.2.1-1 Rendering

Topologie
schematische Darstellungen
Varianten mit neun Linearaktuatoren



Zchng.3.2.1-1 Schematische Ansicht



Zchng.3.2.1-2 Schematischer Grundriss

Diese Simulationen zeigen die drei entscheidenden Worst-Case-Bewegungen, die möglich sind. Begonnen wird mit der maximalen Größenänderung, die das Modul machen kann. Die zweite Spalte zeigt das Umsehen des Modules, um andere Module zu finden. Dieser Prozess wird anschließend noch eingehender diskutiert. In der dritten Spalte wird schließlich gezeigt, wie ein bAm sich fortbewegen kann. Dabei hebt das Modul vorerst die Beine und versetzt diese in die zu gehende Richtung. Anschließend hebt das Modul den Kern und zieht diesen nach.

Verweis auf:
Umsehen, Seite 178

Grundsätzlich ist zu erwarten, dass diese Variante ein hohes Spektrum an möglicher Bewegung abdeckt, zugleich aber wenig effizient ist. Auf den Folgeseiten werden alternative Lösungen vorgestellt.

Topologie
Größenänderung, Umsehen, Gehen
jeweils größtmöglicher Bewegungsfreiraum

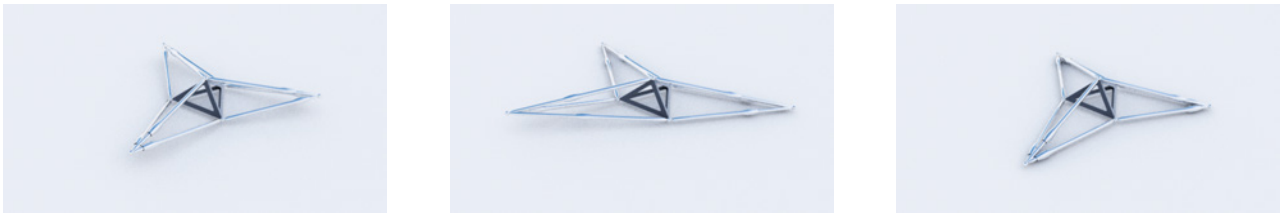


Abb.3.2.1-2 a,b,c Ausgangsposition

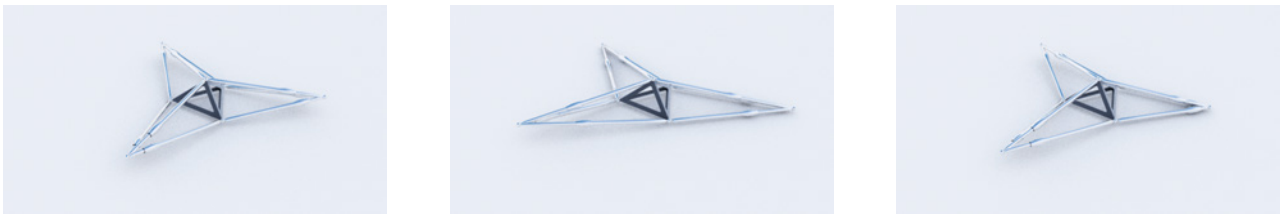


Abb.3.2.1-3 a,b,c 20 %

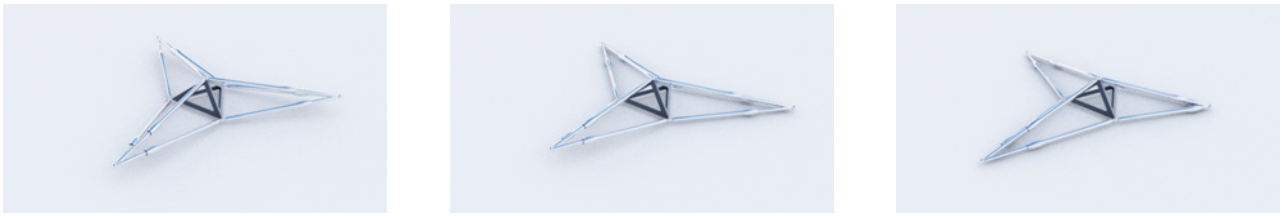


Abb.3.2.1-4 a,b,c 40 %

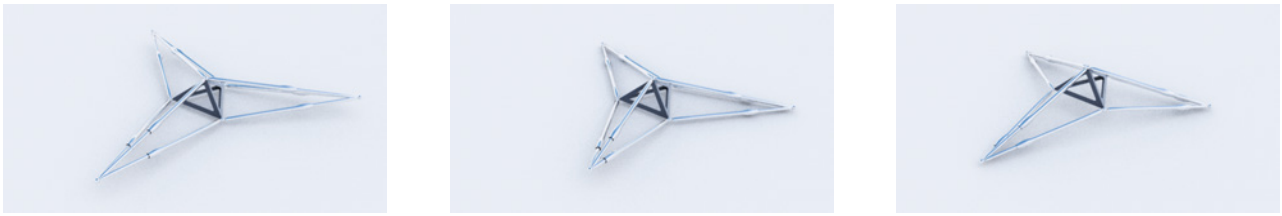


Abb.3.2.1-5 a,b,c 60 %

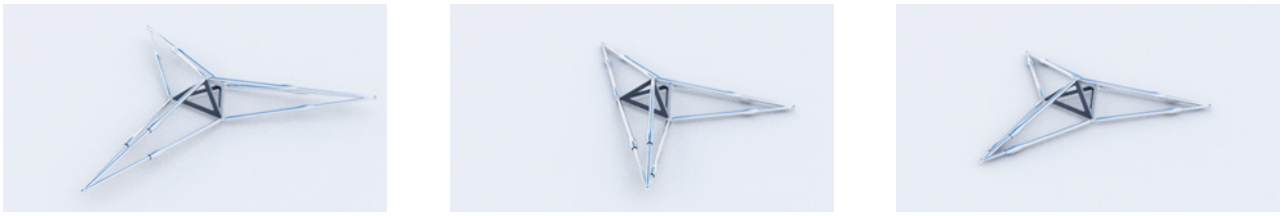


Abb.3.2.1-6 a,b,c 80 %

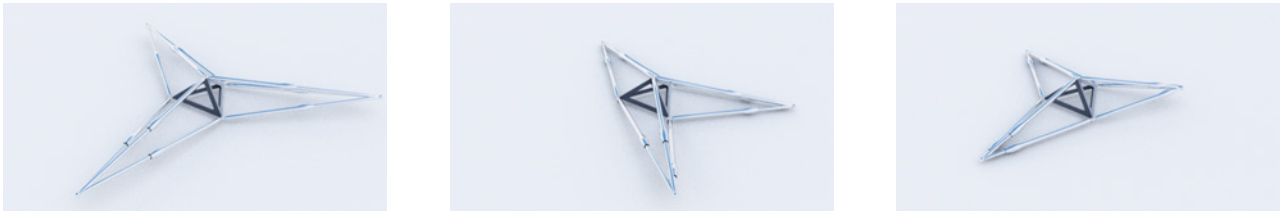


Abb.3.2.1-7 a,b,c Endposition

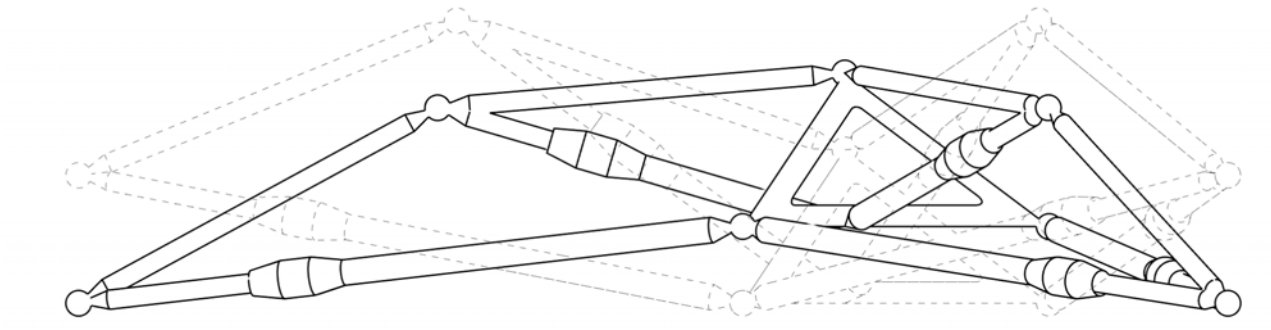
3.2.2 MIT TRÄGER

Die vorausgegangene Variante wäre beim Poker gewissermaßen ein All In. Gibt es bessere Spielzüge die erlauben, den Einsatz von Gewicht, Kosten und Material zu reduzieren?

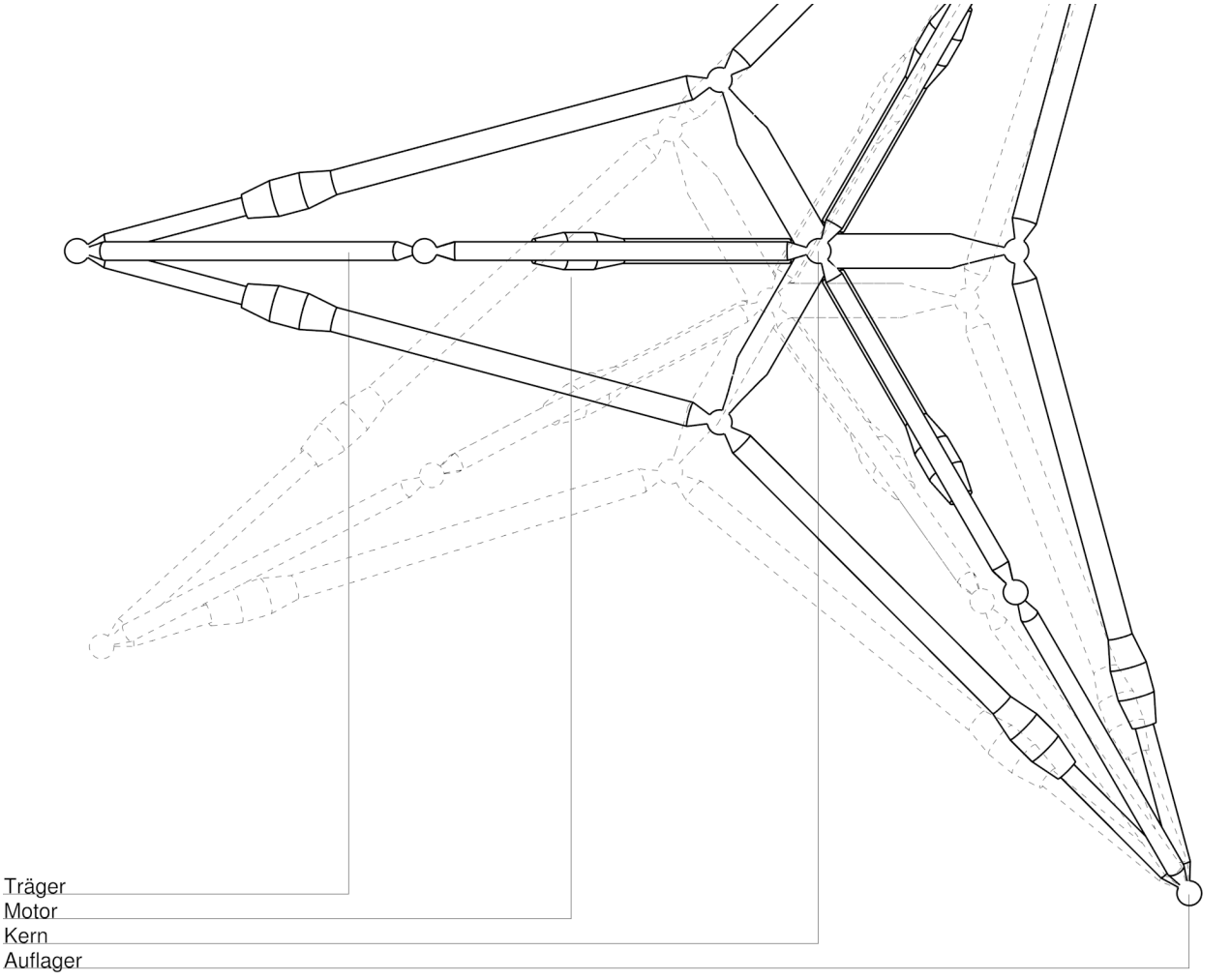
Diese Variante besteht ebenfalls aus neun Motoren. Jedoch sind jene Teile der Konstruktion, die überwiegend Lasten tragen, durch feste Träger ersetzt. Die Motoren müssen dabei lediglich die Ausrichtung der starren Träger übernehmen. Die Motoren können so weniger stark dimensioniert werden. Die Struktur wird insgesamt leichter, da feststehende Elemente ein besseres Verhältnis von Kraft zu Gewicht besitzen als die Motoren.

Abb.3.2.2-1 Rendering

Topologie
schematische Darstellungen
Varianten mit neun Linearaktuatoren



Zchng.3.2.2-1 Schematische Ansicht



Zchng.3.2.2-2 Schematischer Grundriss

In Bezug auf mögliche Bewegungen ergeben sich allerdings Nachteile gegenüber der vollmotorisierten Variante. Die Größenänderung funktioniert dabei nur durch das Abknicken und Aufklappen der Gelenke. Eine Größenänderung resultiert dabei in einer Änderung der Höhe des Modules. Beim Umsehen gilt das Gleiche. Allerdings ist dabei kein wesentlicher Nachteil festzustellen. Beim Gehen des Modules ist ebenso kein Nachteil zu notieren.

Topologie
Größenänderung, Umsehen, Gehen
jeweils größtmöglicher Bewegungsfreiraum

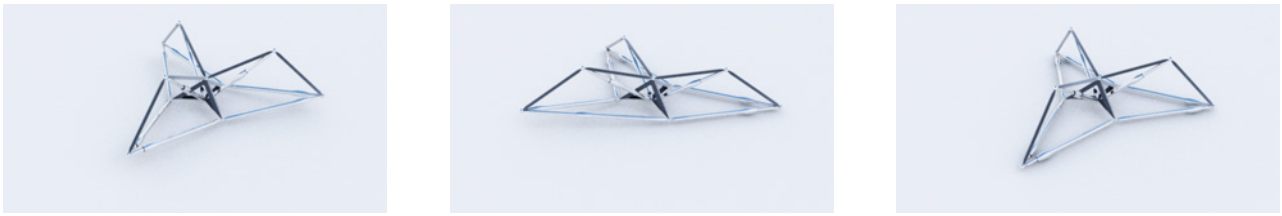


Abb.3.2.2-2 a,b,c Ausgangsposition

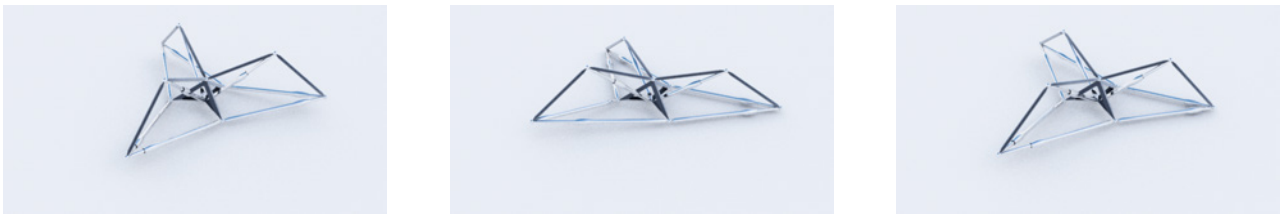


Abb.3.2.2-3 a,b,c 20 %

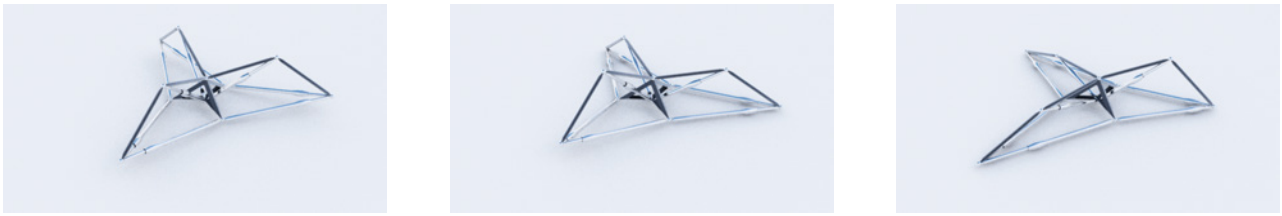


Abb.3.2.2-4 a,b,c 40 %

Abb.3.2.2-5 a,b,c 60 %

Abb.3.2.2-6 a,b,c 80 %

Abb.3.2.2-7 a,b,c Endposition

3.2.3 REDUZIERT

Die nächste Variante ist eine Reduktion der Motoren auf vier Stück. Dabei sind die Füße vollständig durch feste Träger ersetzt. Die Enden der Füße werden durch Motoren verbunden. Ein weiterer Motor wird benötigt, damit der bAm sich fortbewegen kann. Anzumerken ist, dass die Linearaktuatoren in der Aggregation zusammenhängende Dreiecke bilden würden, wie im Abschnitt Struktur bei Dreieck beschrieben.

Verweis auf:
Struktur der bAm, Seite 148

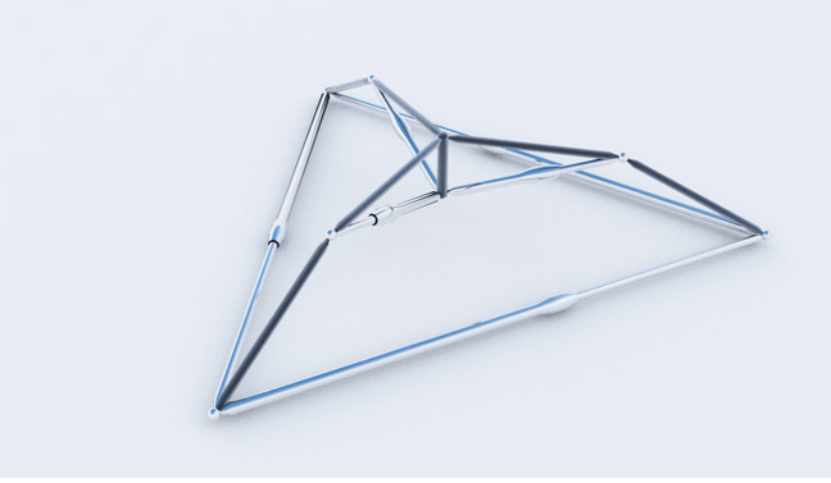
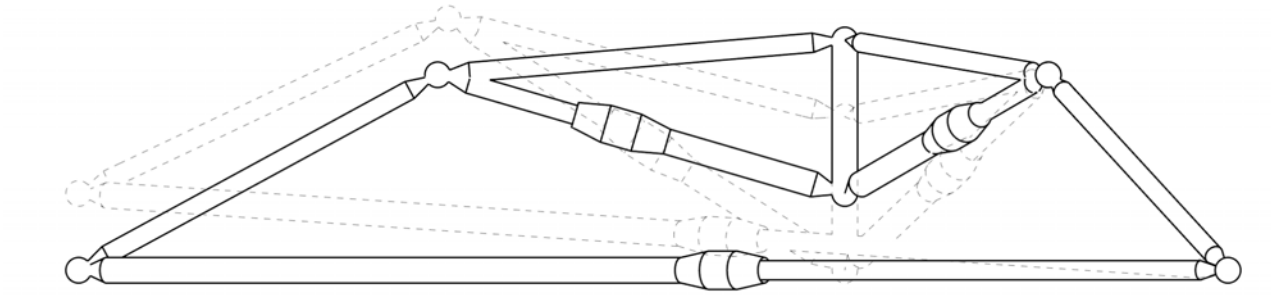
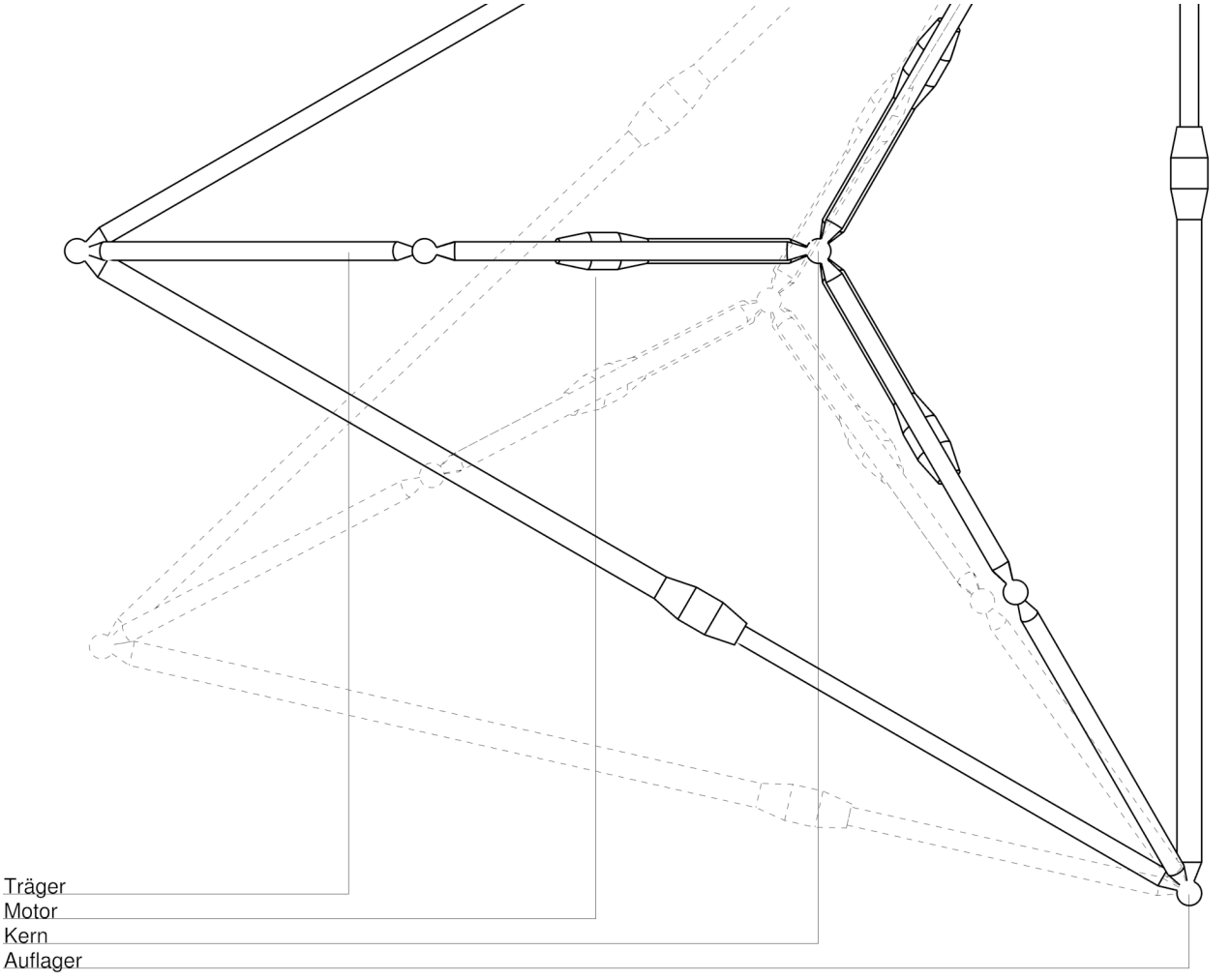


Abb.3.2.3-1 Rendering

Topologie
schematische Darstellungen
Varianten mit vier Linearaktuatoren



Zchng.3.2.3-1 Schematische Ansicht



Zchng.3.2.3-2 Schematischer Grundriss

In Bezug auf die Größenänderung gibt es zur vorausgegangen Variante keinen Nachteil. Das Gleiche gilt für das Umsehen. Einschränkungen sind allerdings beim Gehen festzustellen. Hier kann das Modul jeweils nur einen Fuß oder den Kern gleichzeitig heben.

Topologie
Größenänderung, Umsehen, Gehen
jeweils größtmöglicher Bewegungsfreiraum

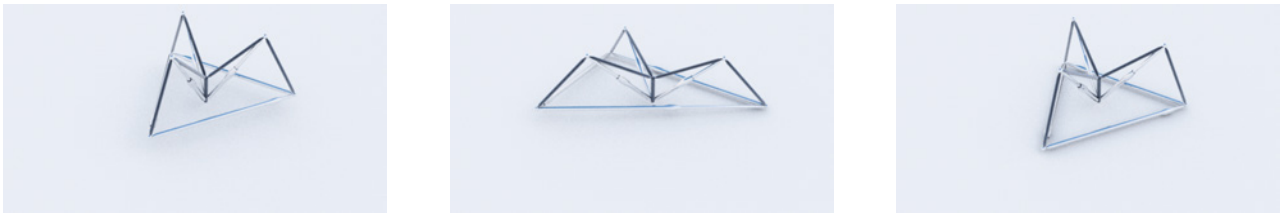


Abb.3.2.3-2 a,b,c Ausgangsposition

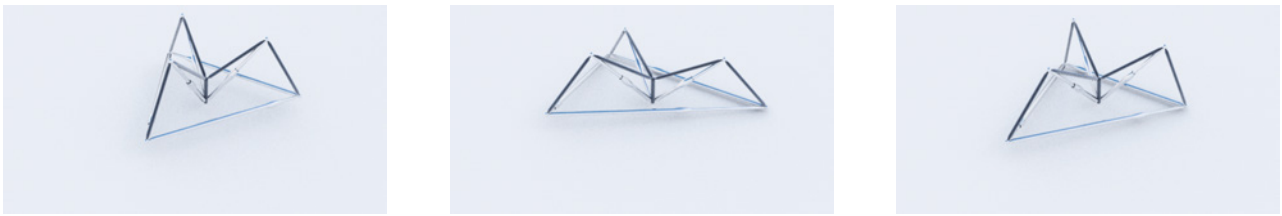


Abb.3.2.3-3 a,b,c 20 %

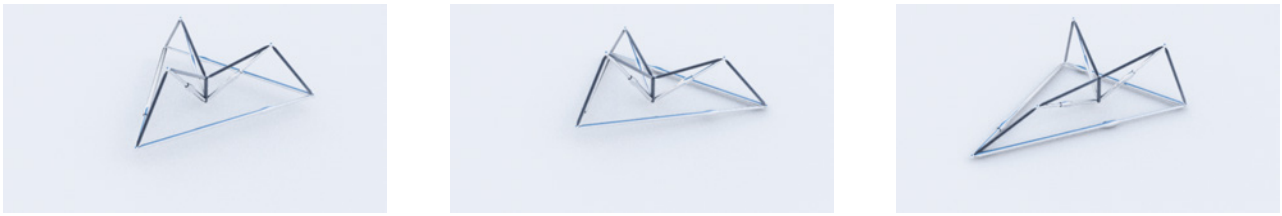


Abb.3.2.3-4 a,b,c 40 %

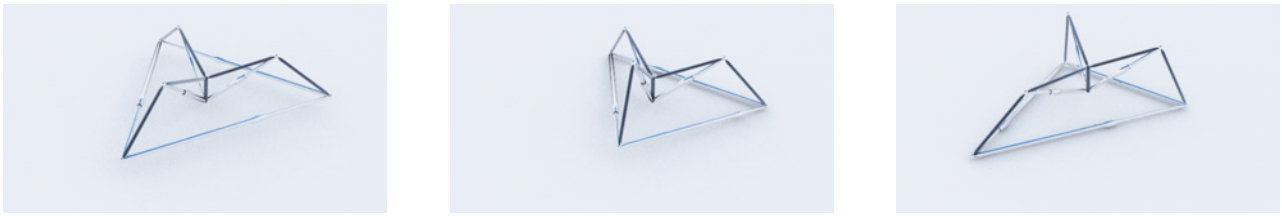


Abb.3.2.3-5 a,b,c 60 %

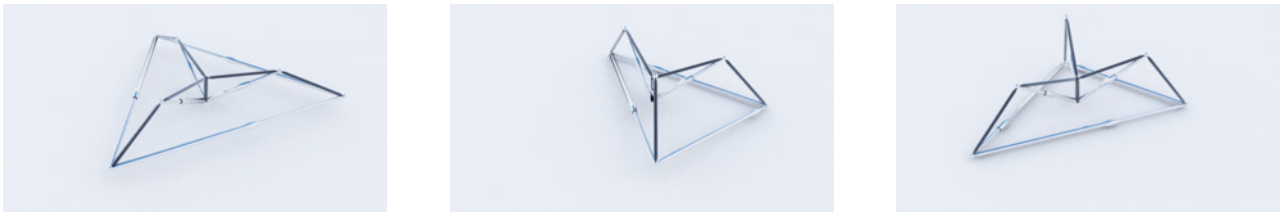


Abb.3.2.3-6 a,b,c 80 %

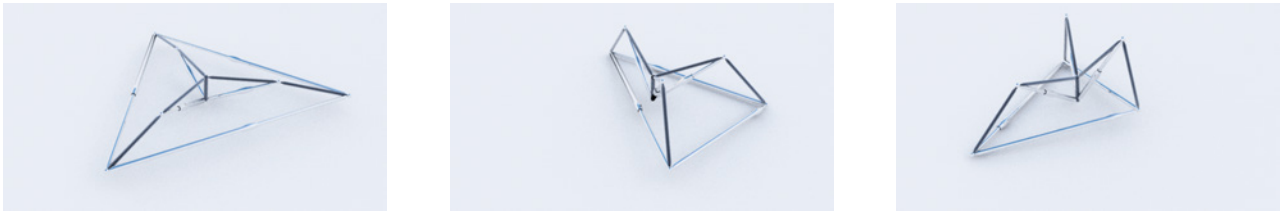


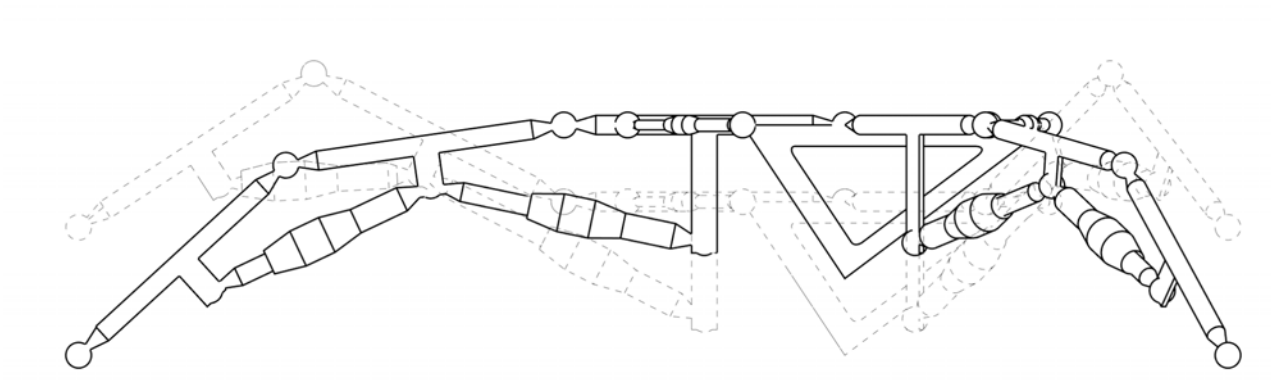
Abb.3.2.3-7 a,b,c Endposition

3.2.4 G E R A D E

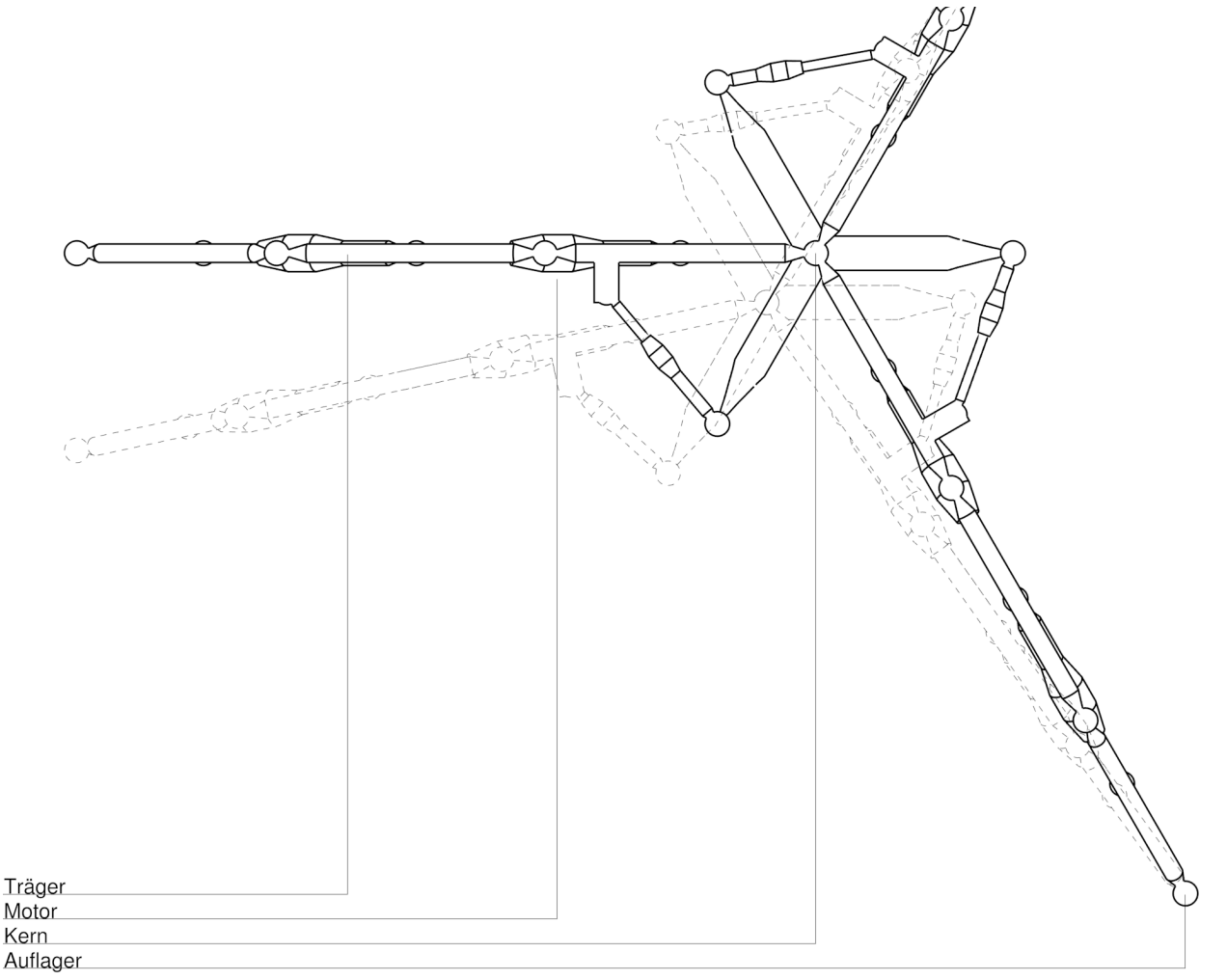
Die Variante mit kombinierten Trägern und Motoren scheint eine Lösungsvariante zu sein, die im Rahmen der Forschungsarbeit ein ausgewogenes Maß an Vorteilen und Nachteilen bietet. Die folgenden zwei Varianten sind daher eine Abwandlung und Weiterführung dieser Idee.

Abb.3.2.4-1 Rendering

Topologie
schematische Darstellungen
Varianten mit geradem Gelenk in der Mitte



Zchng.3.2.4-1 Schematische Ansicht



Zchng.3.2.4-2 Schematischer Grundriss

Vor- und Nachteile bei der Bewegung sind bei dieser Variante analog zu der Variante mit sechs Motoren.

Topologie
Größenänderung, Umsehen, Gehen
jeweils größtmöglicher Bewegungsfreiraum

Abb.3.2.4-2 a,b,c Ausgangsposition

Abb.3.2.4-3 a,b,c 20 %

Abb.3.2.4-4 a,b,c 40 %

Abb.3.2.4-5 a,b,c 60 %

Abb.3.2.4-6 a,b,c 80 %

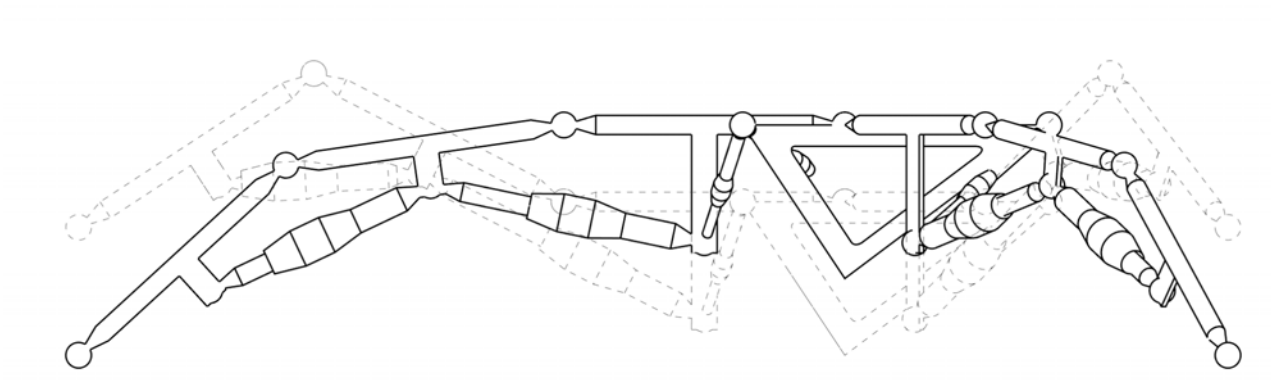
Abb.3.2.4-7 a,b,c Endposition

3.2.5 D R E H E N D

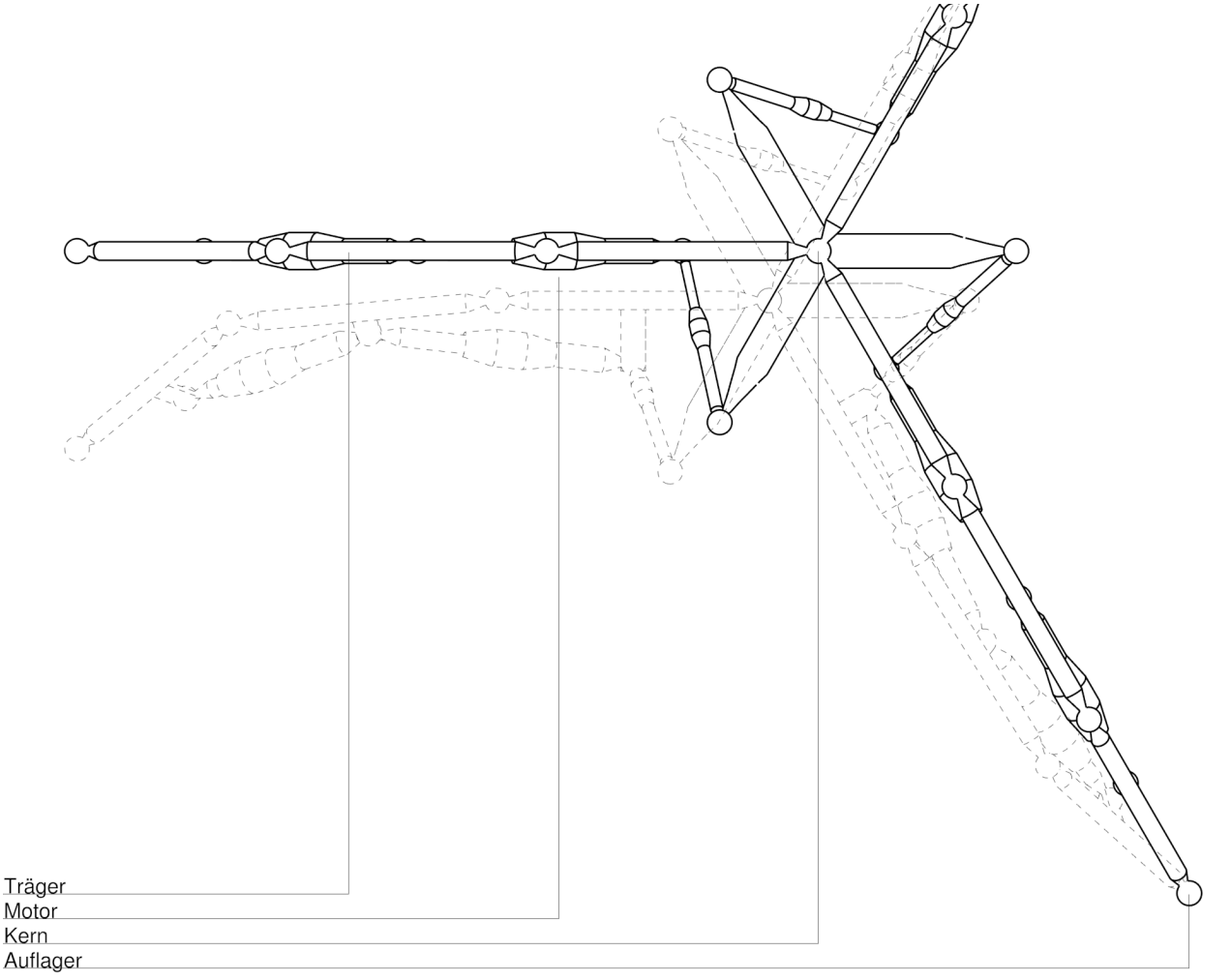
Diese Variante ist eine weitere Abwandlung. Auf den ersten Blick ist diese zur Vorgängerversion identisch. Lediglich das Knotengelenk am Kern verfügt über andere Freiheitsgrade. Dieses kleine Detail erlaubt der Variante aber eine andere Form von Bewegung. Die Füße können sich eindrehen.

Abb.3.2.5-1 Rendering

Topologie
schematische Darstellungen
Varianten mit drehendem Gelenk in der Mitte



Zchng.3.2.5-1 Schematische Ansicht



Zchng.3.2.5-2 Schematischer Grundriss

Vorteile ergeben sich hier beim Umsehen, da das abgedeckte Blickfeld durch das Eindrehen der Füße vergrößert wird. Kritisch zu erwähnen ist, dass bei den eingedrehten Füßen Momente entstehen. Für die Umsetzung des ersten bAm als Prototyp wurde daher die Vorgängervariante weiter ausgearbeitet.

Topologie
Größenänderung, Umsehen, Gehen
jeweils größtmöglicher Bewegungsfreiraum

Abb.3.2.5-2 a,b,c Ausgangsposition

Abb.3.2.5-3 a,b,c 20 %

Abb.3.2.5-4 a,b,c 40 %

Abb.3.2.5-5 a,b,c 60 %

Abb.3.2.5-6 a,b,c 80 %

Abb.3.2.5-7 a,b,c Endposition

3.2.6 PROTOTYPEN 3 BIS 4

Analog zu den digitalen Modellen entstanden auch physische Modelle. Bei den soeben diskutierten Varianten wird ein Linearaktuator eingesetzt, um eine Drehbewegung eines Gelenkes zu erzeugen. Die beiden ersten Prototypen dieses Abschnitts versuchen, die Drehung des Motors unmittelbar zu nutzen. Im architektonischen Maßstab wäre das auch mittels Planetengetriebe möglich.

Prototyp 4 liefert das höchste Spektrum an Bewegungsfreiheit. Jedoch ist festzuhalten, dass die Füße beim Anheben zur Seite ausbrechen. Das Modell ist kinematisch eindeutig bestimmt, aber die Gelenke sind zu elastisch, um der seitlichen Krafteinwirkung entgegen zu wirken.

Bei Prototyp 5 wurde die Anzahl der Motoren ein weiteres Mal reduziert. Das Anheben des Körpers funktioniert wesentlich einfacher. Die Größenänderung des Modules geschieht durch Anziehen und Ausstrecken der Füße. Schwierigkeiten entstehen beim Gehen, da das Modell nicht wie bei seinem Vorgänger nach Anheben des Körpers die Füße weiter setzen kann. Die Fortbewegung funktioniert daher durch Schieben. Zwei Füße schieben dabei in entgegengesetzte Richtung. So schleift sich das Modell schließlich in eine Richtung. Eine Drehbewegung ist durch das Anheben und Schieben eines Fußes möglich. Durch die Drehung kann die Bewegungsrichtung genau bestimmt werden. Das Umsehen funktioniert analog zu den Vorgängern. Das Modell weist eine höhere Effizienz in Bezug auf den Materialeinsatz beim Bau auf. In Kauf genommen werden muss, dass das Modell Reibung bei der Bewegung erzeugt. Das wird sich natürlich unmittelbar im Energieverbrauch und im Verschleiß von Teilen widerspiegeln.

Das letzte Modell in diesem Abschnitt zeigt schließlich eine Rückführung auf die Idee von Linearbewegungen. Jeder Fuß besteht dabei aus einem festen Element und zwei Schubstangen. Größenänderung, Fortbewegung, Drehung und Umsehen funktionieren analog zum Vorgänger.

Folgeseite:
Abb.3.2.6-1 Modellfoto Prototyp 5

Prototypen zu Topologie
mit Drehbewegung und Schubstangen
analoge Servos, Alu, Kunststoff, Edelstahl

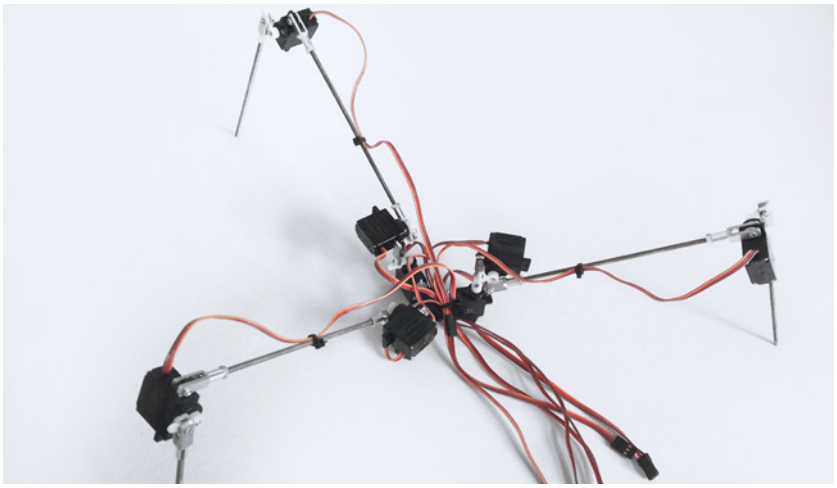


Abb.3.2.6-2 Prototyp 3

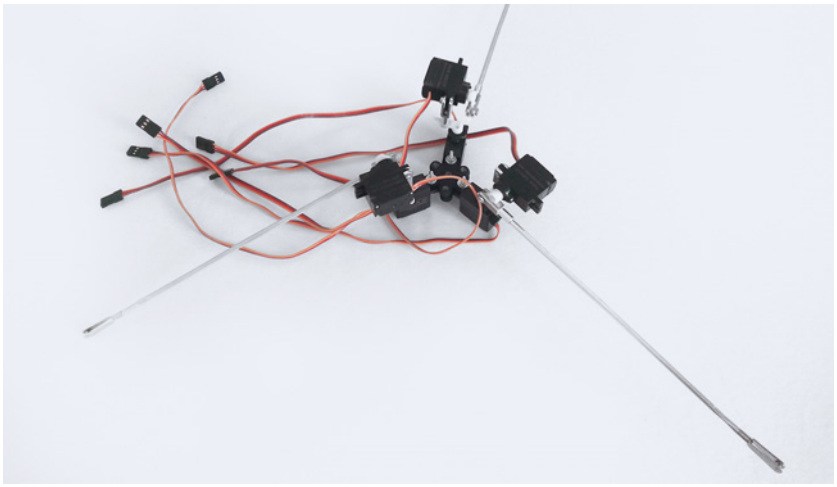


Abb.3.2.6-3 Prototyp 4

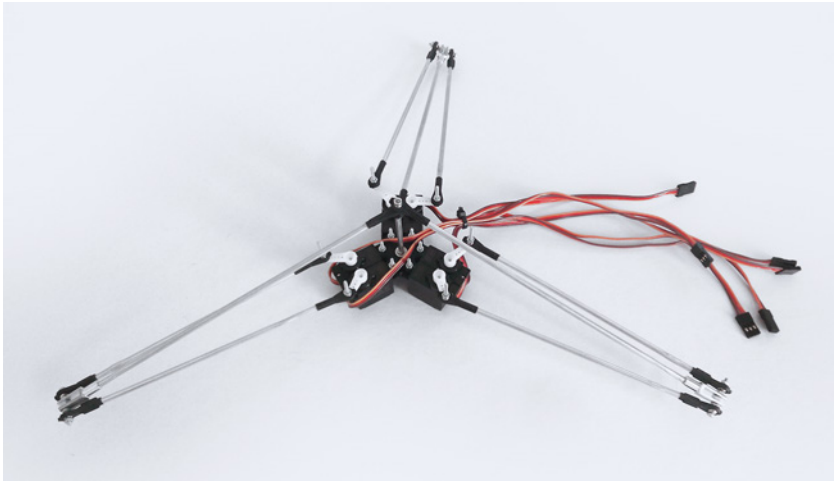
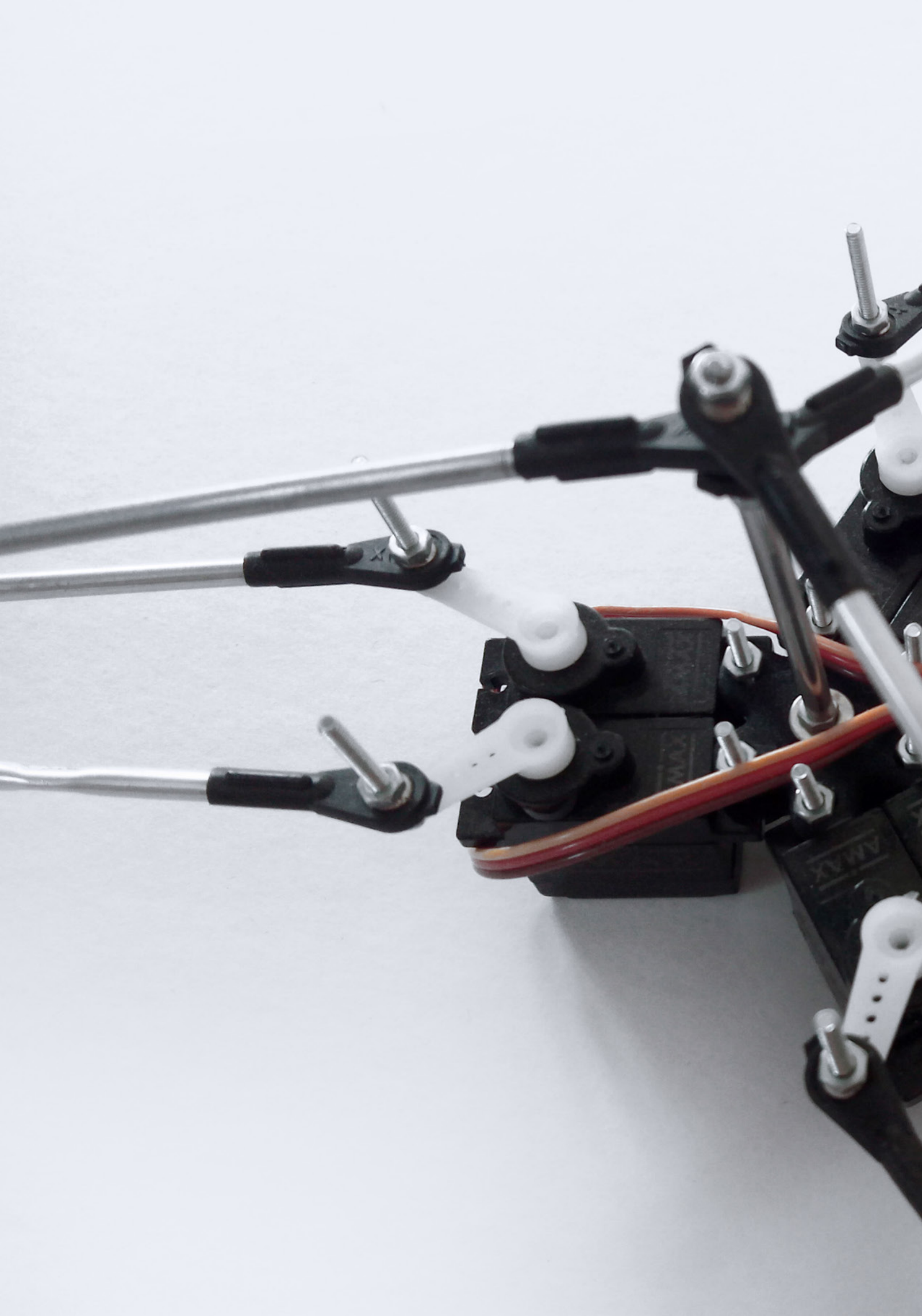


Abb.3.2.6-4 Prototyp 5



3.3 REAKTION

3.3.1 SELBST

Dieser Abschnitt startet mit der Reaktion der Module auf sich selbst. Gemeint ist damit eine grundlegende Überprüfung der eigenen Funktionalität. So wird der Luftdruck der Pneus mittels eines Barometers gemessen. Sinkt der Wert, wird eine integrierte Pumpe aktiviert, die den Pneu wieder auffüllt. Läuft die Pumpe zu lange, ist der Pneu defekt oder ein Ventil geöffnet. Der bAm ändert seinen Status auf defekt.

„Ein junger Salamander kann nach fünf Wochen ein komplettes Bein nachwachsen lassen.“

Eurostemcell, „Regeneration: Was ist das und wie funktioniert es?“, 2011

Zweiter Punkt ist das Messen, wieviel Strom vom Modul gezogen wird. Das Monitoring vom Energieverbrauch ist ein Aspekt. Ein weiterer ist, dass so die Belastung von jedem Motor ermittelt werden kann. Die Motoren bewegen sich dabei separat, ein kleines Stück vor und zurück. Die jeweilige Belastung, in die jeweilige Richtung, kann dann mittels der gemessenen Ampere rückgerechnet werden. Ziehen die Motoren gar keinen Strom, oder nur verschwendend gering, kann ein Defekt festgehalten werden.

Weitere Kontrollen werden auch für das Arduino und das Raspberry Pi angesetzt. Bei einem Defekt signalisiert der bAm diese über ein eingebautes Display. Ist eine Verbindung ins Internet verfügbar, sendet der bAm eine entsprechende Mail an den Support. Bei nächster Gelegenheit versucht sich der bAm aus der Aggregation auszuklinken. Ein Nachbar wird ihn ersetzen. Nach der Reparatur wird er sich wieder einfügen.

Dieser Script zeigt das Auslesen des Luftdrucks und basiert auf den Ausführungen von Martin Kompf. Zu Grunde liegt die Bibliothek Adafruit BMP, welche von Tony Di Cola für Adafruit entwickelt wurde. Für das Senden der Email wird Yagmail von Pascal van Kooten verwendet. Nebenbei bemerkt: In der Abbildung unten ist die erste Email des ersten bAm zu sehen.

Martin Kompf, „Barometer mit dem Raspberry Pi und dem I2C Luftdrucksensor BMP085“, 2016

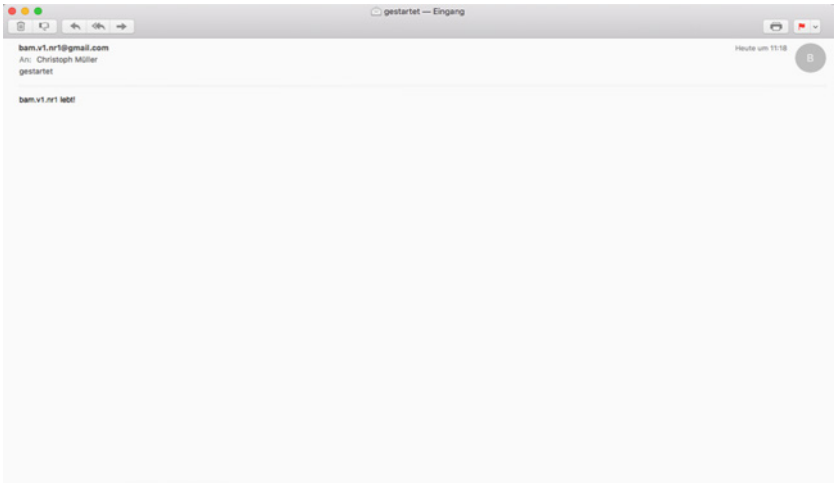
Adafruit Industries LLC, „Adafruit Python BMP“, 2016

Pascal van Kooten, „yagmail“, 2016

Auslesen des Drucksensors und Schreiben von Emails
Scriptsprache: Python
Plattform: Raspberry Pi 2

```python
# Setup Email
import yagmail
yag = yagmail.SMTP('bam.v1.nr1@gmail.com', 'bAmPasswort')
Nachricht = "Die Pneus von bam.v1.nr1 verlieren mehr Luft, ↩
    ↪ als nachgepumt werden kann."

# Setup Sensor
import Adafruit_BMP.BMP180 as BMP180
Sensor = BMP180.BMP180()

def Drucktest():
    Druck = sensor.read_pressure()
    return Druck

def Temperatur():
    Temperatur = sensor.read_temperature()
    return Temperatur

# Ablauf
if Drucktest() > 1000:
    pumpe(120)
    if Drucktest() > 1000:
        yag.send('mail@mueller-christoph.com', subject = "↩
            ↪ Defekt", contents = Nachricht)
```

Scp.3.3.1-1 Selbstkontrolle und Support



Diagr.3.3.1-1 Erste Email

3.3.2 KONTEXT

Verweis auf:
Umsehen, Seite 178

Die Module haben grundlegende Funktionen, ihre unmittelbare Umgebung zu registrieren. Jeder Fuß ist dabei mit einer Webcam und einem Ultraschall-Sensor ausgestattet. Das Umsehen und Scannen der Umgebung geschieht dabei wie im Abschnitt Topologie beschrieben.

Der Script zeigt die Verwendung von Ultraschallsensoren. Die Grundlagen dazu werden von Dejan Nedelkovski sehr anschaulich aufgezeigt. Zum Auslesen wird die neue Ping-Bibliothek verwendete, die von Tim Eckel für Arduino entwickelt wurde. Analog zu den Motoren wird ebenfalls die serielle Schnittstelle als Kommunikationsweg zwischen Controller und Computer verwendet.

Dejan Nedelkovski, „Ultrasonic Sensor HC-SR04 and Arduino“, 2016

Tim Eckel, „New Ping Example“, 2016

Das Erkennen von Menschen und Agenten mittels der Webcams wird auf den Folgeseiten genauer beschrieben.

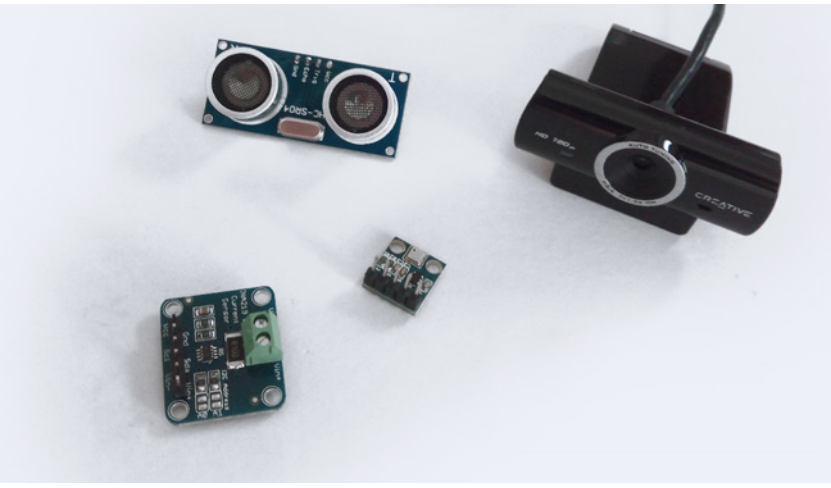


Abb.3.3.2-1 Sensoren und Webcam

Auslesen der Ultraschallsensoren
Scriptsprache: C++
Plattform: Arduino Mega 2560

```
#include <NewPing.h>

#define TRIGGER_PIN 12
#define ECHO_PIN 11
#define MAX_DISTANCE 500

NewPing sonar(TRIGGER_PIN, ECHO_PIN, MAX_DISTANCE);

void setup() {
  Serial.begin(9600);
}

int Ultraschallsensor;

void loop() {
  {
    while (Serial.available() == 0);
    Ultraschallsensor = Serial.read();

    // Relevant bei mehreren Ultraschallsensoren, wie beim ↩
        ↪ bAm
    if (Ultraschallsensor == 20){
      delay(50);
      int uS = sonar.ping();
      Serial.println(uS / US_ROUNDTRIP_CM);
      }

    Ultraschallsensor = 0;
  }
}
```

Scp.3.3.2-1 Ultraschallsensor

$$v = 340\ m/s$$

Frl.3.3.2-1 Schallgeschwindigkeit

$$t = s/v$$

Frl.3.3.2-2 Zeit ermitteln

$$s = \frac{t \cdot 0,034}{2}$$

Frl.3.3.2-3 Distanz ermitteln

3.3.3 AGENTEN

Anthony Oliver, „Object Tracking with SimpleCV“, 2012

Verweis auf:
Verbindung, Seite 214

Dieser Abschnitt beschreibt, wie sich die Agenten gegenseitig erkennen können. Das geschieht mittels dem Umsehen und der freien Bibliothek Simple CV, welche wiederum auf Open CV von Intel basiert. Anthony Oliver zeigt ein Beispiel auf, das für die bAm adaptiert wurde und weiter ausgearbeitet wird.

Gesucht wird dabei nach den kugelförmigen Verbindungsstücken anderer bAm, die noch eingehender besprochen werden. Die gefundenen Verbindungsstücke werden dabei für jede Kamera separat mit ihrer Größe gespeichert. Der Mittelwert aller Punkte, mit ihrer Größe als Faktor, wird errechnet. Schließlich kann der bAm sich in die Richtung mit den meisten Treffern bewegen.

Da es derzeit nur einen bAm gibt, ist dieser Abschnitt nur schwer zu evaluieren. Auf der Folgeseite wird daher eine alternative Simulationsmethode vorgestellt.

Finden von anderen Agenten
Scriptsprache: Python
Plattform: Raspberry Pi 2

```
import SimpleCV

display = SimpleCV.Display()
cam = SimpleCV.Camera(0)
normaldisplay = True

while display.isNotDone():

    if display.mouseRight:
        normaldisplay = not(normaldisplay)
        print "Display Mode:", "Normal" if normaldisplay else "segmented"

    img = cam.getImage().flipHorizontal()
    dist = img.colorDistance(SimpleCV.Color.BLACK).dilate(2)
    segmented = dist.stretch(200,255)
    blobs = segmented.findBlobs()
    if blobs:
        circles = blobs.filter([b.isCircle(0.2) for b in blobs])
        if circles:
            print("Verbindungsstück gefunden")
            img.drawCircle((circles[-1].x, circles[-1].y), circles[-1].radius(),(0,255,0),3)

            # An diesem Punkt werden die gefunden Verbindungsstücke für jede Kamera separat gespeichert.
            # Der Mittelwert aller Punkte wird errechnet und schließlich der bAm in die Richtung mit den meisten Treffern bewegt

    if normaldisplay:
        img.show()
    else:
        segmented.show()
```

Scp.3.3.3-1 Verbindungsknoten erkennen

Dieser Abschnitt zeigt lediglich eine Simulation der Annäherung. Verwendet wird als Grundlage eine Cloth-Simulation mit Sewing. Das Sewing erlaubt, dass festgelegte Kanten sich zusammenziehen. Die Module sind in der Simulation so gescriptet, dass ihre Füße sich dabei kinematisch korrekt bewegen und die Fortbewegung untersuchbar ist.

Folgeseite:
Abb.3.3.3-1 Rendering

Simulation der Zusammensetzung
Scriptsprache: Python
DCC: Blender

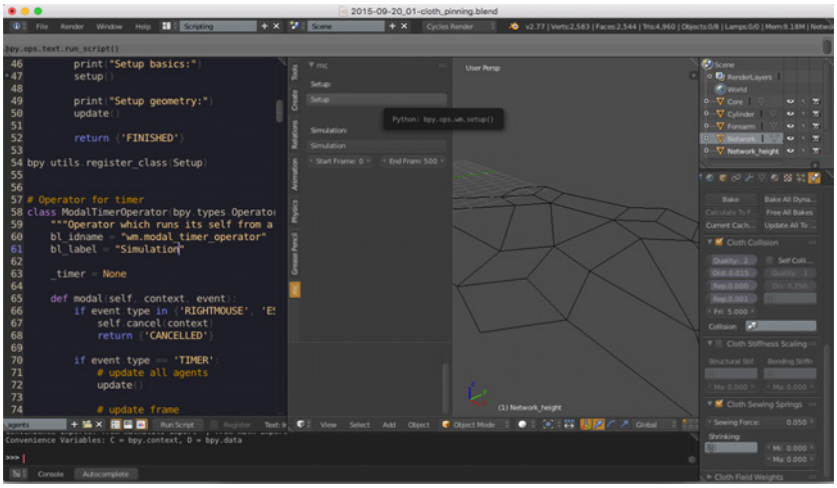


Abb.3.3.3-2 Anfangsposition

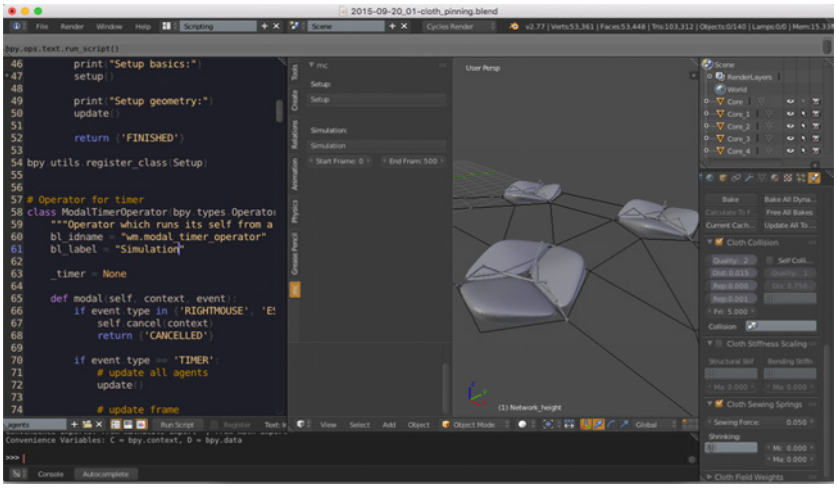


Abb.3.3.3-3 Zwischenschritt

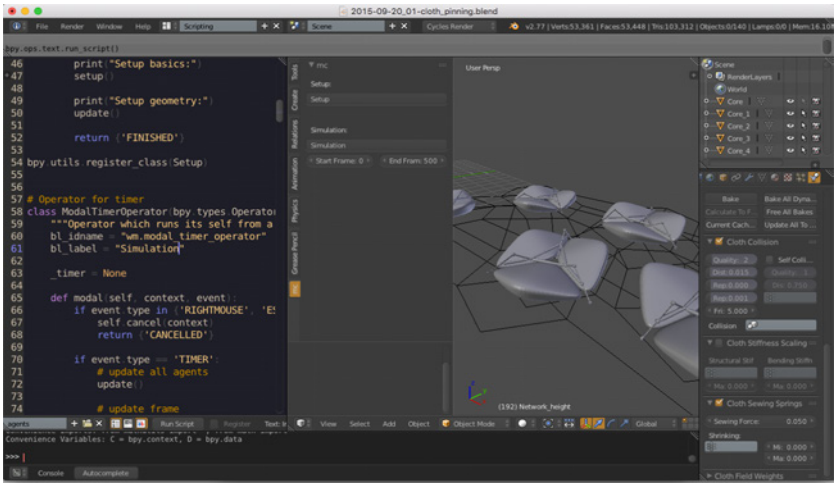
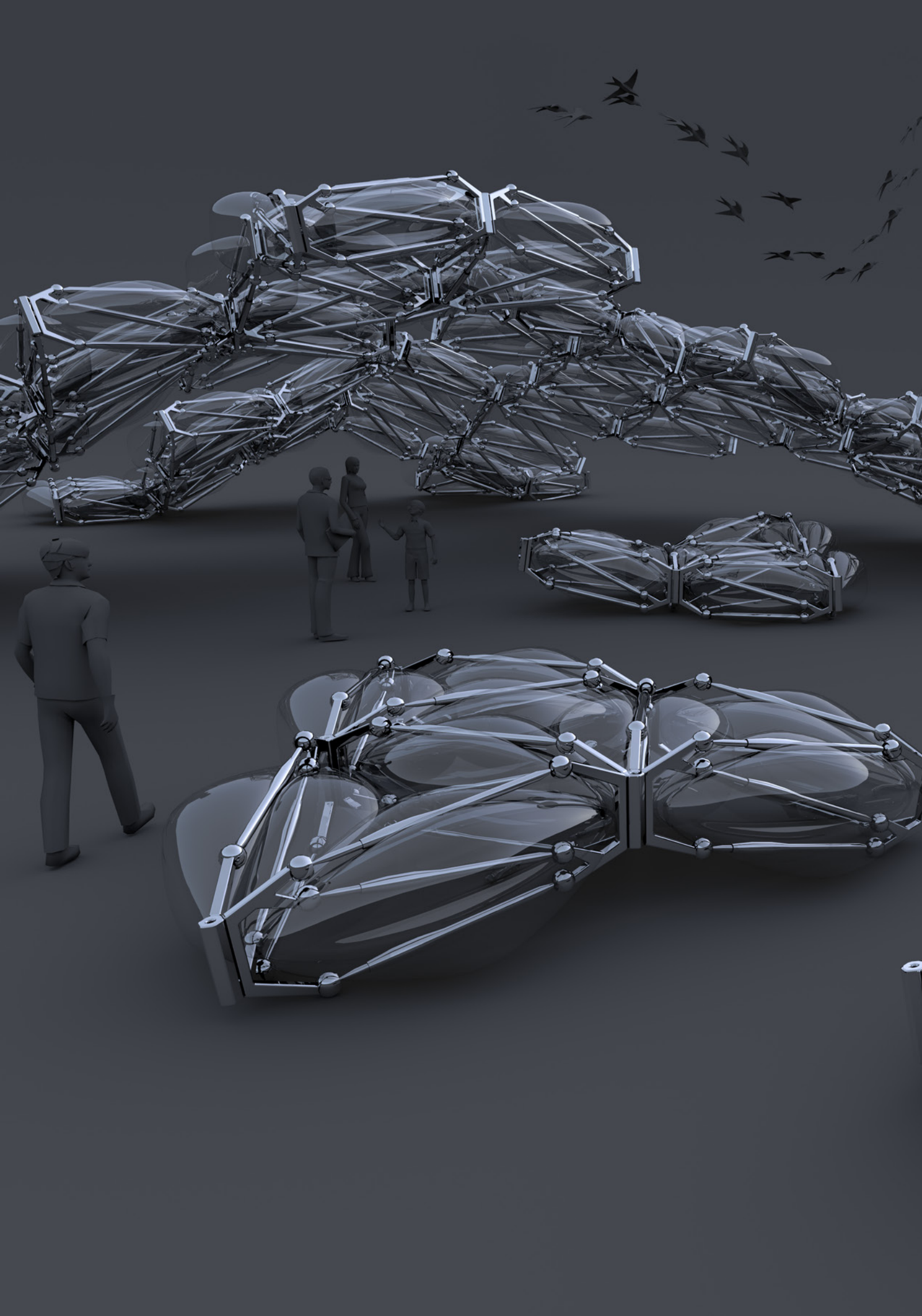
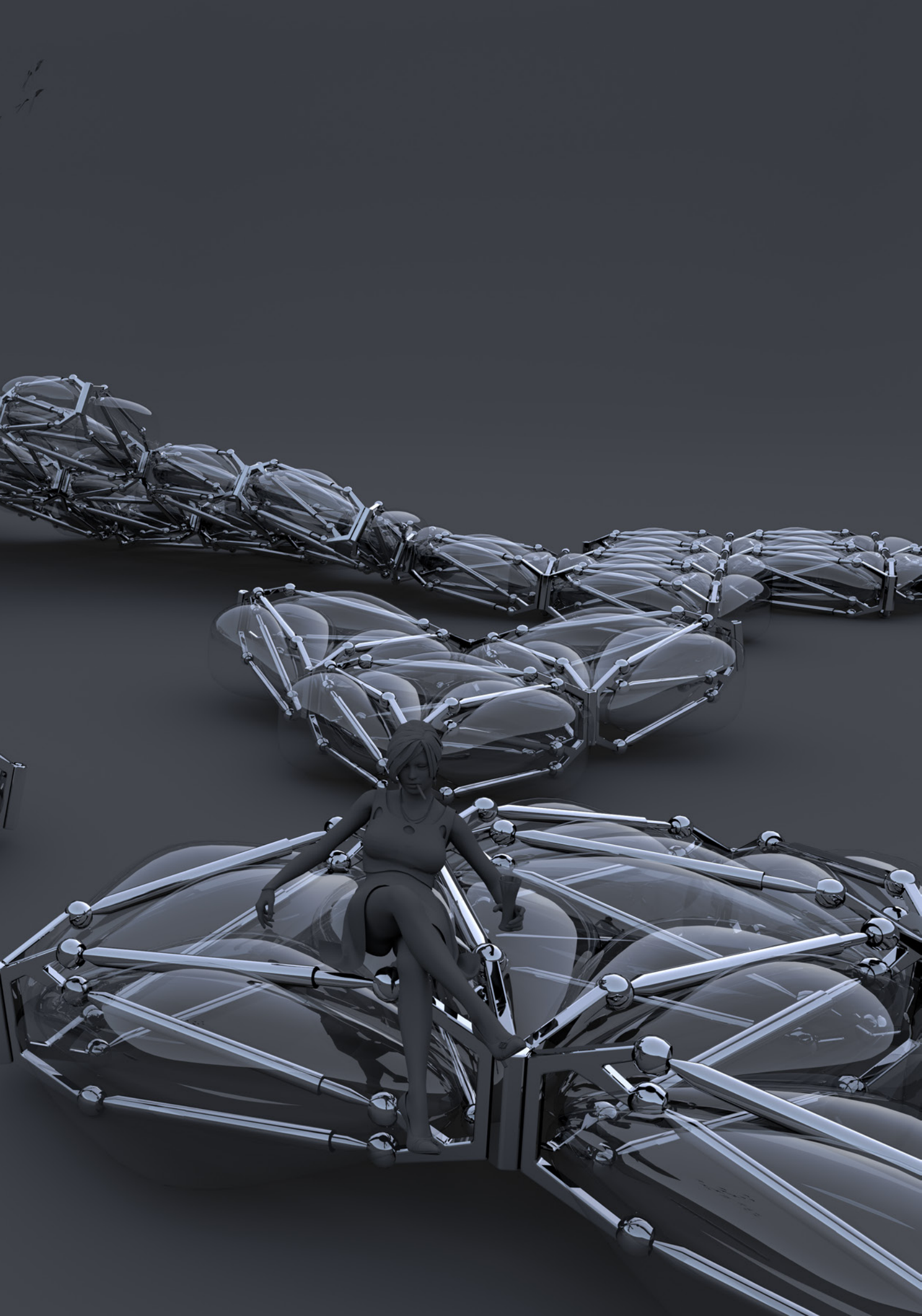


Abb.3.3.3-4 Weiterer Zwischenschritt





3.3.4 KETTENLINIE

Doch wie fügen sich die bAm schließlich zu Raum? Wie kann eine Aggregation aus einzelnen Modulen, die vorerst keine übergeordneten Regeln kennt, eine Entscheidung zum Bilden einer Tragstruktur treffen?

Zugrunde liegt ein einfaches Prinzip. Die Kettenlinie. Eine durchhängende Kette folgt dem Weg des geringsten Widerstandes. Dabei nimmt die gesamte Kette eine Form an, bei der für jedes der einzelnen Glieder eine günstige Situation entsteht. Dieses System kennt lediglich Normalkraft und ist eine ideale Lösung in Bezug auf eine Optimierung des Kräfteverlaufs.

Wird ein Glied durch äußere Einwirkung aus dieser Position gerissen, nimmt die gesamte Kette nach kurzer Zeit wieder die Form der Kettenlinie an. Mit etwas Phantasie kann die Kette als ein sehr einfaches Beispiel für ein sich selbst organisierendes System gesehen werden. Die jeweiligen Glieder verhandeln, mit der Schwerkraft als Beweggrund, die Kräfte und erreichen eine ideale Lösung. Ein sehr intelligentes und friedvolles System also, bei der die einzelnen Elemente ihre Aufgabe brüderlich teilen und großartiges leisten.

Wie bereits diskutiert, stellt die Umkehr dieser Geometrie einen idealen Bogen dar. Theoretisch wirkt auf diesen nur Normalkraft. Leider verliert dieses System aber durch die Umkehr diese Intelligenz. Wird nun ein Glied aus seiner Position gerissen, verliert das gesamte System seinen Frieden. Alle Nachbarn schieben ihre Last auf den einen Sonderling, der aus der Reihe tanzt. Das System kollabiert. Was also tun?

„Together we stand, divided we fall."

Pink Floyd, „Hey You", 1979

Analyse Kettenlinie
Problemstellung: Kette verliert Intelligenz
fallende Kette, Selbstorganisation, Problemstellung bei Bogen

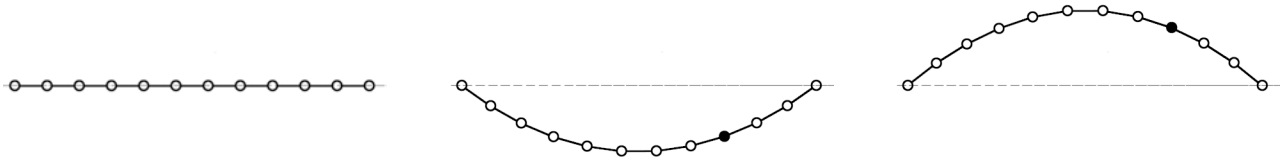


Abb.3.3.4-1 a,b,c Ausgangsposition

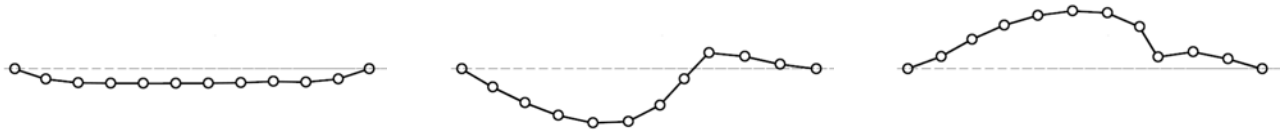


Abb.3.3.4-2 a,b,c 20 %

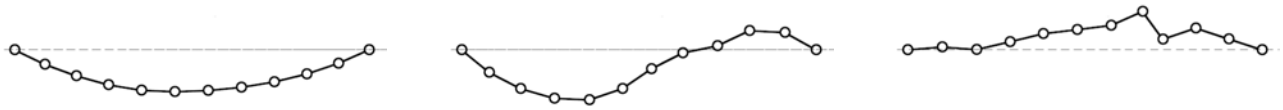


Abb.3.3.4-3 a,b,c 40 %

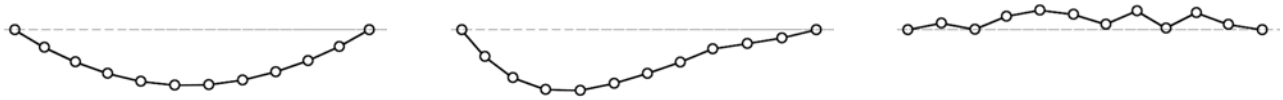


Abb.3.3.4-4 a,b,c 60 %

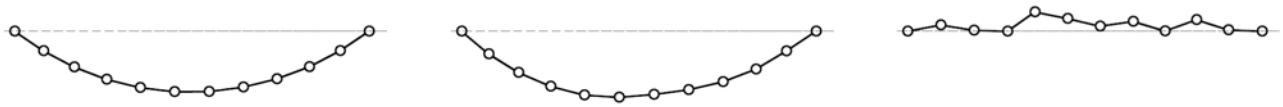


Abb.3.3.4-5 a,b,c 80 %

Abb.3.3.4-6 a,b,c Endposition

3.3.5 UMKEHRUNG

Verweis auf:
Selbst, Seite 174

Die Lösung ist einfach. Angenommen, die Knoten zwischen den einzelnen Gliedern wären nicht gelenkig sondern durch einen Motor ersetzt. Angenommen, jeder dieser Motoren hätte die Möglichkeit, seine aktuelle Belastung zu testen. Wie im Abschnitt Reaktion auf sich Selbst beschrieben . Dann könnte jeder Motor sich genau entgegen dem wirkenden Moment drehen. Die Kettenlinie entsteht wieder. Auf dem Kopf stehend.

Das Konzept der Tragstruktur funktioniert genau auf diesem Prinzip. Natürlich dreidimensional in Form einer Gitterschale. Beziehungsweise vierdimensional, wenn die Reaktion auf Einwirkungen von Außen einbezogen wird. Die gesamte Konstruktion würde beispielsweise automatisch auf sich ändernde Windlasten reagieren.

Angenommen wird, dass die Aggregation aus bAm sich von einer gewölbten Fläche in eine Stützfläche wie die von Frei Otto formen kann.

Wie umsetzen? Vorerst wird eine Cloth Simulation nachgeschrieben. Jesper Mosegaard erklärt alle Grundlagen auf seinem Blog. Der weitere Schritt ist denkbar einfach: Die Schwerkraft wird wie bereits angekündigt umgekehrt.

Jesper Mosegaard, „Mosegaards Cloth Simulation Coding“, 2009

Für die Module selbst ist der Ablauf ähnlich: Eine Kraftmessung der Motoren in Form von Druck führt zum Ziehen des Motors. Eine Kraftmessung in Form von Zug führt zum Drücken des Motors. Eine Kraftmessung unterhalb eines gewissen Spielraums bewegt den Motor nicht.

Die Folgeseite zeigt, wie sich die bAm zu einer Kuppel aufrichten. Zur Untersuchung der Kraftumkehr wurde ein Hängemodell im Maßstab 1:10 erstellt. Zusammengesetzt sind insgesamt 25 bAm mit 29 cm Radius. Das Gesamtausmaß des Modells ist 240 x 120 x 240 cm.

Folgeseite:
Zchng.3.3.5-1 Aufrichten der bAm

Folgeseite danach:
Abb.3.3.5-1 Hängemodell 1:10

Umkehr der Kette
Scriptsprache: Python
DCC: Blender

```python
import bpy

# Ausgangsgeometrie
Netzwerk = bpy.data.objects['Netzwerk']

# Globale Variablen
Schwerkraft = 0.0981
Verbindungs_Laenge = 1.2
Verbindungs_Staerke = 0.5

# Klasse für Agenten
class Agent:
    Liste = []
    def __init__(self, vertex):
        self.vertex = vertex
        self.Naechster_1 = False
        self.Naechster_2 = False
        self.Pinning = False

        Agent.Liste.append(self)

    def Kette(self):
        if not self.Pinning:
            # Schwerkraft
            self.vertex.co[2] -= Schwerkraft

            # Kräfte für Verbindungen
            v_1 = self.Naechster_1.vertex.co - self.vertex.co
            Staerke = (v_1.length - Verbindungs_Laenge) * Verbindungs_Staerke
            v_1 = v_1 * Staerke

            v_2 = self.next_2.vertex.co - self.vertex.co
            Staerke = (v_2.length - Verbindungs_Laenge) * Verbindungs_Staerke
            v_2 = v_2 * Staerke

            self.vertex.co = self.vertex.co + v_1 + v_2

# Ausgangsgeometrie einlesen
for vertex in Netzwerk.data.vertices:
    Neuer_Agent = Agent(vertex)

# Pinning aufsetzten
Agent.Liste[0].pinning = True
Agent.Liste[4].pinning = True

# Nachbarn definieren
for Agent in Agent.list:
    if not Agent.pinning:
        Agent.Naechster_1 = Agent.Liste[Agent.vertex.index -1]
        Agent.Naechster_2 = Agent.Liste[Agent.vertex.index +1]

# Echtzeitteil
for Agent in Agent.list:
    Agent.Kette()
```

Scp.3.3.5-1 Umkehr Kettenlinie

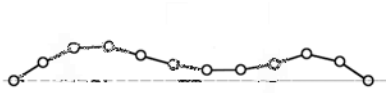


Abb.3.3.5-2 In Fehlhaltung

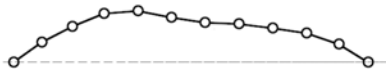


Abb.3.3.5-3 Korrektur

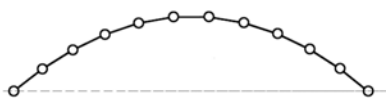
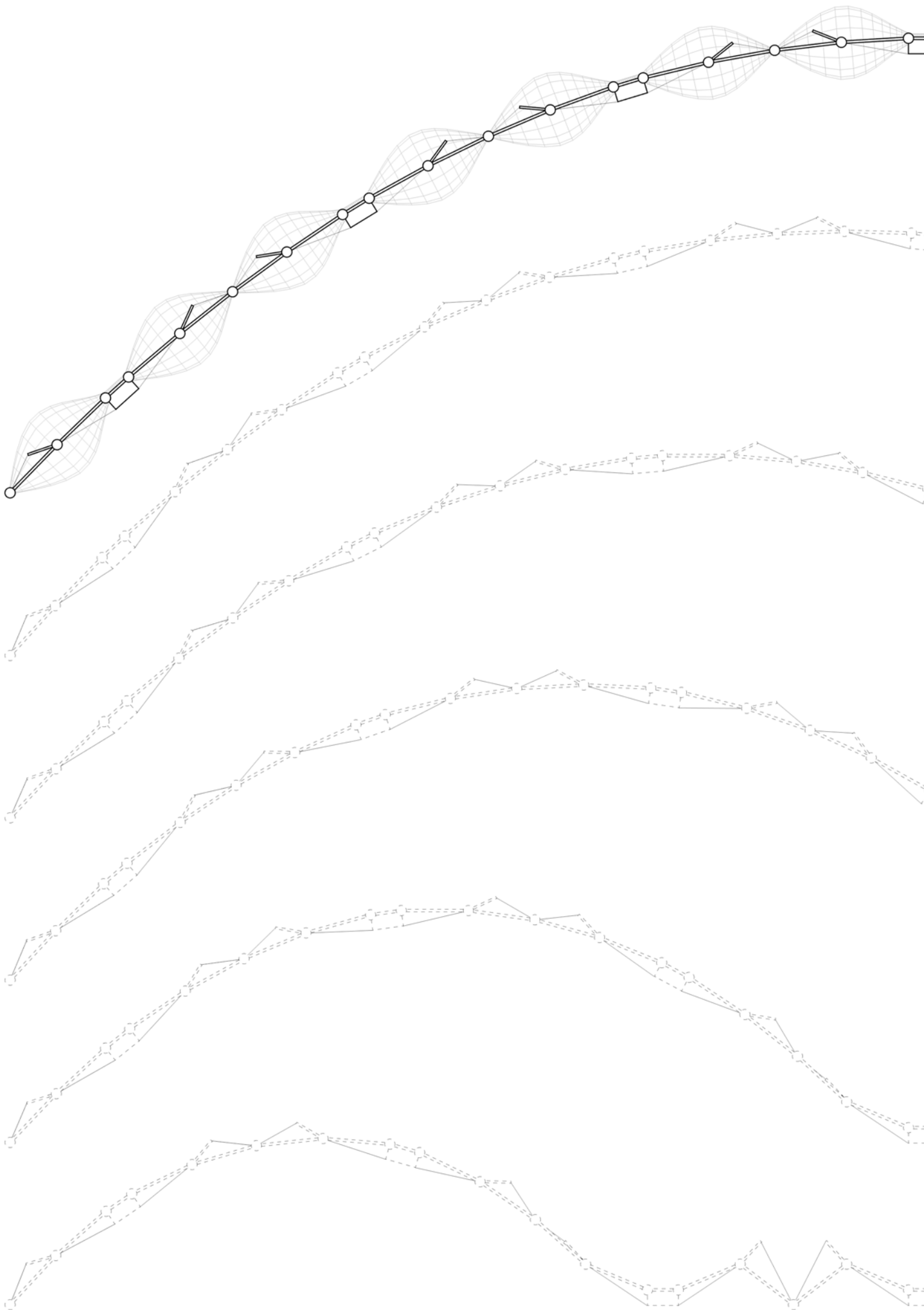
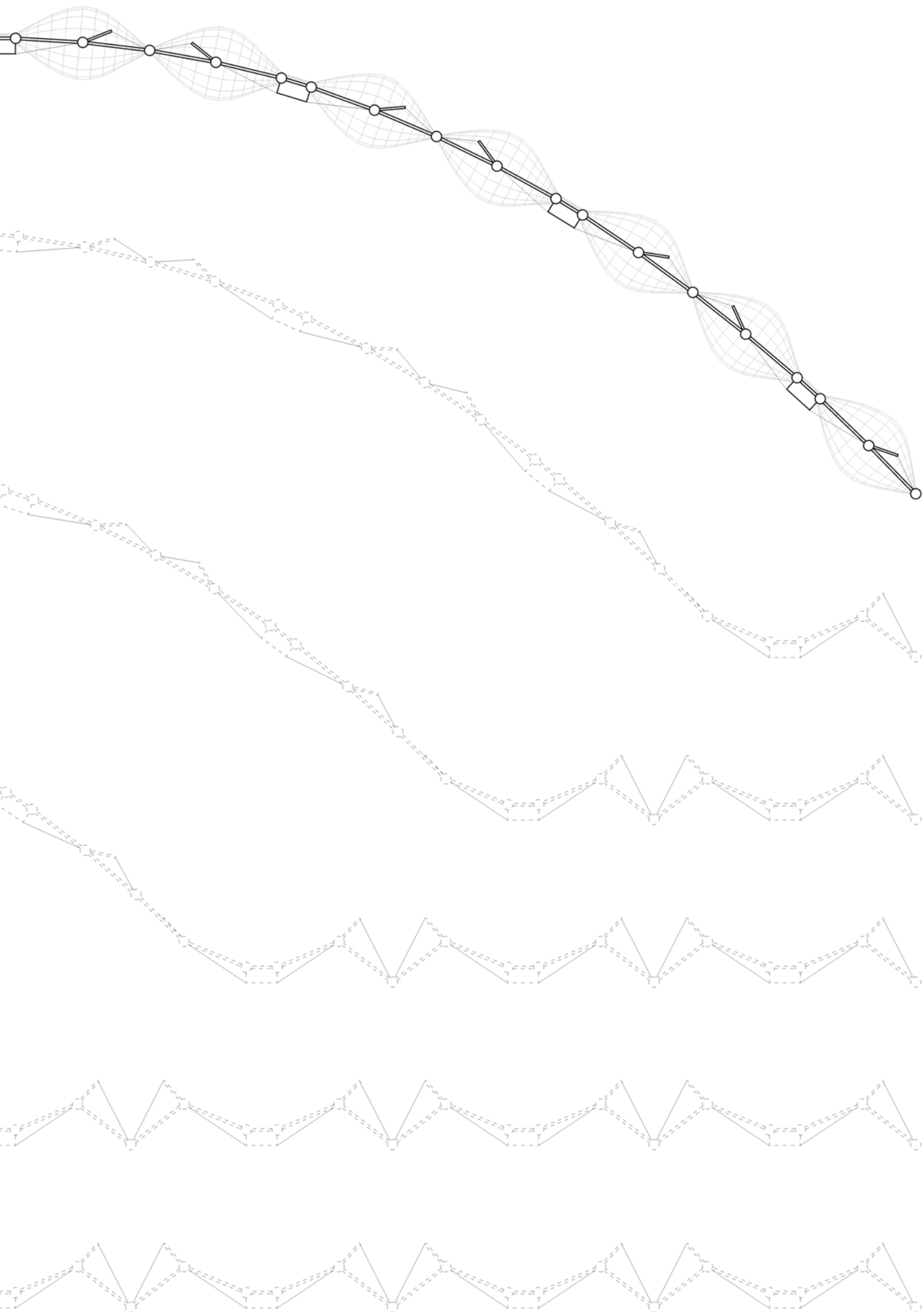
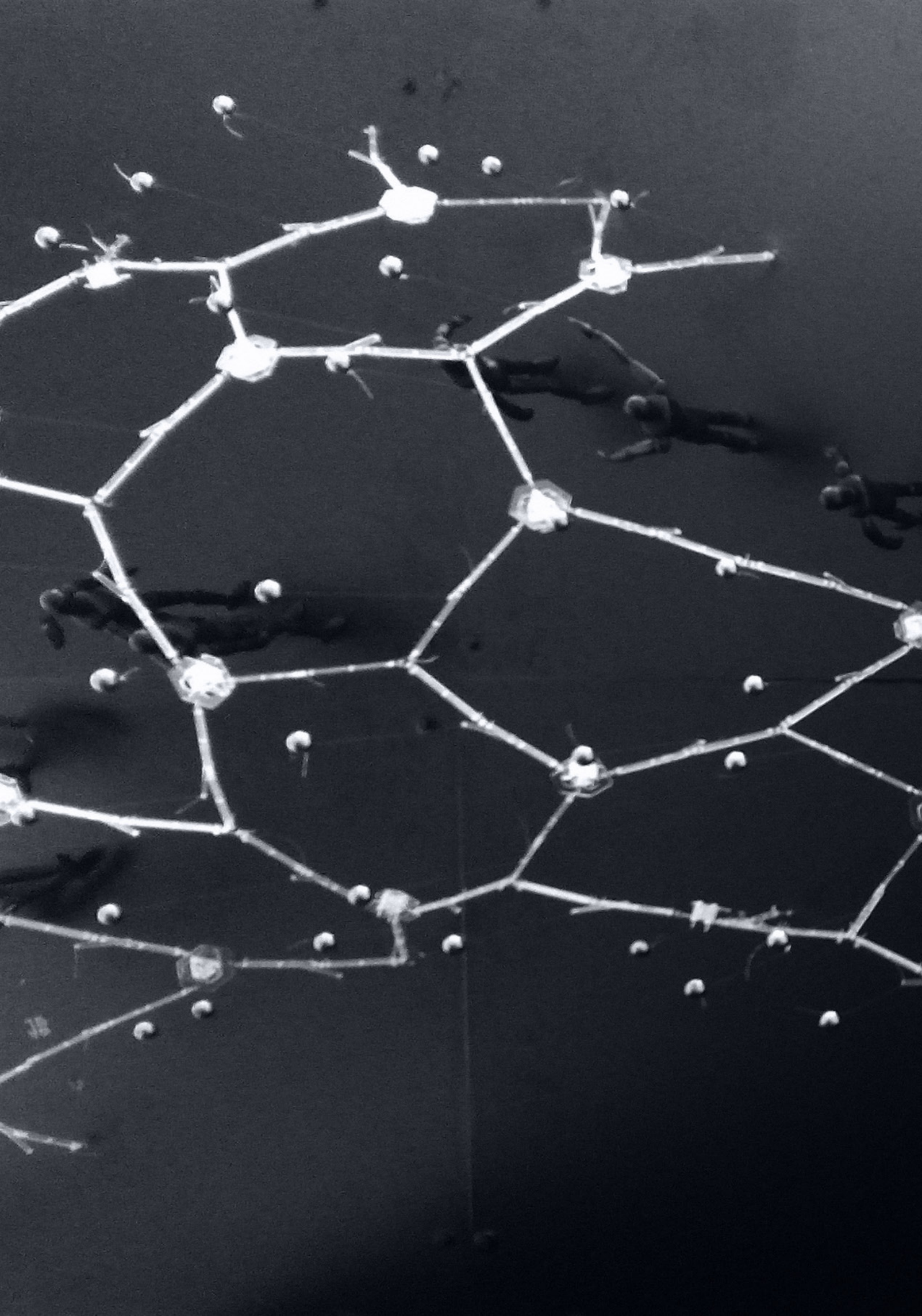


Abb.3.3.5-4 Günstigere Form







3.3.6 MENSCHEN

„Don't talk to strangers, cause they're only there to do you harm."

Ronnie James Dio, „Holy Diver", 1983

Anthony Zhang, „Speech Recognition", 2016

Jonathan Duddington, „eSpeak text to speech", 2014

Dieser Abschnitt beschäftigt sich mit der Interaktion zwischen Menschen und bAm. Auch wenn die Module sich selbst organisieren und eigene Interessen verfolgen, können Benutzer ihr Verhalten auf verschiedene Art beeinflussen. Ein Weg dazu ist Sprache. Die Module warten dabei stets auf das Schlüsselwort bAm um sich angesprochen zu fühlen. Alle anschließenden Inputs werden als Befehl interpretiert.

Mögliche Befehle sind dabei: das Bewegen der Module, die Abfrage diverser Informationen zur Umgebung und weitere.

Zur Spracherkennung sind freie Pakete verfügbar. Der genauere Fokus im Rahmen dieser Arbeit liegt auf Pocketsphinx und einer Lösung von Google. Der Vorteil von Pocketsphinx ist, dass die Bearbeitung ohne Internetzugang geschehen kann. Nachteil wiederum ist, dass das Modul Rechenleistung in Anspruch nimmt. Die Google Speech Recognition berechnet die Spracherkennung auf den Servern von Google. Der Script zeigt eine Umsetzung mittels Google und basiert auf der Python-Bibliothek von Anthony Zhang.

Umgekehrt ist natürlich auch eine Sprachausgabe möglich. Diese geschieht über das Modul Espeak von Jonathan Duddington. Bei Wartungsarbeiten ist bei Abnahme der Abdeckung der bAm auch ein Analogdisplay abzulesen.

Hören und Sprechen
Scriptsprache: Python
Plattform: Raspberry Pi 2

```python
import speech_recognition as sr
from espeak import espeak

espeak.set_voice("de")

Befehl = ["ben", "bäm", "BÄM"]

def say(text):
    try:
        espeak.synth(text)
        print(text)
    except:
        pass

r = sr.Recognizer()

say("Sprachmodul aktiv")

def Spracherkennung():
    with sr.Microphone() as source:
        audio = r.listen(source)

    try:
        reco = str(r.recognize_google(audio, language="de"))
        print(reco)
        return(reco)

    except sr.UnknownValueError:
        pass

    except sr.RequestError as e:
        pass

while True:
    if Spracherkennung() in Befehl:
        say("Was kann ich tun")
        Prozess = Spracherkennung()
        say(Prozess)
```

Scp.3.3.6-1 Spracherkennung und Ausgabe

Neben dem Erkennen von Sprache ist auch das Erkennen von Personen integriert. Facedetection bezeichnet das Erkennen von Gesichtern. Facerecognition erlaubt auch ein Wiedererkennen von Personen. Vorerst nicht namentlich, aber anhand deren Gesichtsproportionen. Das Thema des Datenschutzes muss geklärt werden. Aus technischer Sicht können die bAm bestimmte Daten, wie Name, Alter, Gewicht und weitere, mittels der Spracheingabe erfragen. Ob das gewünscht ist, bleibt vorerst offen. Nachteile und Potenziale stehen aus Sicht des Datenschutzes im Zielkonflikt. Grundsätzlich könnten die bAm auch einer Überwachung des öffentlichen Raumes dienen. Möglicherweise hat auch die Präsenz der bAm an sich eine abschreckende Wirkung.

„Knight Rider, ein Mann und sein Auto kämpfen gegen das Unrecht."

Intro der Serie, „Knight Rider", 1982

Das Speichern der Gesichtsproportionen würde jedenfalls ausreichen, um die Funktionalität der bAm grundlegend zu erweitern. So könnten mittels maschinellen Lernens die Benutzereingaben analysiert und schließlich zu einem gewissem Grad auch prognostiziert werden.

Verweis auf:
Maschinelles Lernen, Seite 200

Derzeit verfügen die bAm über eine einfache Facedetection, die auf dem Blog von Carlo Mascellani basiert. Diese Implementierung erlaubt das Absuchen eines Bildes innerhalb 8 Sekunden am Raspberry Pi 2.

Carlo Mascellani, „Face detection with Raspberry Pi", 2015

Gesichter erkennen
Scriptsprache: Python
Plattform: Raspberry Pi 2

```
import cv2

# gespeicherte Bilddatei laden
Bild = cv2.imread("Bild.jpg", 1)

# Cascade für Gesichtserkennung laden
Cascade = cv2.CascadeClassifier('/usr/share/opencv/↩
    ↪ haarcascades/haarcascade_frontalface_alt.xml')

# In Graustufen konvertieren
Graustufenbild = cv2.cvtColor(Bild, cv2.COLOR_BGR2GRAY)

# Gesichter erkennen
Gesichter = Cascade.detectMultiScale(Graustufenbild, 1.1, 5)

print(str(len(faces)), "Gesichter gefunden")

# Rechteck um jedes Gesicht zeichnen
for (x,y,w,h) in Gesichter:
    cv2.rectangle(Bild, (x,y), (x+w,y+h), (0,255,0), 2)

# Bild speichern
cv2.imwrite('Ergebnis.jpg', Bild)
```

Scp.3.3.6-2 Gesichtserkennung

Angesprochen wurde bereits die Möglichkeit, dass die bAm sich mit dem Internet verbinden können. Über das Netzwerkprotokoll SSH kann auf die Module zugegriffen werden. Selbstverständlich sind die bAm damit auch über das Internet steuerbar. Abzuwägen ist grundsätzlich, ob das die Module nicht als gesamte Aggregation anfällig macht. Schließlich können so alle Module auch negativ beeinflusst werden.

Möglich ist, dass einige der bAm mit mobilem Internet ausgestattet sind. Diese Module können dann als WLAN-Router funktionieren. Verbindet sich ein Modul in das Internet, kann die Verbindung an alle Nachbarn weiter gegeben werden. Die Module werden so zu Signalverstärkern. Möglicherweise wird die Verbindung auch mit Nutzern im öffentlichen Raum geteilt, was attraktiv sein kann.

Das Potenzial läge darin, dass die Module Informationen aus dem Internet abfragen können, welche die Umgebung betreffen. Das wird im Abschnitt Kontext noch genauer diskutiert.

Verweis auf:
Urban, Seite 198

Ein weiterer Bonus wäre das Auslagern von Rechenleistung. Die Module könnten beispielsweise Rechenleistung der Server von Google in Anspruch nehmen. Umgekehrt, können die Module bei Stillstand auch ihre Rechenleistung teilen, um gemeinsam Kalkulationen abzuarbeiten. Besonders bei einer hohen Anzahl von Modulen summiert sich diese potenzielle Rechenleistung schnell.

Am Rande bemerkt: Viele Simulationen und Visualisierungen dieser Arbeiten haben hohe Rechenleistung erfordert und wurden mittels Google-Cloud auf Google-Servern gerendert.

SSH-Verbindung
Auslagern von Rechenleistung auf Google Cloud
Zugriffsart: Terminal

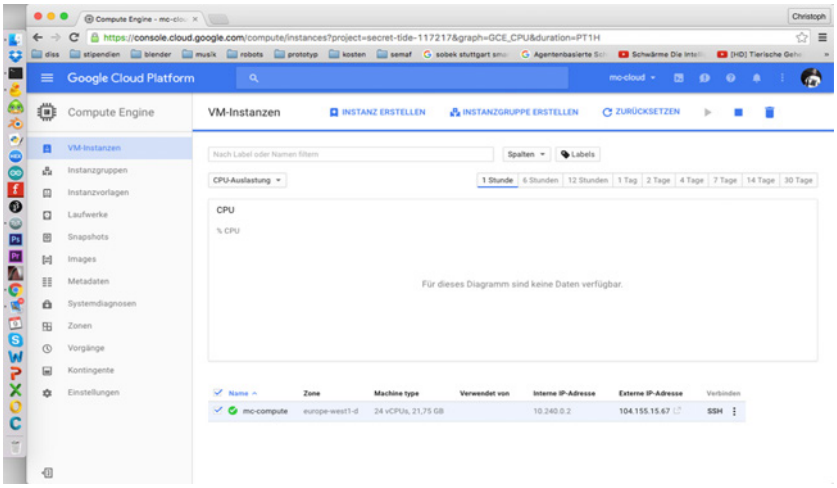


Abb.3.3.6-1 Starten einer externen virtuellen Maschine

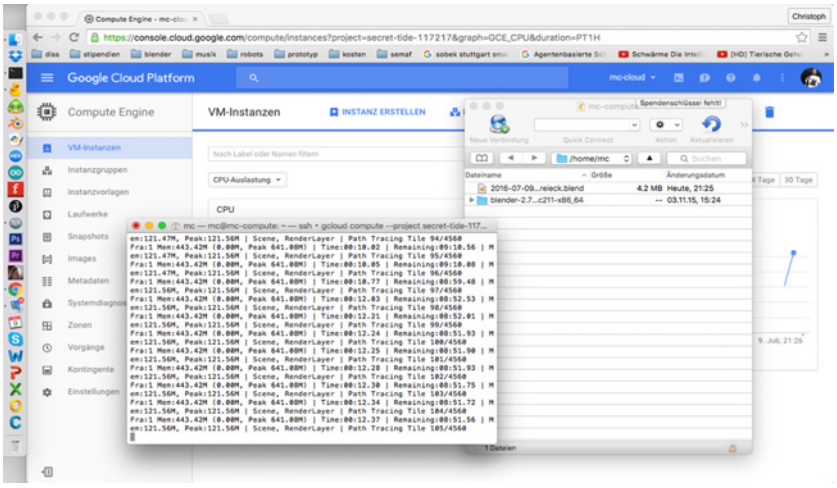


Abb.3.3.6-2 Terminalausgabe der Berechnung

3.3.7 U R B A N

Das Registrieren von Umgebung erlaubt den bAm auf gebauten Kontext zu reagieren. Eine weitere Möglichkeit ist der Zugriff auf Informationen aus dem Internet. So können beispielsweise Daten wie Opendata in Wien, als Grundlage für die Reaktion auf Menschenmengen und Weiteres verwendet werden. Die Ankunft von U-Bahnen beispielsweise. Wäre es möglich, dass die bAm in bestimmen Fällen auch zur Deeskalation oder Prävention von Panik innerhalb großer Menschenmengen beitragen?

Weitere nutzbare Daten sind jedenfalls die Zeit, das Wetter und viele mehr. Versucht wird, anhand dieser Informationen auch ein Daten-Mining zu betreiben, das mittels maschinellem Lernen analysiert wird und eine immer effizienter vorausschauende Reaktion möglich macht.

Verweis auf:
Maschinelles Lernen, Seite 200

Die Simulation zeigt, wenn auch abstrakt, wie die Reaktion der bAm im städtischen Kontext aussehen könnte. Diese Simulation basiert auf einem Boid-System wie im Abschnitt Boids beschrieben.

Verweis auf:
Boids, Seite 145

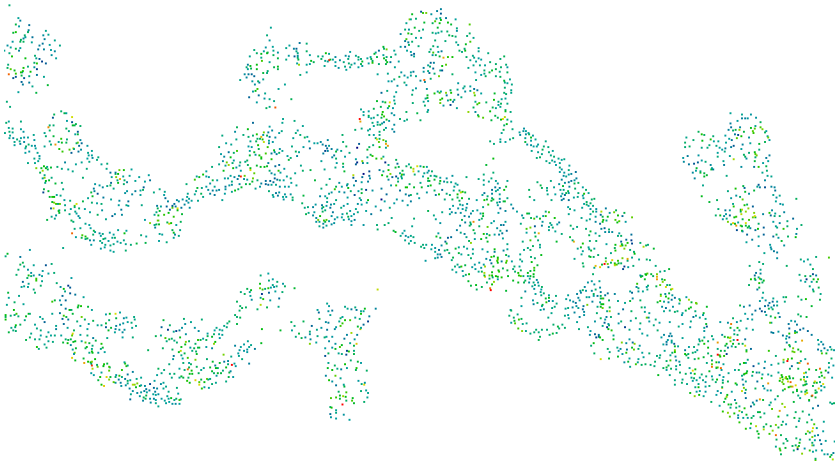


Abb.3.3.7-1 Freie Entwicklung

Urbanität
Boid-System im städtischen Kontext
DCC: Blender

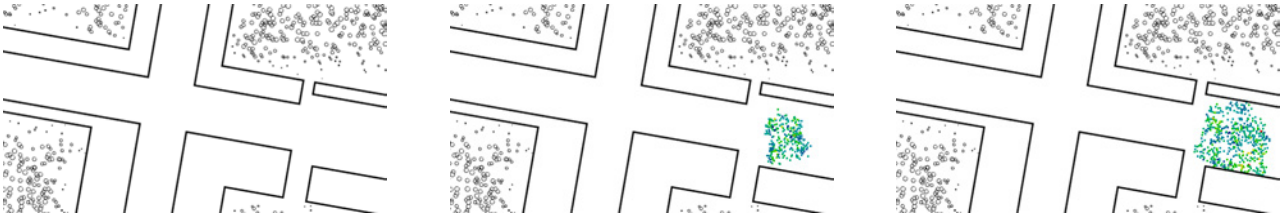


Abb.3.3.7-2 a,b,c Ausgangsposition


Abb.3.3.7-3 a,b,c 20 %

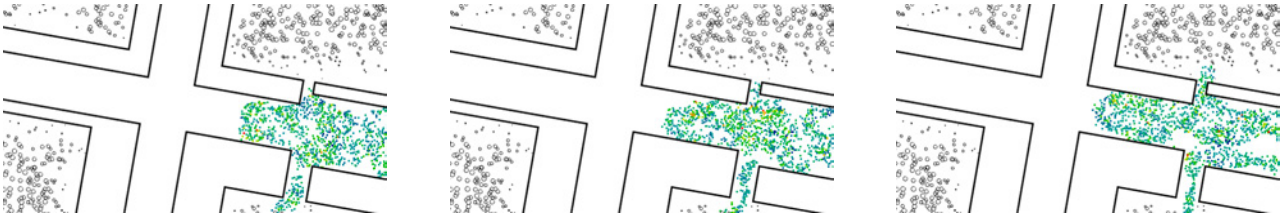


Abb.3.3.7-4 a,b,c 40 %


Abb.3.3.7-5 a,b,c 60 %


Abb.3.3.7-6 a,b,c 80 %

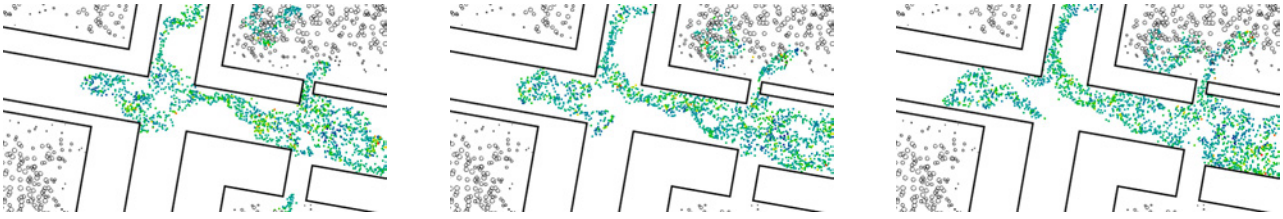


Abb.3.3.7-7 a,b,c Endposition

3.4 MASCHINELLES LERNEN

Dieser Abschnitt beschäftigt sich mit der künstlichen Intelligenz der Module. Insbesondere geht es darum, wie die bAm aus bereits vergangenen Situationen künftige Szenarios ableiten können.

3.4.1 ORANGE

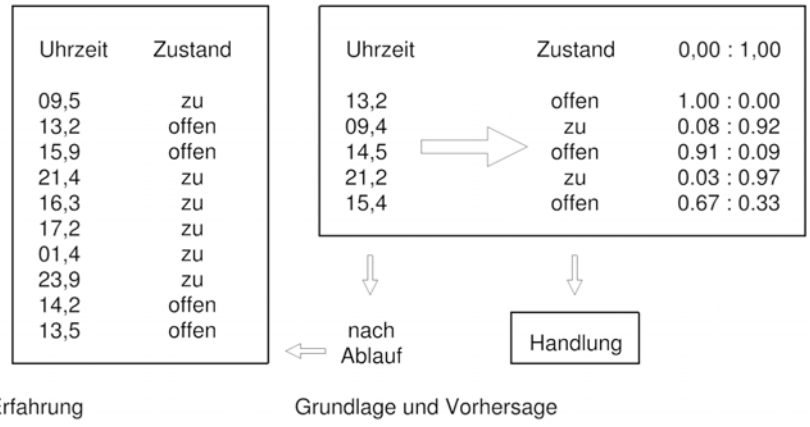
Die Problemstellung ist: Ein bAm erhält im Laufe der Zeit mehrmals die Anweisung, sich sich zu öffnen oder sich zu schließen. Bei einer Variante mit Pneus wird Luft nachgepumpt oder abgesaugt. Als zweite Information wird angegeben, wann der Befehl erteilt wurde. Ziel ist nun, dass der bAm selbstständig Rückschlüsse ziehen kann, in welchem Zusammenhang der Öffnungszustand mit der Uhrzeit steht. Erweitert werden kann dieses Modell mit Informationen über den Nutzer, den Ort, die Wetterinformationen und viele Weitere.

Zum Finden des geeigneten Lösungsansatzes und zur Visualisierung der Daten wird das Programm Orange verwendet.

Verweis auf:
Orange, Seite 275

Angegeben wird, wie im ersten Screenshot zu sehen, die Exceldatei als Input. Die Visualisierung im zweiten Bild zeigt Uhrzeit und Öffnungszustand als Diagramm. Zu erkennen ist eine Gruppe mit vier blauen Punkten, bei denen das Modul in der Vergangenheit geöffnet war. Mittels einer Regression von Nearest Neighbors kann nun das Programm eigenständig jene Gruppe finden und definieren.

Der letzte Schritt zeigt schließlich eine Prognose von eingegeben Daten und die Treffergenauigkeit des maschinellen Lernens.



Tab.3.4.1-1 Daten

Maschinelles Lernen
Finden eines geeigneten Algorithmus
Programm: Orange

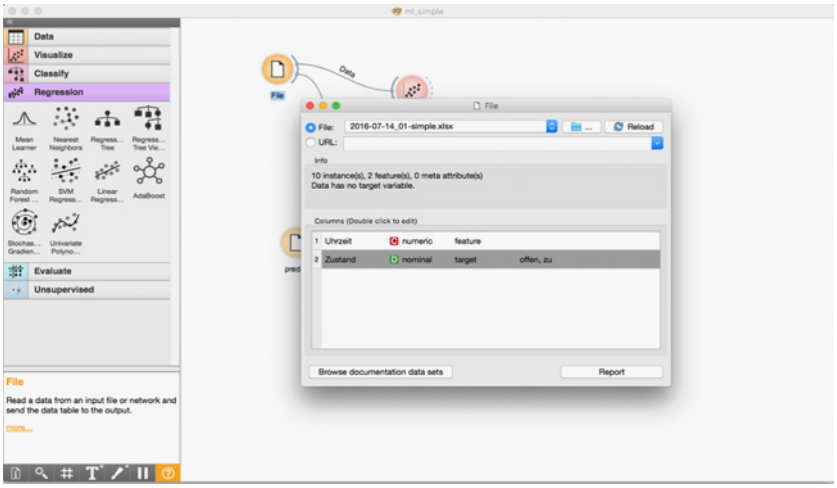


Abb.3.4.1-1 Eingabe

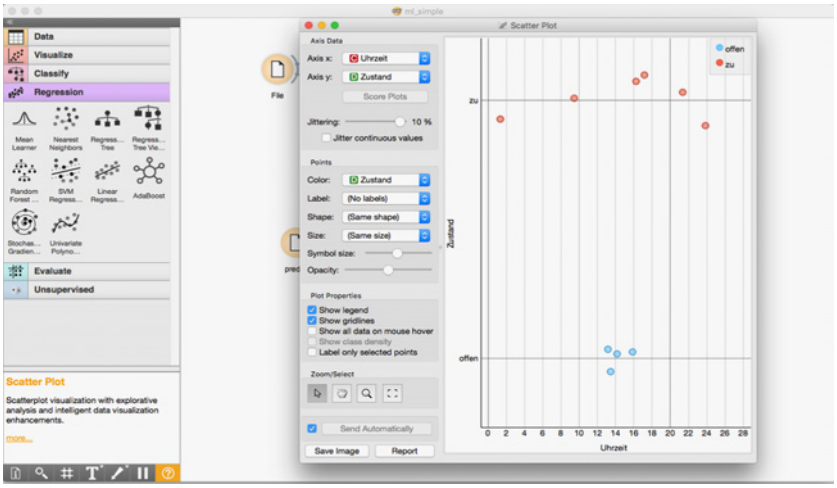


Abb.3.4.1-2 Analyse

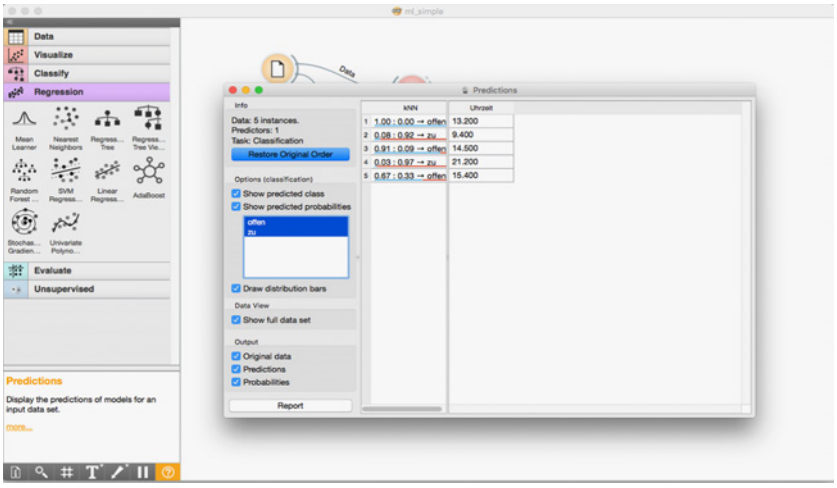


Abb.3.4.1-3 Lösung mittels graphischer Benutzeroberfläche

3.4.2 PYTHON

Der Script zeigt die Umsetzung des maschinellen Lernens im Script. Selbstverständlich ist das nur ein sehr einfaches Beispiel. Mit gesundem Menschenverstand ist nachvollziehbar, dass das Modul zur Mittagszeit geöffnet ist und sonst geschlossen. Jedoch können sich diese Parameter ändern. In diesem Fall ist der bAm durch den rechts gezeigten Script in der Lage, diese Veränderung selbstständig zu erkennen und darauf zu reagieren.

Orange, „Orange Data Mining Library“, 2015

Maschinelles Lernen
Anwendung auf Script
Programm: Orange

```
import Orange

# Lädt die gespeicherten Daten in Form von Tabelle
Daten = Orange.data.Table("2016-07-14_01-Daten")

# Lädt Testdaten zur Evaluierung als Tabelle
Test = Orange.data.Table("2016-07-14_01-Daten-Vorhersage")

# Auswahl des Algorithmus als Grundlage des maschinellen Lernens
knn = Orange.classification.KNNLearner(weights="distance")

# Trainiert das maschinelle Lernen
classifier = knn(Daten)

# Geht durch sämtliche Einträge der Test-Tabelle und stellt eine Prognose
for i in range((len(Test))):
    Ergebnis = classifier(Test[i], Orange.classification.Model.ValueProbs)
    von = Ergebnis[1][0][0]
    bis = Ergebnis[1][0][1]
    if Ergebnis[0] == 1:
        Zustand = "zu"
    else:
        Zustand = "offen"
    print("Prognose:", von, ":", bis, Zustand)
```

Scp.3.4.2-1 Anwendung im Script

3.5 DETAILS

Die bAm sind so einfach konstruiert wie möglich. Beispielsweise könnten die Module als Bausatz geliefert werden, der lediglich zusammengefügt wird. Auch in Folge der weiteren Nutzung können defekte Teile mit wenig Aufwand ausgetauscht werden. Es wird möglich, dass Nutzer selbst die bAm aufbauen und weiter verbessern.

Der verwendete Nirosta mit der Legierung 1.4301 ist der meist produzierte Edelstahl. Das verspricht eine langfristige Verfügbarkeit und erlaubt eine effiziente Wiederverwendbarkeit. Allerdings können bei einem größeren Wandel des Stahlmarktes und daran gekoppelte Preise, auch andere Materialien eingesetzt werden. Die pragmatische Gestaltung der Grundgeometrie erlaubt das.

Der Open-Source-Gedanke spiegelt sich in erster Linie an der verwendeten Computertechnik wieder. Die Software der bAm kann kostenlos heruntergeladen und variiert werden. Das gleiche gilt für zugrunde liegende Bibliotheken und das verwendete Raspbian Linux. Auch Elektronik verbessert und verändert sich schnell. Der eingesetzte Raspberry Pi sowie das Arduino sind auswechselbar, um neuere Versionen mit beispielsweise mehr Leistung oder neueren Schnittstellen verfügbar zu machen.

3.5.1 METALLTEILE

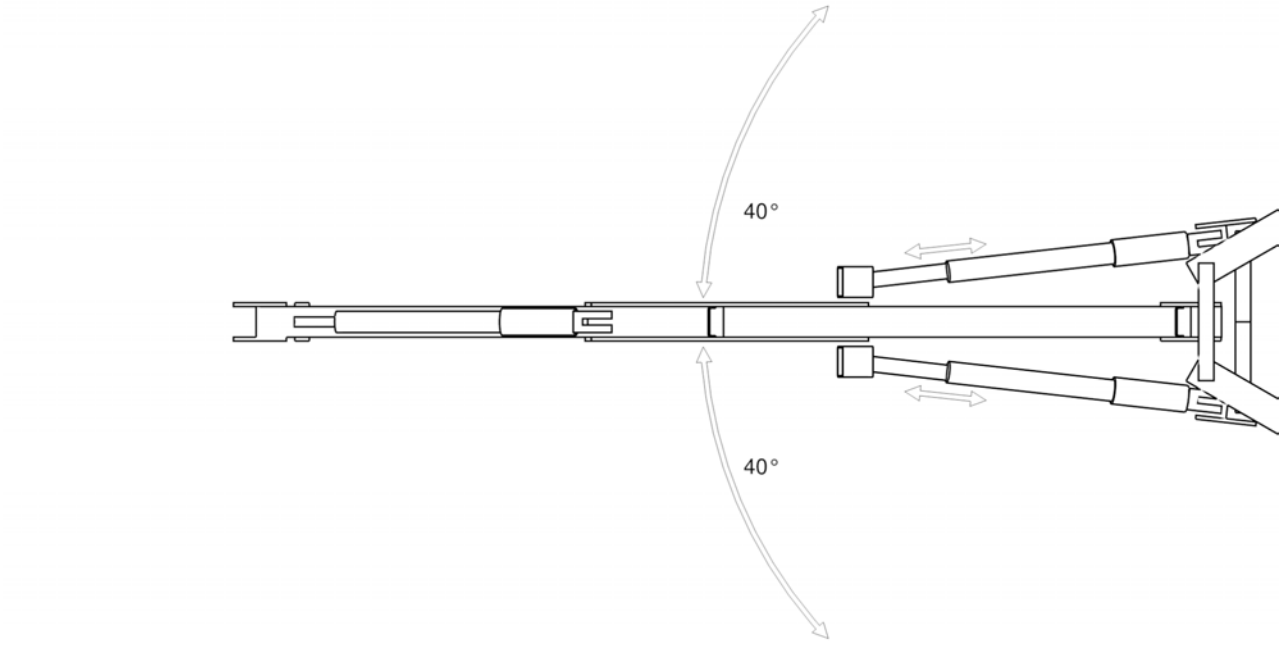
Einleitend wird der Aufbau der Füße in Grundriss und Ansicht gezeigt. Dargestellt ist der maximale Bewegungsraum sowie die Freiheitsgrade der Gelenke. Mit drei Linearaktuatoren kann der Fuß sich in alle drei Richtungen des kartesischen Raumes bewegen. Der Motor des Unterschenkels erzeugt ein Ausfahren und Einklappen des Gelenkes. Es ist lediglich eine Verbindung mit einem Freiheitsgrad notwendig. Die beiden Motoren des Oberschenkels bewegen das Gelenk auch seitlich. Es ist eine Verbindung gefragt, die in zwei Freiheitsgraden beweglich ist.

Verweis auf:
Freiheitsgrade, Seite 124

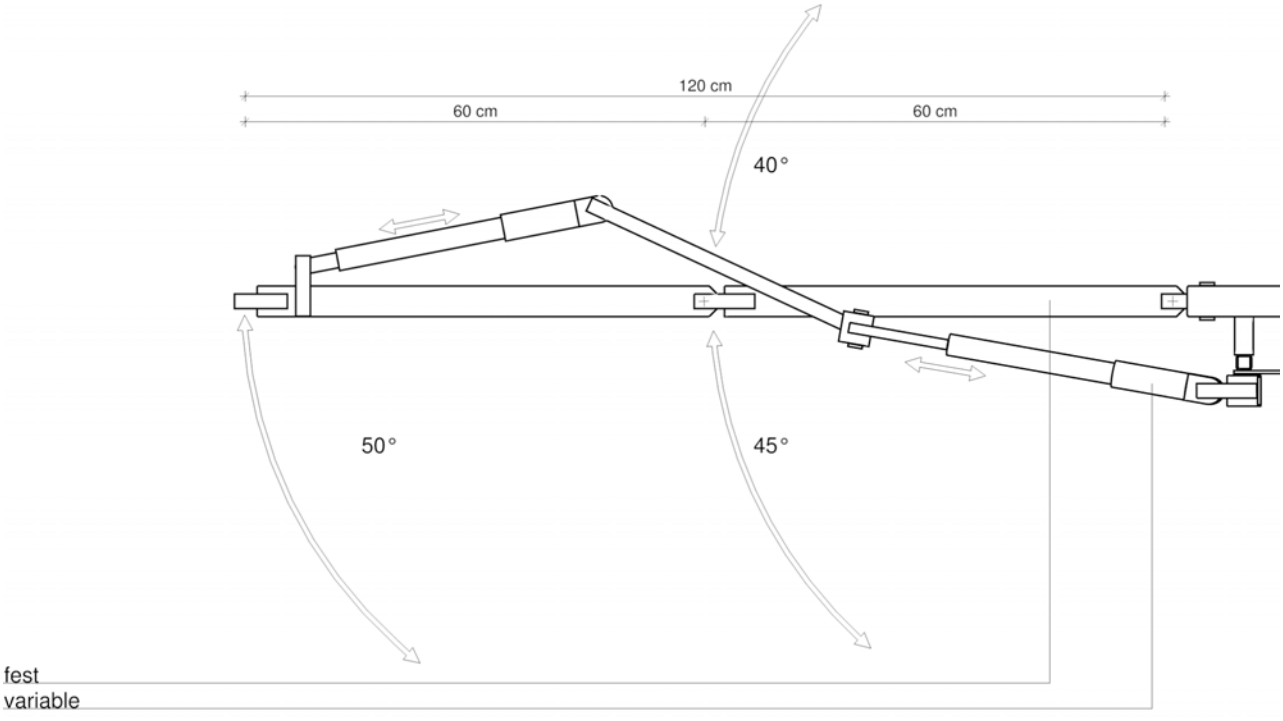
Wichtig war, die Momente innerhalb der Konstruktion zu minimieren. Die Auflager der Motoren zielen so weit wie möglich auf die Auflager der Träger.

Folgeseite:
Abb.3.5.1-1 Rendering

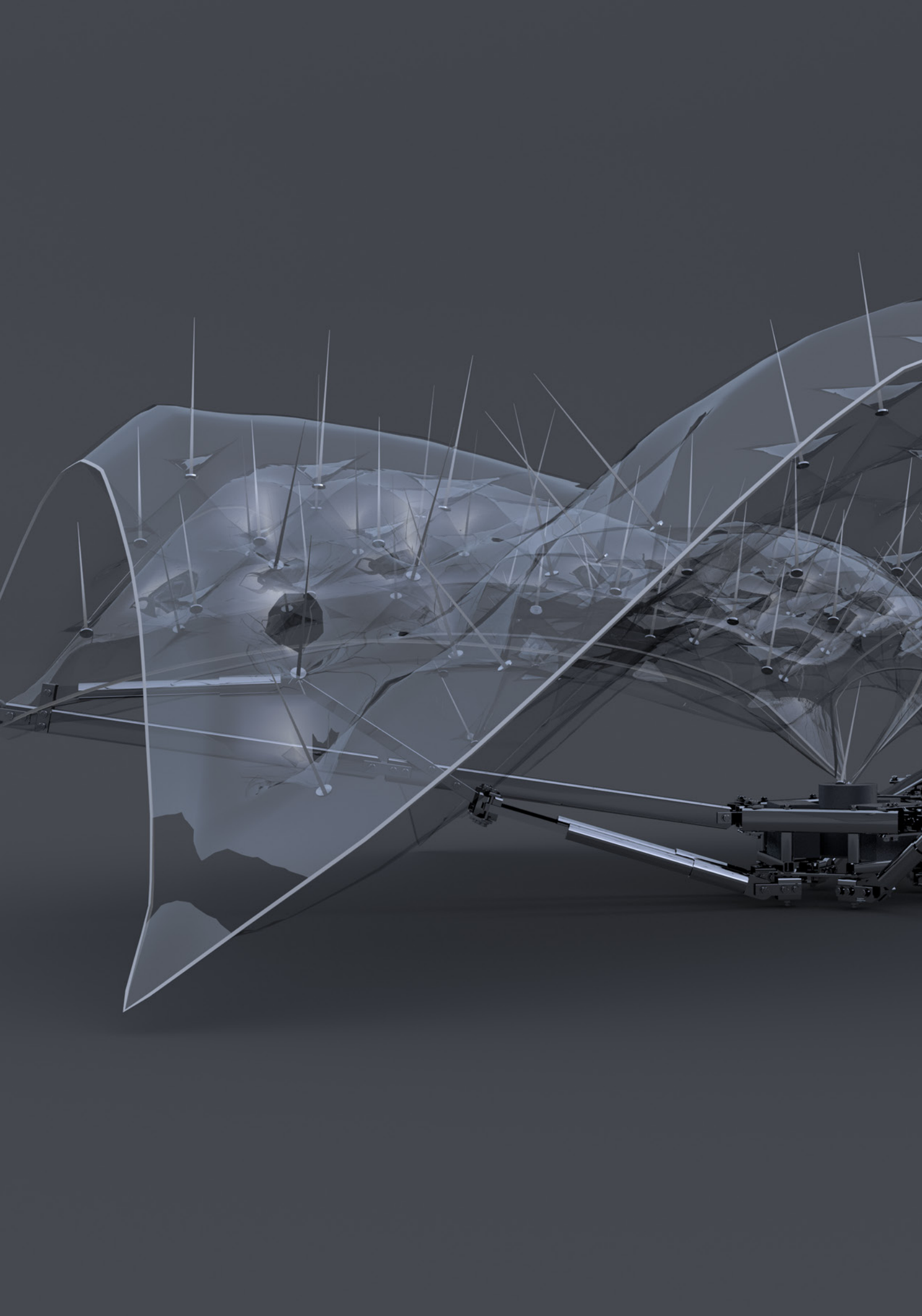
Schemadetail Metallbau
Fokus auf einen Fuß
in Grundriss und Ansicht

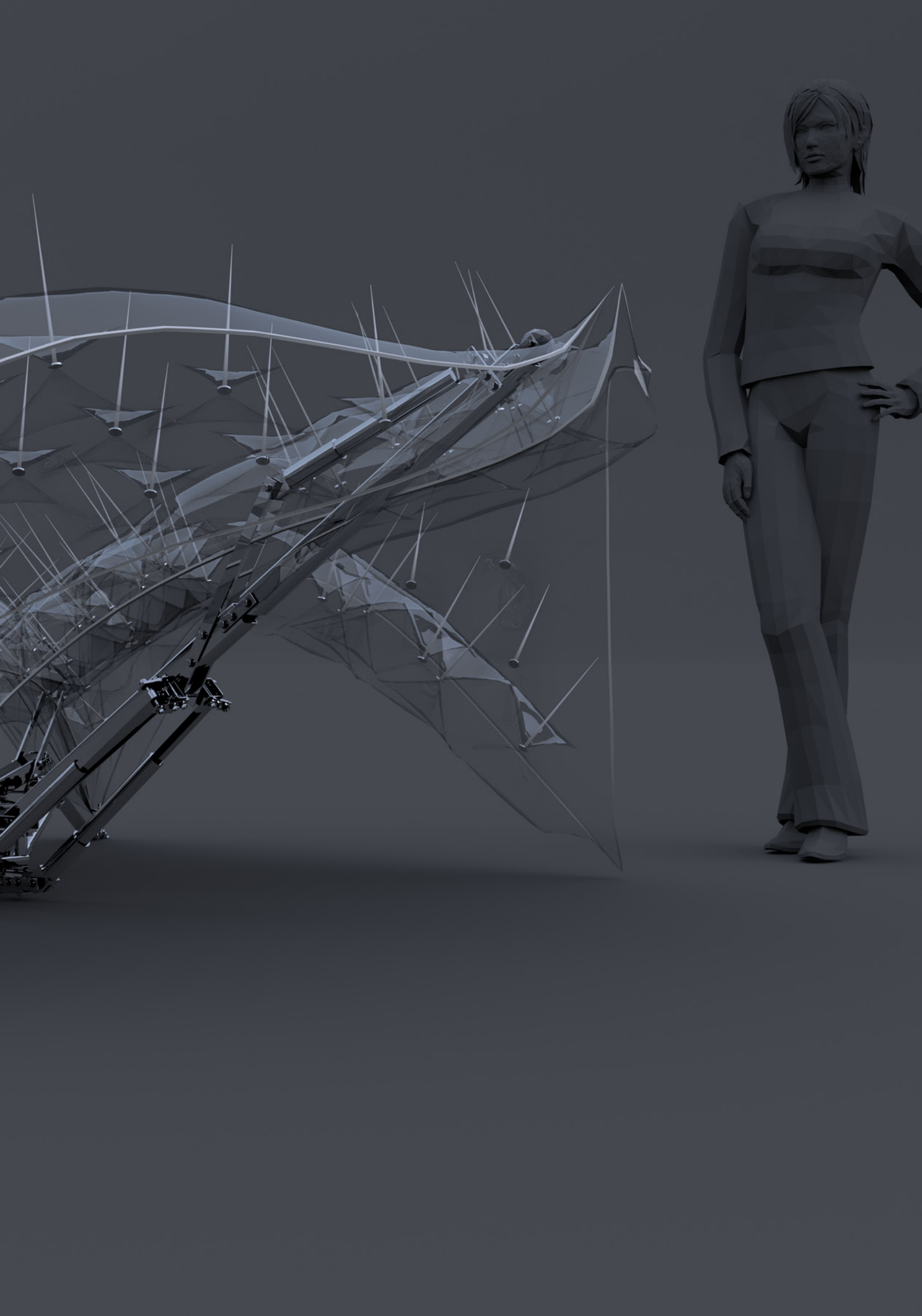


Zchng.3.5.1-1 Grundriss



Zchng.3.5.1-2 Ansicht

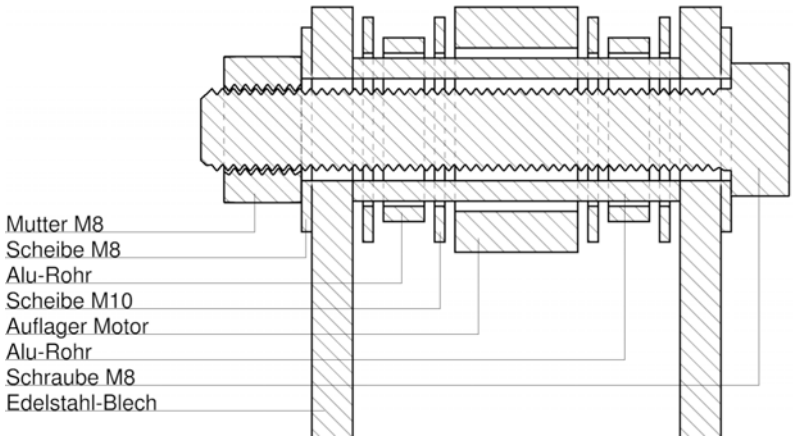




Verweis auf:
Verbindung, Seite 214

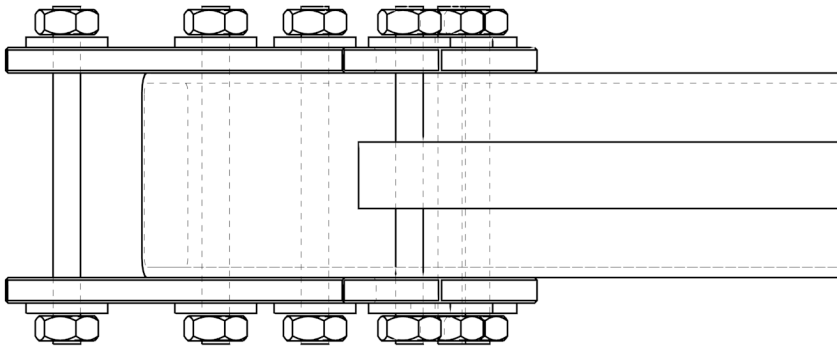
Dieser Abschnitt beschreibt das Auflager des vorderen Fußes. Die möglichen Verbindungen mit anderen bAm wird später in Varianten aufgezeigt.

Grundsätzlich wird zwischen zwei Formen von Lagern unterschieden: Wälzlager und Gleitlager. Im Prototyp werden provisorische Gleitlager verwendet. Zu Grunde liegt ein Aluminiumrohr das eine geringer Dichte hat, als die zu verbindenden Edelstahlprofile. Daher verschleißt erst das Aluminium-Röhrchen, bevor tragende Teile betroffen sind. Anstelle von Aluminium sollte jedoch idealerweise Grafit verwendet werden. Eine digitale Stoppuhr wird integriert werden, um die notwendige Erneuerung der Gleitlager vom bAm zu signalisieren. Derzeit ist diese Funktion nicht eingebaut, da keine Erfahrungswerte vorliegen.

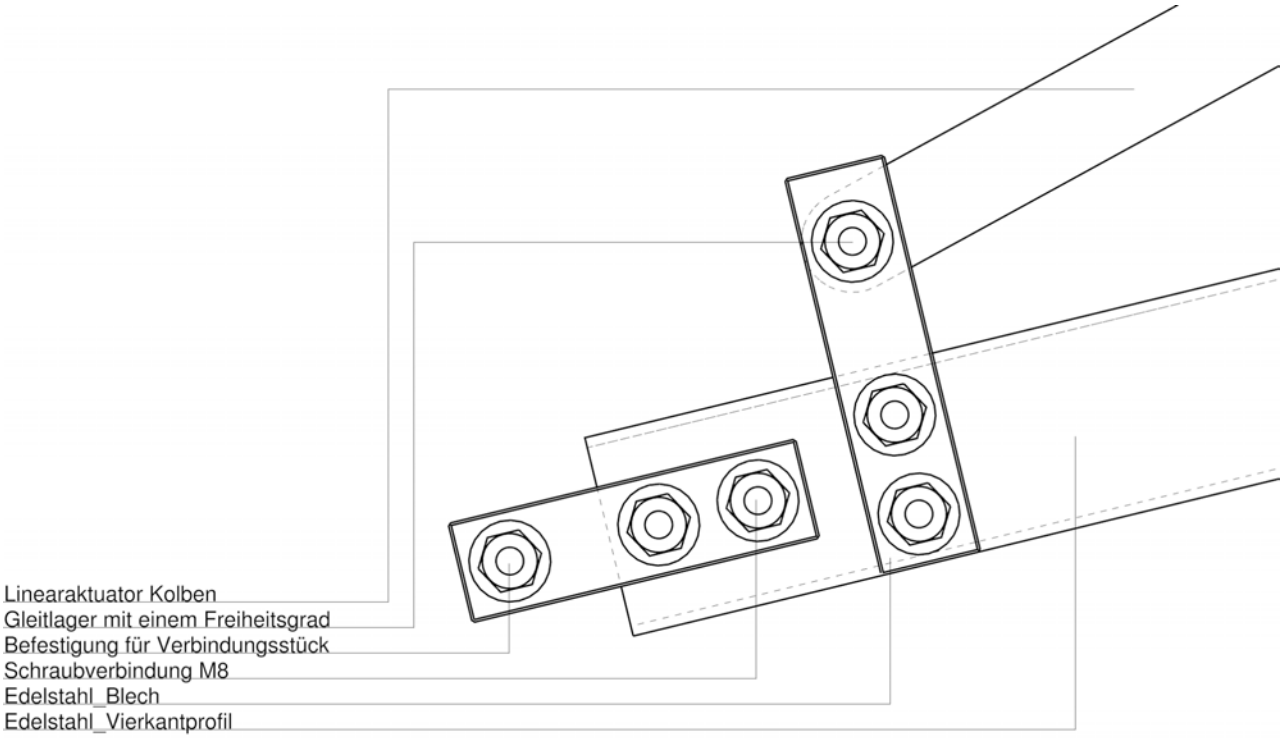


Zchng.3.5.1-3 Schemadetail Gleitlager

Schemadetail Metallbau
Fokus auf einen Fuß
in Grundriss und Ansicht



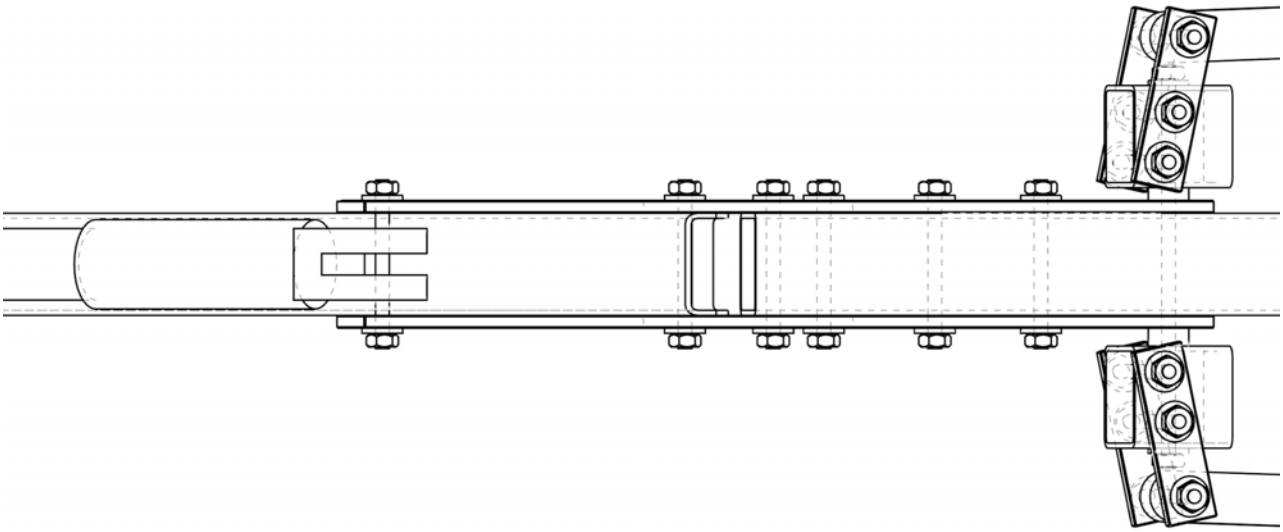
Zchng.3.5.1-4 Grundriss



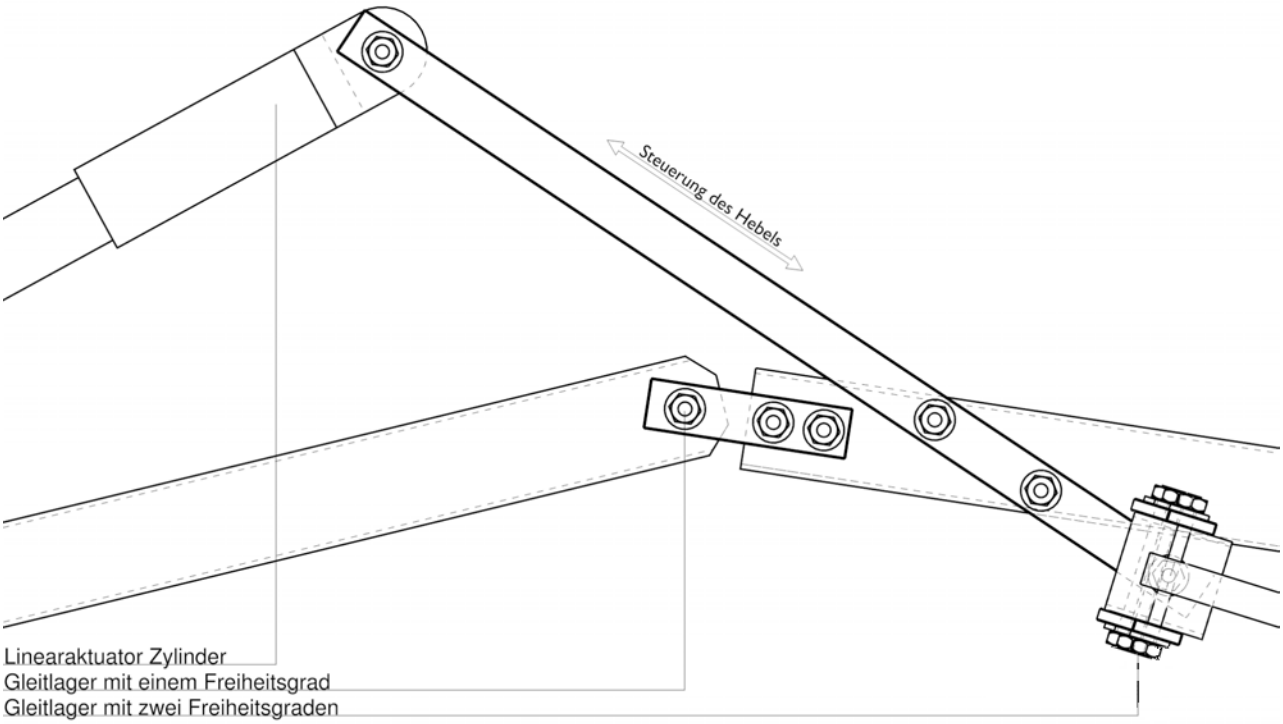
Zchng.3.5.1-5 Ansicht

Bei der Befestigung des Motors am Unterschenkel kann durch die Länge der Stahlbleche bestimmt werden, wie groß der Hebel des Schenkels ist. Werden längere Stahlbleche verbaut erhöht sich der Hebel und der Weg verkürzt sich. Werden kürzere Stahlbleche verbaut reduziert sich der Hebel und der Weg verlängert sich.

Schemadetail Metallbau
Fokus auf einen Fuß
in Grundriss und Ansicht



Zchng.3.5.1-6 Grundriss



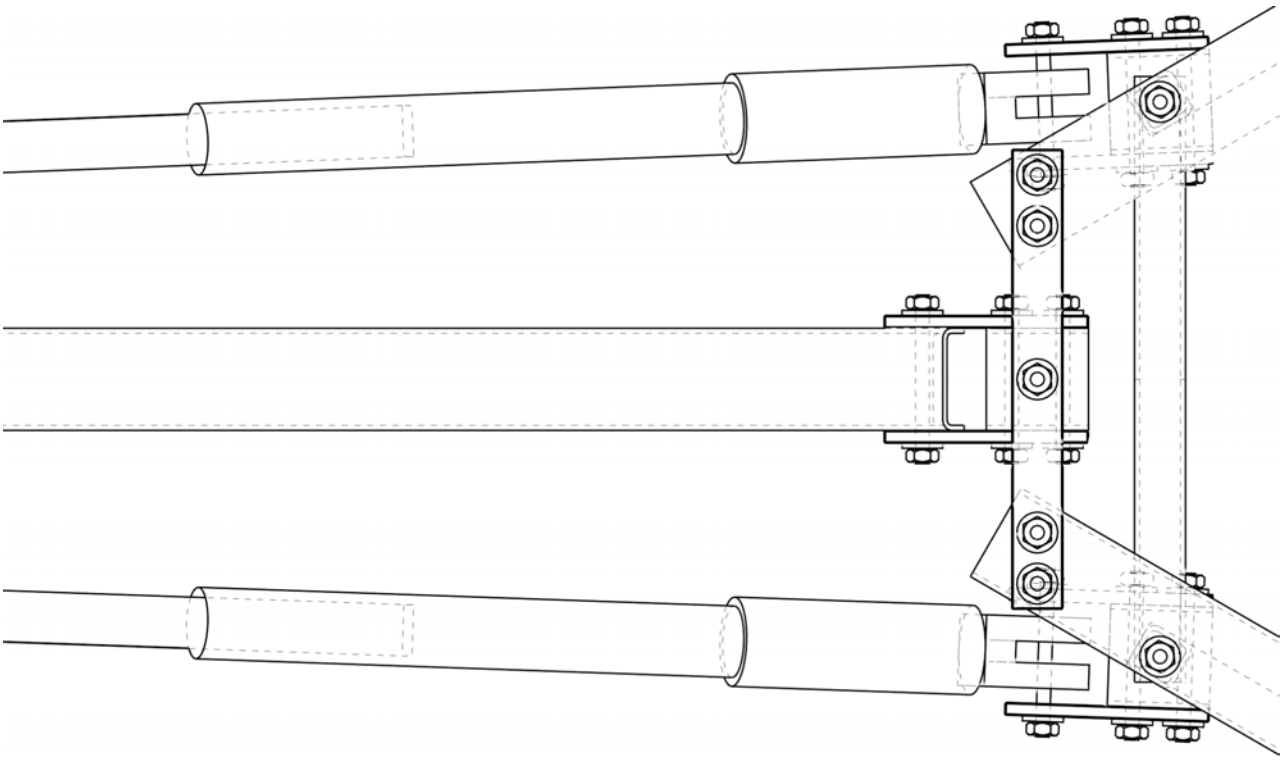
Zchng.3.5.1-7 Ansicht

Verweis auf:
Freiheitsgrade, Seite 124

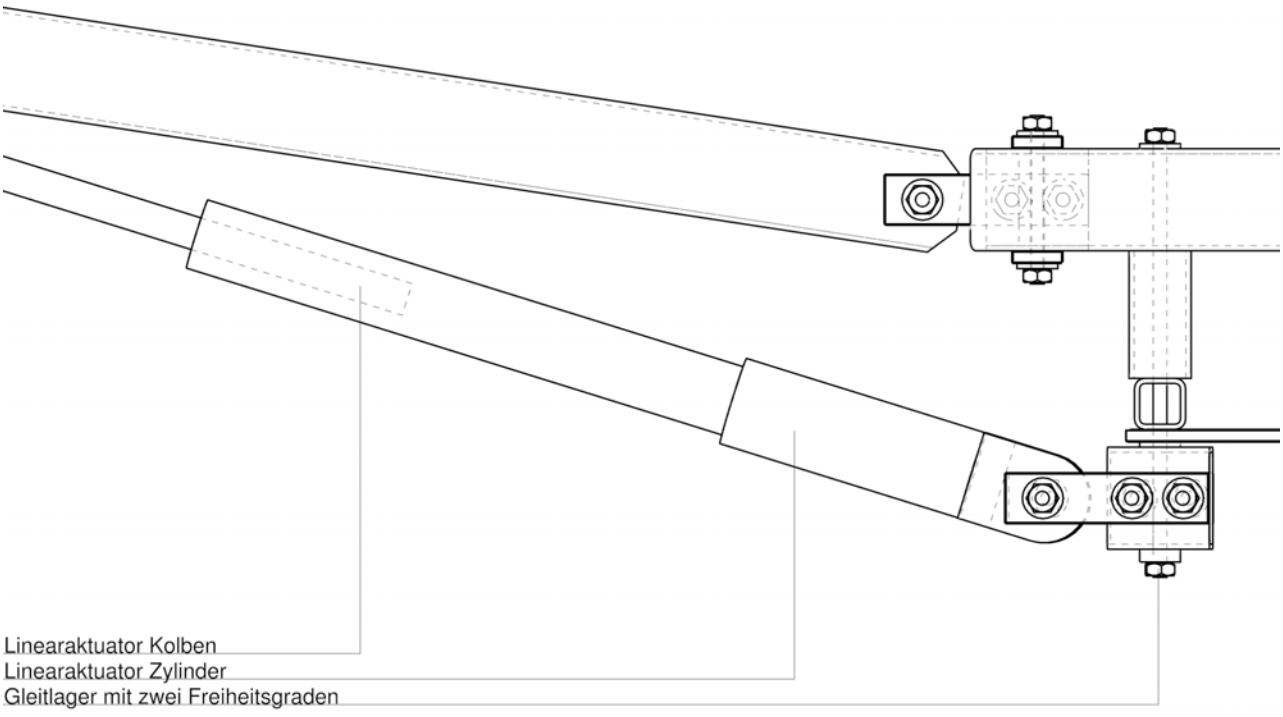
Hier zu sehen ist die Verbindung des Oberschenkels mit dem Kern und dessen Motoren. Sämtliche Verbindungen sind in zwei Freiheitsgrade zerlegt.

Die genaue Dimensionierung der Teile ist noch nicht abgeschlossen und muss noch erörtert werden. Im Rahmen der Arbeit wurden Formrohre QU 40 x 2 mm und Formrohr QU 20 x 2 mm, sowie Flachstangen 20 x 4 mm verwendet.

Schemadetail Metallbau
Fokus auf einen Fuß
in Grundriss und Ansicht



Zchng.3.5.1-8 Grundriss



Zchng.3.5.1-9 Ansicht

3.5.2 VERBINDUNGEN

Verweis auf:
Kettenlinie, Seite 184

So frei sich die Module auch bewegen, um eine Tragstruktur zu bilden, müssen sie sich schließlich doch kraftschlüssig verbinden können. Vorausgesetzt ist dabei, dass die Koppelung von den Modulen eigenständig durchgeführt werden kann. Die Verbindungen sind lediglich druckbelastet, wie im Bereich Reaktion bei Kettenlinie beschrieben. Das erleichtert die Konstruktion erheblich. Mechanismen zum Einrasten oder Einhaken, die möglicherweise aus Sicherheitsgründen erforderlich sind, sind im Rahmen der Arbeit noch nicht berücksichtigt. Diskutiert werden in diesem Abschnitt drei schematische Ansätze.

Die erste Variante besteht aus ineinander greifenden Gabeln. Dabei werden Stahlbleche seitlich an den Profilen montiert. Treffen die drei Füße mit exakt 120° aufeinander, liegt jeder einzelne Fuß auf zwei Punkten auf. Wird dieser Winkel variiert, liegen mindestens zwei der Füße lediglich auf einen Punkt auf. Das ist insofern problematisch, da die Füße dann nicht nur mit Normalkraft belastet sind, sondern auch Momente auftreten. Wäre ein Modul so konstruiert, dass die Verbindung starr sein kann, wäre dieser Lösungsansatz jedoch denkbar.

„Despite all my rage I am still just a rat in a cage“

The Smashing Pumpkins, „Bullet with butterfly wings“, 1995

Schemadetail Verbindung
Variante Gabel
Annäherung in drei Schritten

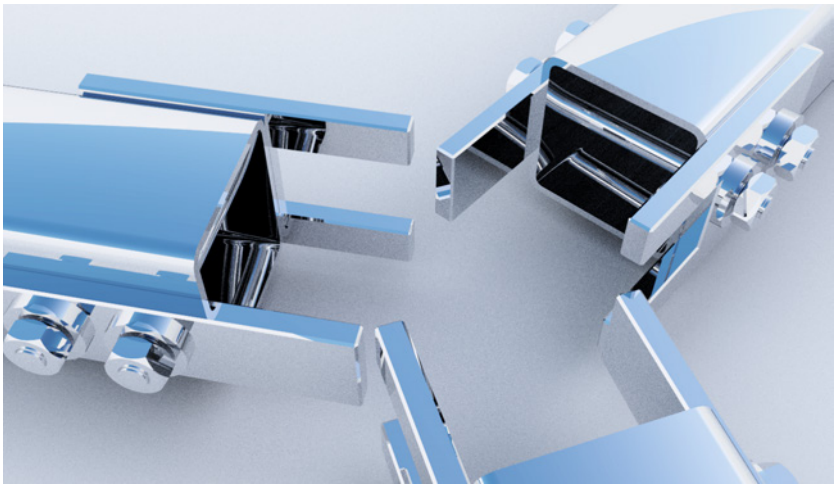


Abb.3.5.2-1 Entfernt

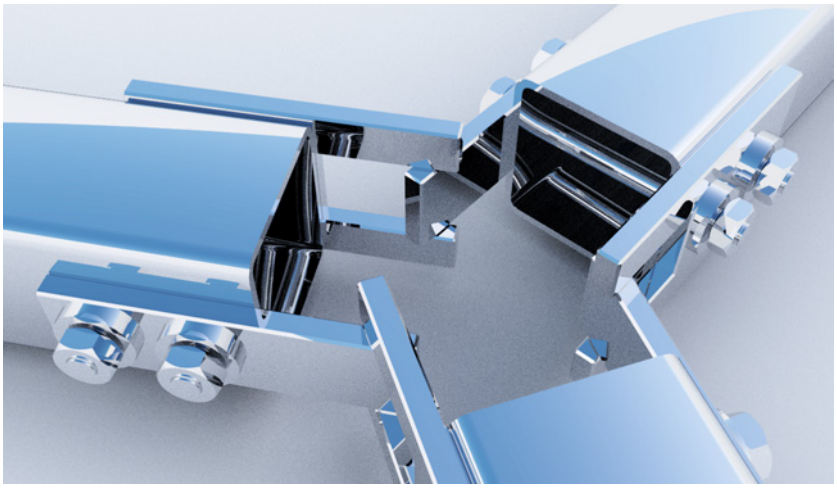


Abb.3.5.2-2 Annäherung

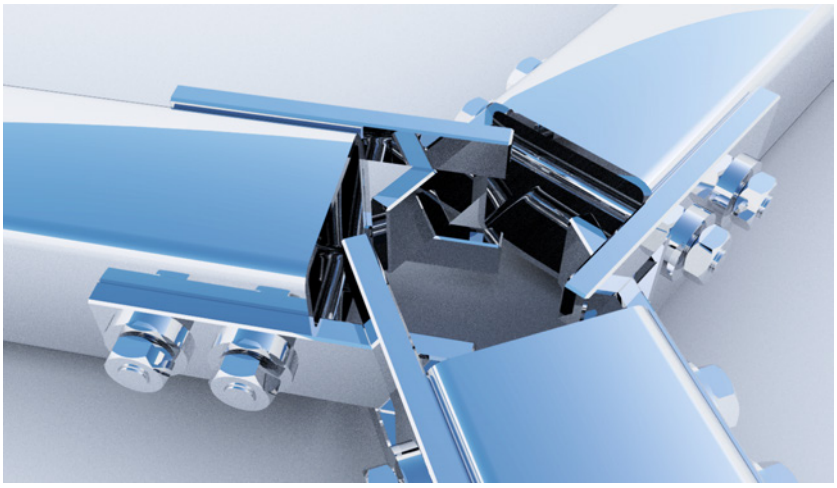


Abb.3.5.2-3 Verbunden

Die zweite Variante ist eine weiterer Versuch, die Verbindung mittels ineinander greifenden Gabeln zu lösen. Diese sind an den Enden gelenkig gelagert. Die Kräfte je Fuß konzentrieren sich auf jeweils einen Punkt. Momente entstehen lediglich durch den Versatz der Gabeln in Richtung Z.

Schemadetail Verbindung
weitere Variante mit Gabeln
Annäherung in drei Schritten

Abb.3.5.2-4 Entfernt

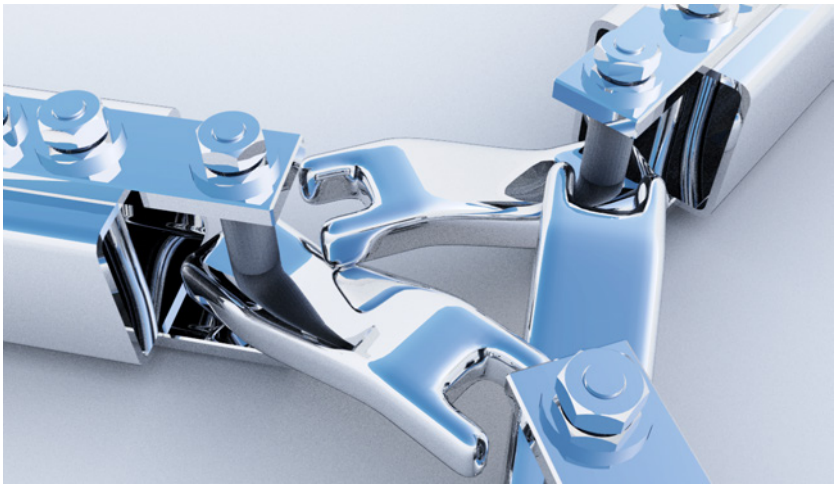


Abb.3.5.2-5 Annäherung

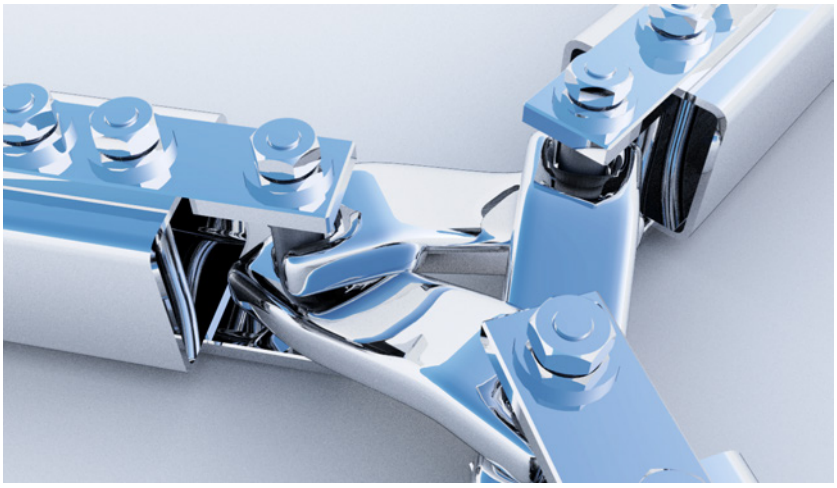


Abb.3.5.2-6 Verbunden

Die letzte Variante versucht, die Verbindung mittels Zahnradkugeln zu lösen. Vorausgesetz ist, dass die Module sich sehr präzise aneinander fügen können. Der Nachteil dieser Variante ist, dass ein Verdrehen der Gelenke Reibung innerhalb der Kugeln erzeugen würde.

Alles in Allem: ein Drahtseilakt. Die weitere Ausarbeitung der bAm legt den Fokus auf bereits vorhandene Technologien. Möglicherweise sind Anhängerkupplungen, wie in PKWs verbaut, eine zielführende Lösung.

Folgeseite:
Abb.3.5.2-7 Rendering

Schemadetail Verbindung
Variante mit Zahnradkugeln
Annäherung in drei Schritten

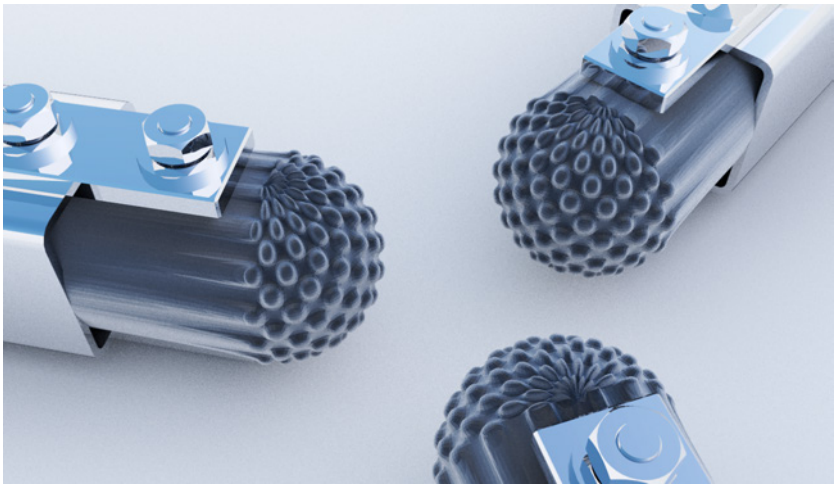


Abb.3.5.2-8 Entfernt

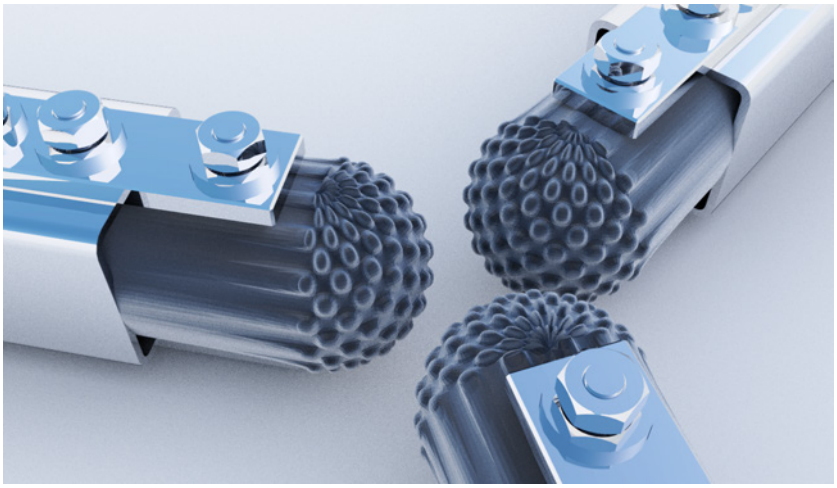


Abb.3.5.2-9 Annäherung

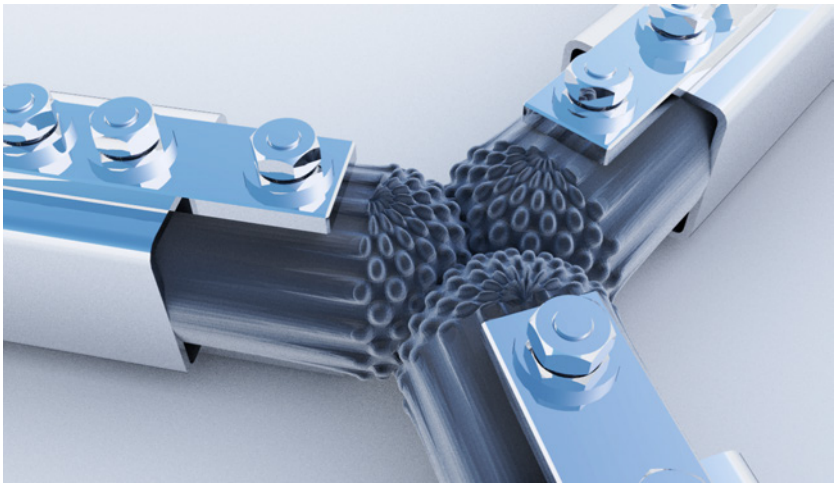
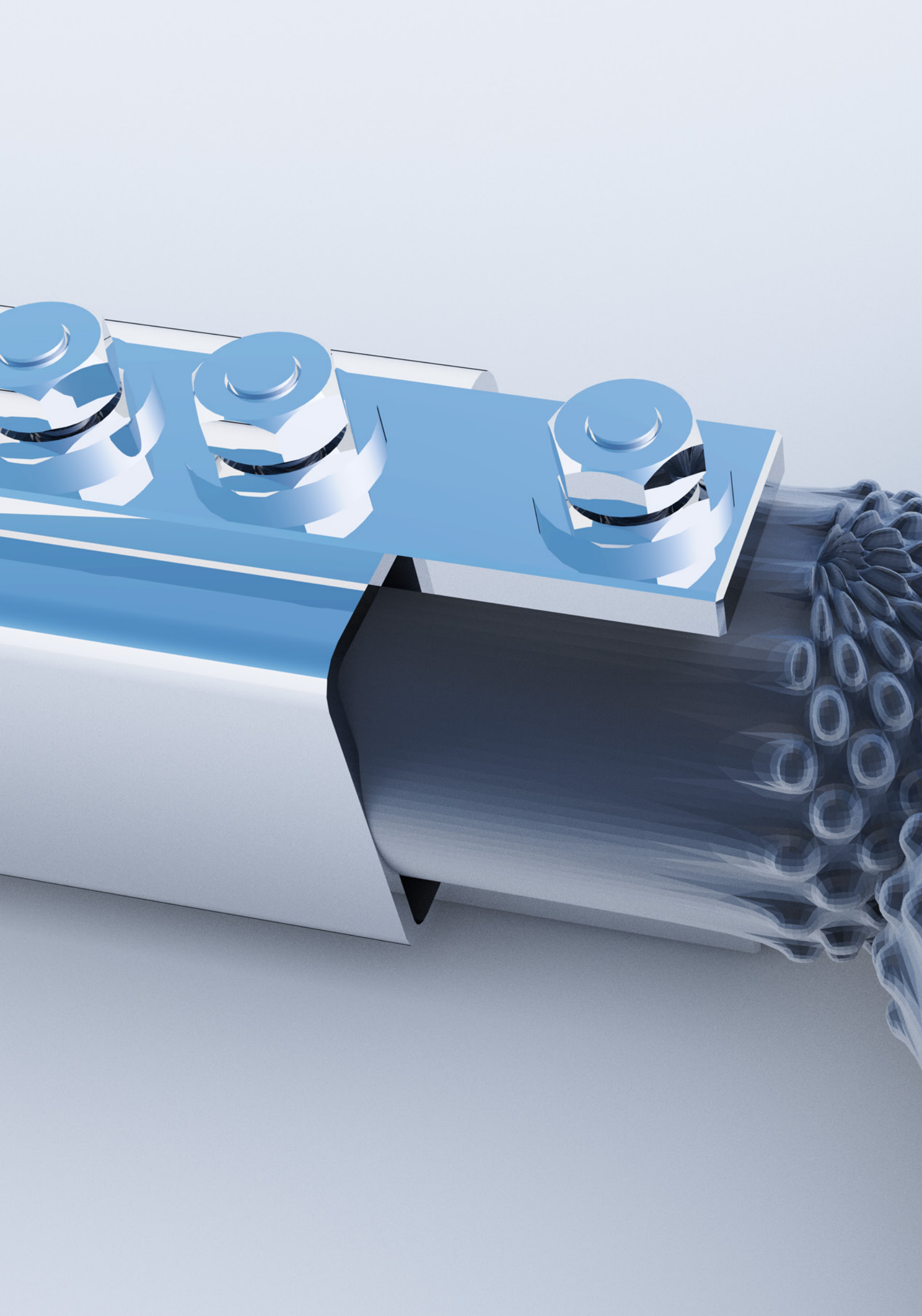
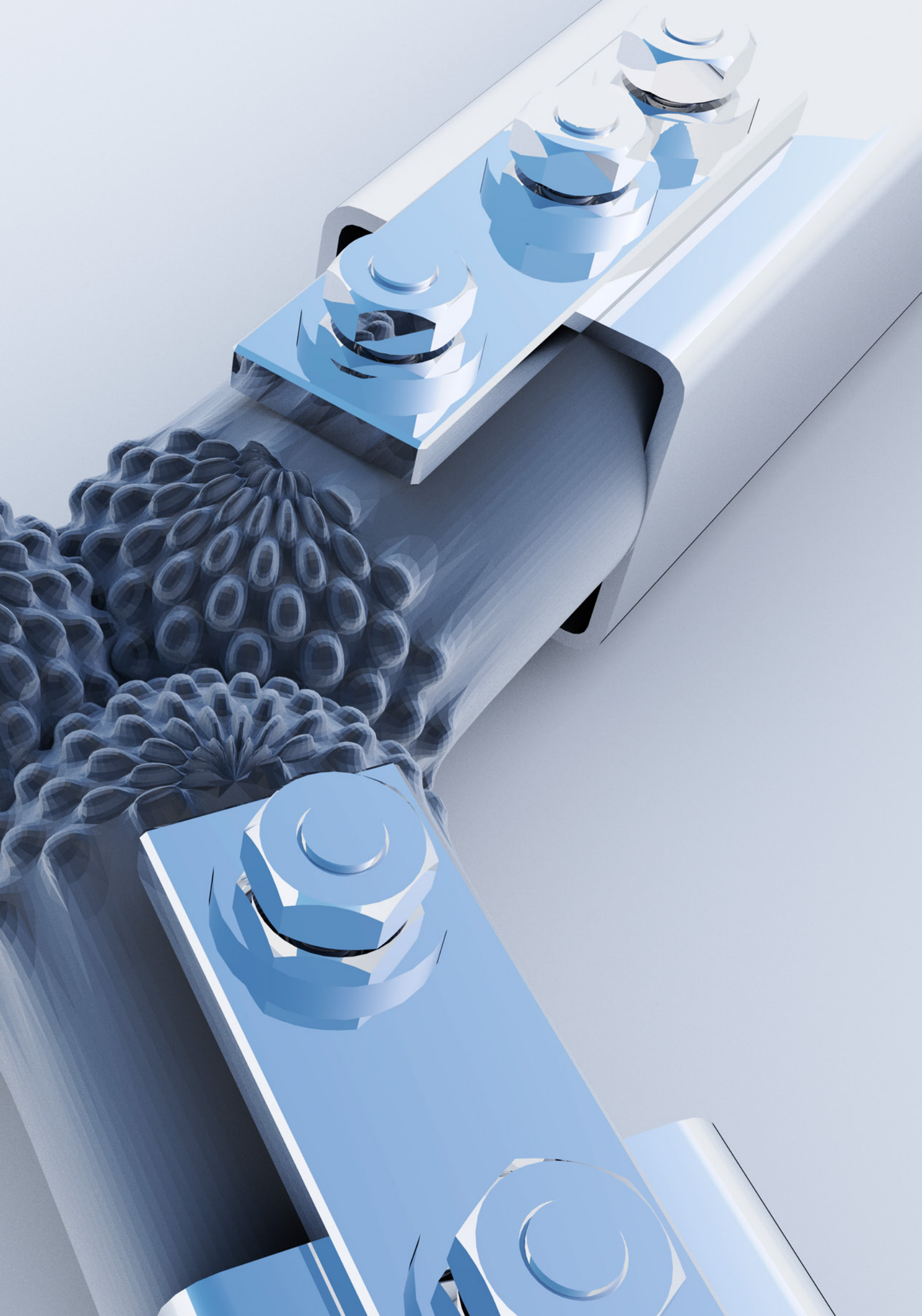


Abb.3.5.2-10 Verbunden





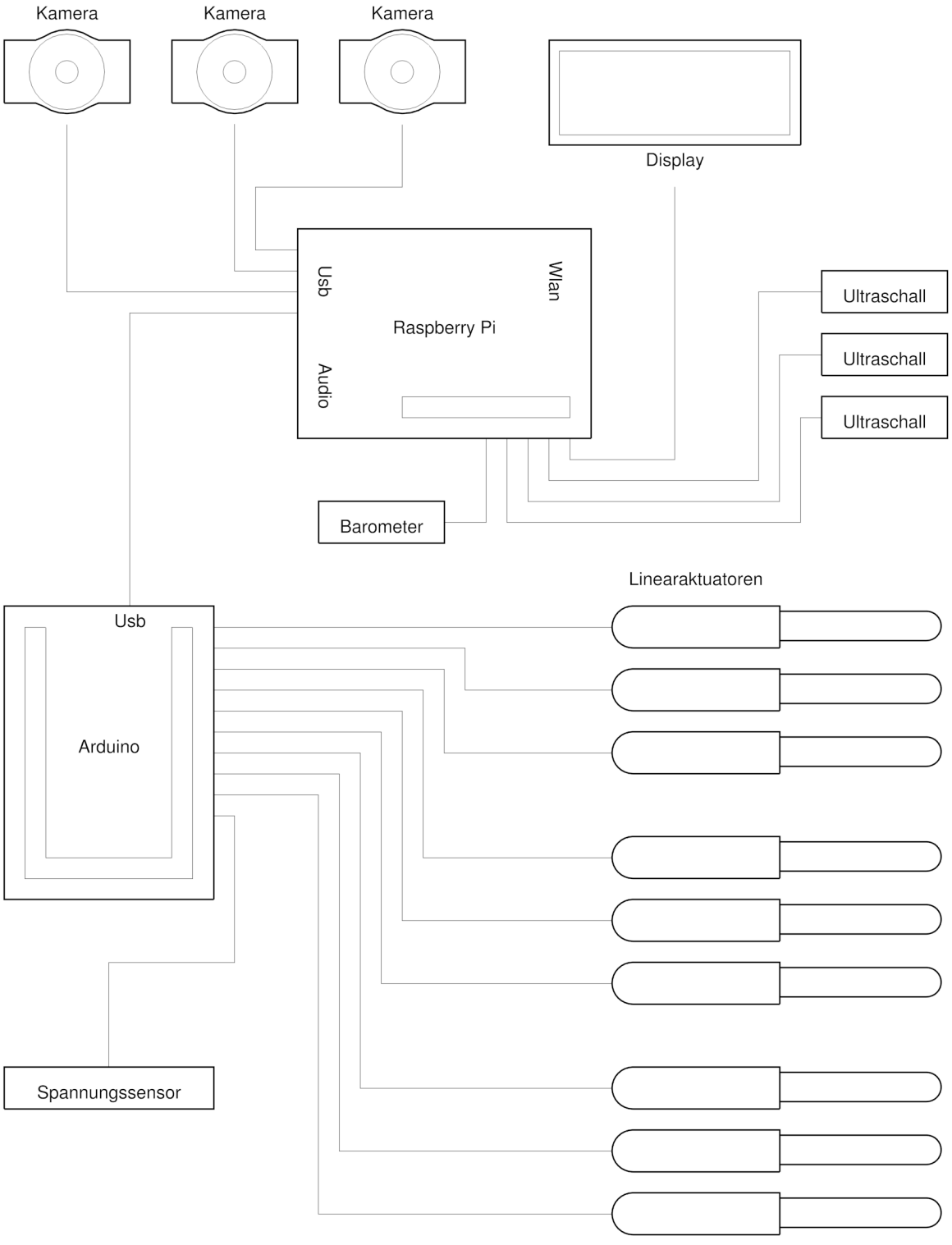
3.5.3 ELEKTRO

Kernstück eines bAm ist der Raspberry Pi. Dieser ist, metaphorisch geschrieben, das Gehirn jeder Einheit. Unmittelbar daran angeschlossen ist ein Display zur Ausgabe von einfachen Textzeilen. Drei Kameras sind über die entsprechenden USB-Schnittstellen verbunden und machen das visuelle Abtasten der Umgebung sowie die Spracheingabe möglich. Darüber hinaus bietet das Raspberry Pi die Möglichkeit, mittels Wlan und Ethernet, auf den bAm zuzugreifen.

Die Steuerung der Motoren erfolgt über ein Arduino Mega, das über USB an den Raspberry Pi angeschlossen ist. Zwar wäre es auch möglich, die Motoren über den Raspberry Pi direkt zu steuern, die Kombination dieser beiden Controller bietet jedoch mehrere Vorteile. Einerseits wird die Anzahl der Pins, sprich Anschlussmöglichkeiten, von 26 um 48 Stück erweitert. Zudem bietet das Raspberry Pi Pins mit 3V und das Arduino Pins mit 5V. So entsteht auch eine größere Bandbreite an Möglichkeiten, da sowohl Sensoren mit 3V als auch Sensoren mit 5V direkt verwendet werden können.

Die Stromversorgung des Prototypen basiert derzeit auf Netzstrom mit 240 V. Möglich wäre ein Speisen der bAm über zwei handelsübliche 12 V Autobatterien, die in Reihe geschaltet, 24 V liefern. Integrierte Solarzellen in den Pneus sind eine angedachte Erweiterung. So würden die Module auch autonom funktionieren können.

Schemadetail Elektro
diagrammatischer Aufbau
Schaltplan in abstrahierter Form



Diagr.3.5.3-1 Schaltplan

3.5.4 P N E U S

Technische Umsetzbarkeit, geringes Gewicht und niedrige Kosten machen Luftkissen zu einer attraktiven Möglichkeit die Hülle zu gestalten. Die folgenden Prototypen und Studien entstanden mit Unterstützung und in den Räumlichkeiten von experimonde | die Welt des Experiments. Dieser Verein setzt sich seit Jahrzehnten mit der Entwicklung von luftgeformten Konstruktionen im architektonischen Kontext auseinander und hat in diesem Feld eine enorme Expertise aufgebaut.

Zu sehen ist ein erster Versuch die bAm mit Hüllen auszustatten. Ansatz ist ein Kissen zwischen je zwei Füße zu spannen. Diese werden wiederum zu einem großen Kissen zusammengesetzt. Da die gewählte PVC-Folie überwiegend in Rollenbreiten von 135 cm verfügbar ist, muss der Zuschnitt aus mehreren Teilen bestehen. Gezeigt ist eine mögliche Zerlegung im Grundriss. Das Ausmaß des Kissens beträgt im Maßstab 1:5 in etwa 84 x 42 cm. Die gewählte Folie hat eine Stärke von 200 µm. Die Schweißnähte sind mit Schweiß-Stempeln von 5 mm Breite ausgeführt.

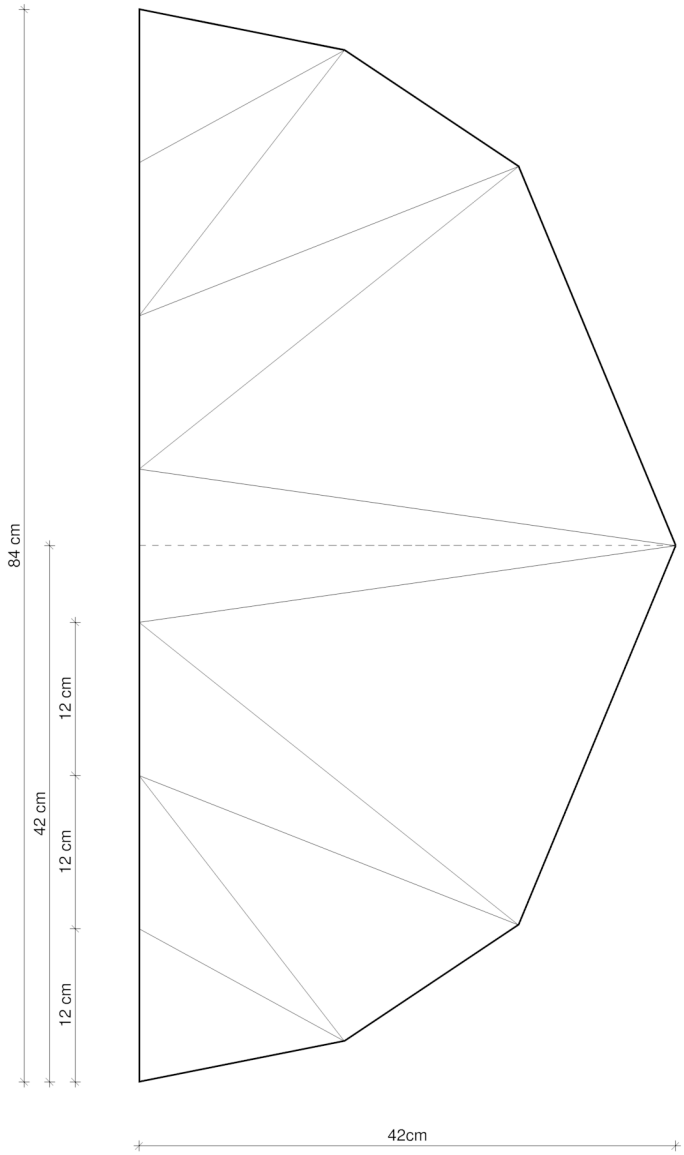
Info zur Angabe der Folienstärke:
1 µm = 1 Mikrometer = 0.001 mm

Das entstandene Kissen besteht aus mehreren Kammern. Diese sind durch Luftbrücken miteinander verbunden. Es gilt der Weg des geringsten Widerstandes. Die Luft verteilt sich von einer Kammer zur nächsten, bis der Druck in allen Kammern gleich ist. Bei einem Modell im Maßstab 1:1 könnten die Kammern mit separaten Luftzuführungen angesteuert werden. So wird das Auffinden von Löchern erleichtert.

Die Nähte in der Mitte versagen am schnellsten. Wie erwähnt ist der Druck in allen Kammern gleich, aber: je größer die Fläche, auf die der Druck wirkt, desto größer die Kraft, die auf die Schweißnaht und die Folie wirkt. Die Schweißnaht ist weniger tragfähig als die Folie. Da die Schweißnaht an allen Bereichen identisch ausgeführt ist, bilden die Schweißnähte somit in den größeren Kammern grundsätzlich die Schwachstelle. Andererseits sind gekrümmte Schweißnähte an der konvexen Seite der Krümmung umso weniger belastbar, je geringer der Krümmungsradius ist.

Abb.3.5.4-1 Foto

Schemadetail Pneus
erster Prototyp
Entwicklung der Kissenform



Zchng.3.5.4-1 Grundriss

Die Luft in einem Pneu folgt stets dem Weg des geringsten Widerstandes. Folglich nimmt ein Pneu stets die Form einer Kugel an, wenn er aufgeblasen wird. Wie aber kann anstelle einer Kugel eine flachere Geometrie erzeugt werden? Dieser digitale Zugang zeigt einen möglichen Lösungsansatz. Ähnlich wie bei einer Luftmatratze werden Stege eingeführt, die das ermöglichen sollen. Positiver architektonischer Nebeneffekt: Es entsteht eine Rautenheftung, die fast wie ein antiquiertes Biedermeier-Möbel anmutet.

Schemadetail Pneus
digitale Studie
Entwicklung der Kissenform

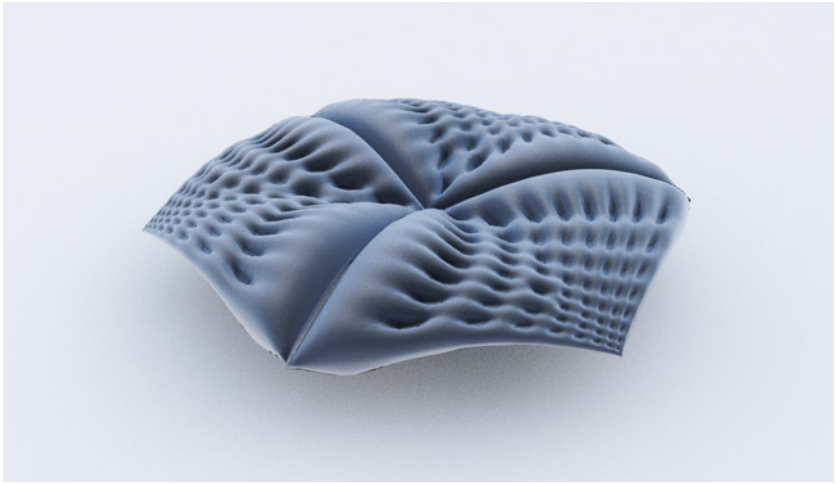


Abb.3.5.4-2 Einzeln

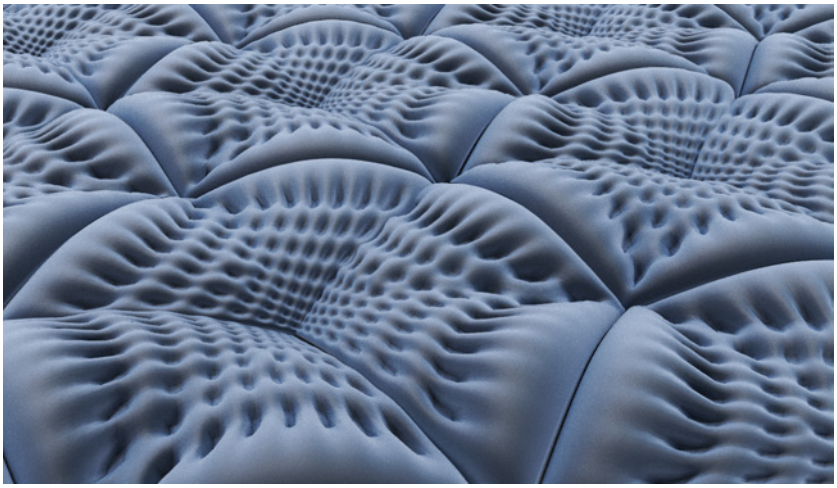


Abb.3.5.4-3 Kombination

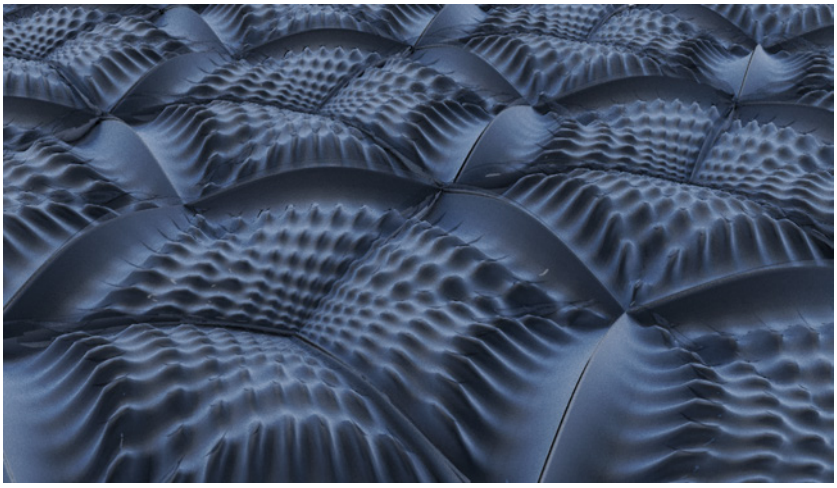
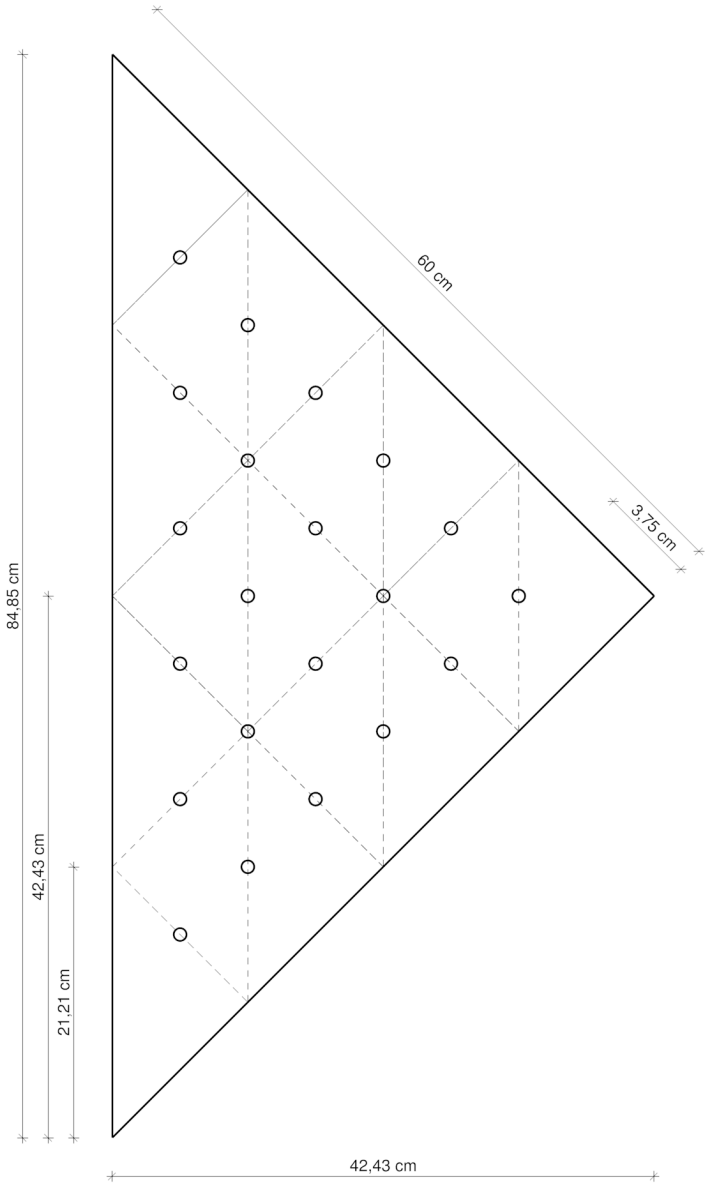


Abb.3.5.4-4 Umkehr

Dieser Prototyp zeigt eine Umsetzung der vorausgegangen digitalen Simulation. Zwei Folien werden übereinander gelegt und durch gezielt gesetzte Schweißnähte miteinander verbunden. Um möglichst wenig Widerstand zu bieten und so das Einreißen zu vermeiden, sind die Schweißnähte der Verbindungen kreisrund ausgeführt. Ihr Durchmesser beträgt 20 mm. Die Schweißnähte am Rand sind wie beim vorausgegangenen Prototyp mit 5 mm Stärke ausgeführt.

Abb.3.5.4-5 Foto

Schemadetail Pneus
analoge Studie
Entwicklung eines flachen Kissens



Zchng.3.5.4-2 Grundriss

Bewegen sich die Füße zueinander, muss der Pneu dieser Bewegung folgen können. Zielsetzung ist daher, dass der Pneu entsprechend überdimensioniert und in eine Richtung vorgewölbt wird. Eine Lösung muss gefunden werden, bei der die Wölbung immer nach oben ausweicht, damit das Kissen nicht am Boden schleift. Erster Versuch ist das Verwenden von zwei unterschiedlichen Folienstärken. Die untere mißt dabei 500 µm, die obere 200 µm. Durch den höheren Widerstand der unteren Folie sollte sich der Pneu zwangsläufig nach oben durchwölben. Dem ist nicht so. Der Druck im Inneren des Pneus ist zu gering. Die kreisrunden Schweißnähte stellen dabei die Schwachpunkte dar. Auf den Folgeseiten wird ein Lösungsansatz diskutiert.

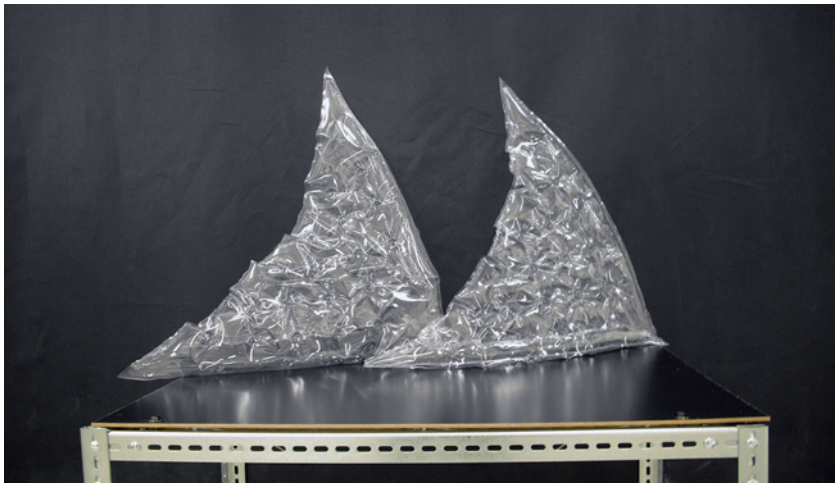
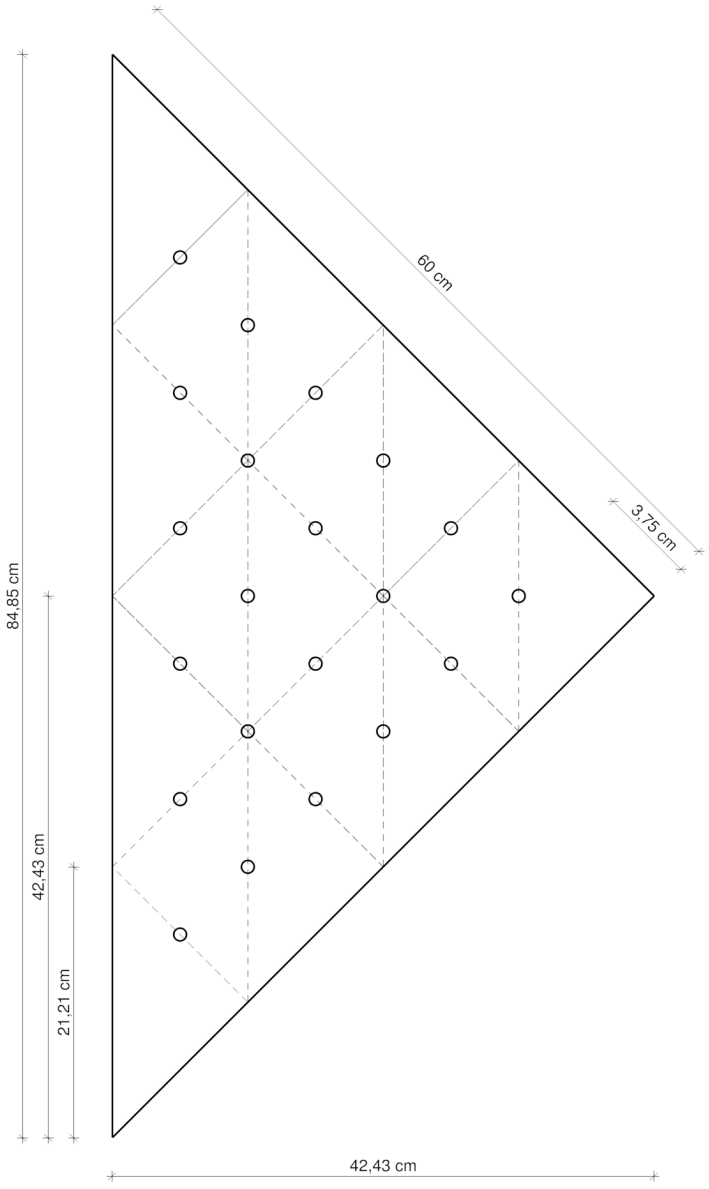


Abb.3.5.4-6 Foto

Schemadetail Pneus
analoge Studie
Entwicklung eines gewölbten Kissens



Zchng.3.5.4-3 Grundriss

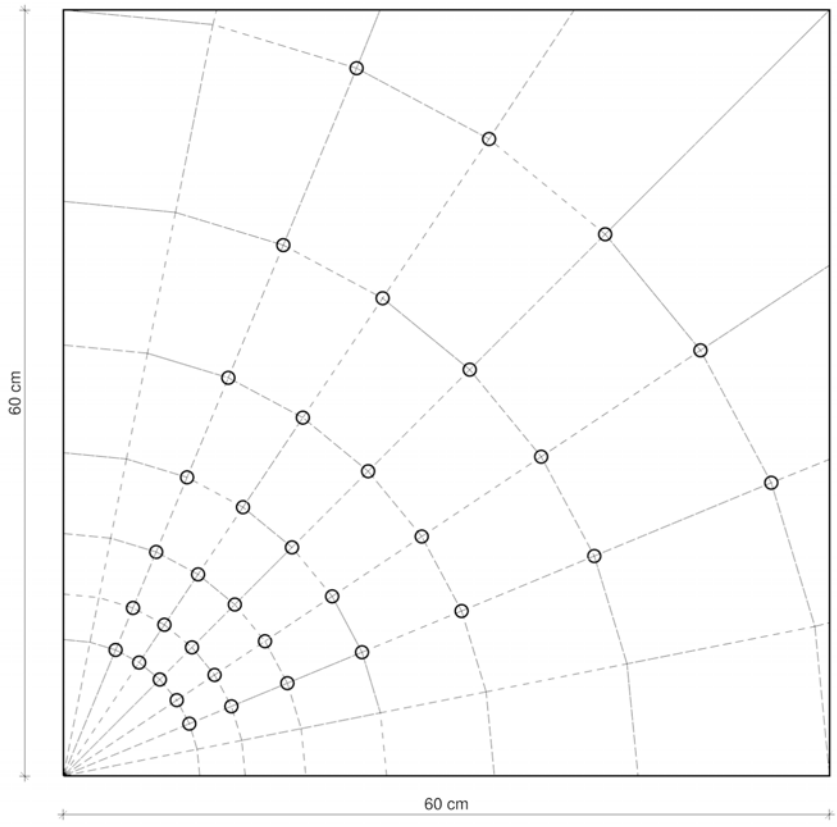
Um ein Einreißen der abspannenden Schweißnähte zu verhindern, werden diese mittels Schrauben und Beilagscheiben verstärkt. Eine ausführlichere Diskussion des Themas ist im Abschnitt Schweißnaht zu finden. Das Foto zeigt einen ersten Prototyp mit diesem Konzept. Dieses Kissen lässt sich mit deutlich höherem Druck befüllen. Daraus resultiert eine weitaus größere Verformung, die das Kissen beim Aufblasen annimmt. Eine gezielte Verkleinerung der Abstände von Schweißnähten zueinander führt zu einer variierbaren Größenänderung der Bereiche. Stellen mit geringem Abstand der Schweißnähte zueinander dehnen sich weniger aus. Stellen mit größerem Abstand dehnen sich stärker aus. Es entsteht eine Wölbung des Kissens. Ohne Zutun erfolgt diese Wölbung des Kissen, entweder nach oben oder nach unten. Beim Aufblasen kann jedoch manuell die Richtung vorgegeben werden.

Verweis auf:
Naht, Seite 236

Folgeseite:
Abb.3.5.4-7 Foto

Abb.3.5.4-8 Foto

Schemadetail Pneus
analoge Studie
Entwicklung eines gewölbten Kissens

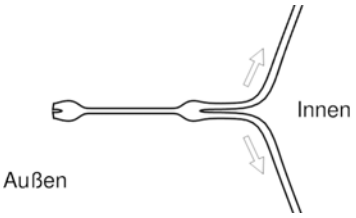


Zchng.3.5.4-4 Grundriss

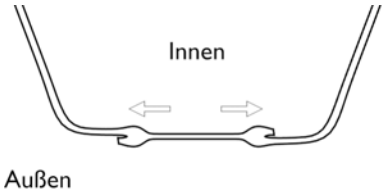
Dieser Abschnitt beschäftigt sich speziell mit den Schweißnähten der Kissen. Unten zu sehen sind die zwei grundlegenden Anordnungen der Folien beim Schweißen. Bei Variante 1 entsteht ein schmaler überstehender Streifen in Breite der Schweißnaht. Dieser kann verwendet werden, um den Pneu einzuklemmen und so zu befestigen. Variante 2 ist aufgrund der Krafteinwirkung in Richtung der Folie, ohne zusätzliche Verstärkung, etwas stabiler.

Die Zeichnung rechts oben zeigt eine mögliche Ausführung der Schweißnähte zur Abspannung der Kissen. Ausgeführt wird die Schweißnaht dabei mit 32 mm Durchmesser. Ein Loch, das mittig in der Schweißnaht sitzt, erlaubt das Durchführen einer Schraube, die mit zwei Beilagscheiben mit 36 mm die beiden Folien kraftschlüssig zusammen drückt. Die Schweißnaht dient in diesem Detail lediglich der Abdichtung der beiden Folien und hat keine weitere tragende Wirkung. Zum Schutz der dünnen Folie der Kissen wird eine mehr als doppelt so dicke Folie zwischen Beilagscheibe und Kissen gelegt.

Analog zur Schweißnaht der Abspannungen wird der Randbereich ausgeführt. An Stelle der Beilagscheiben treten Aluprofile, die über die gesamte Länge des Kissens verlaufen.

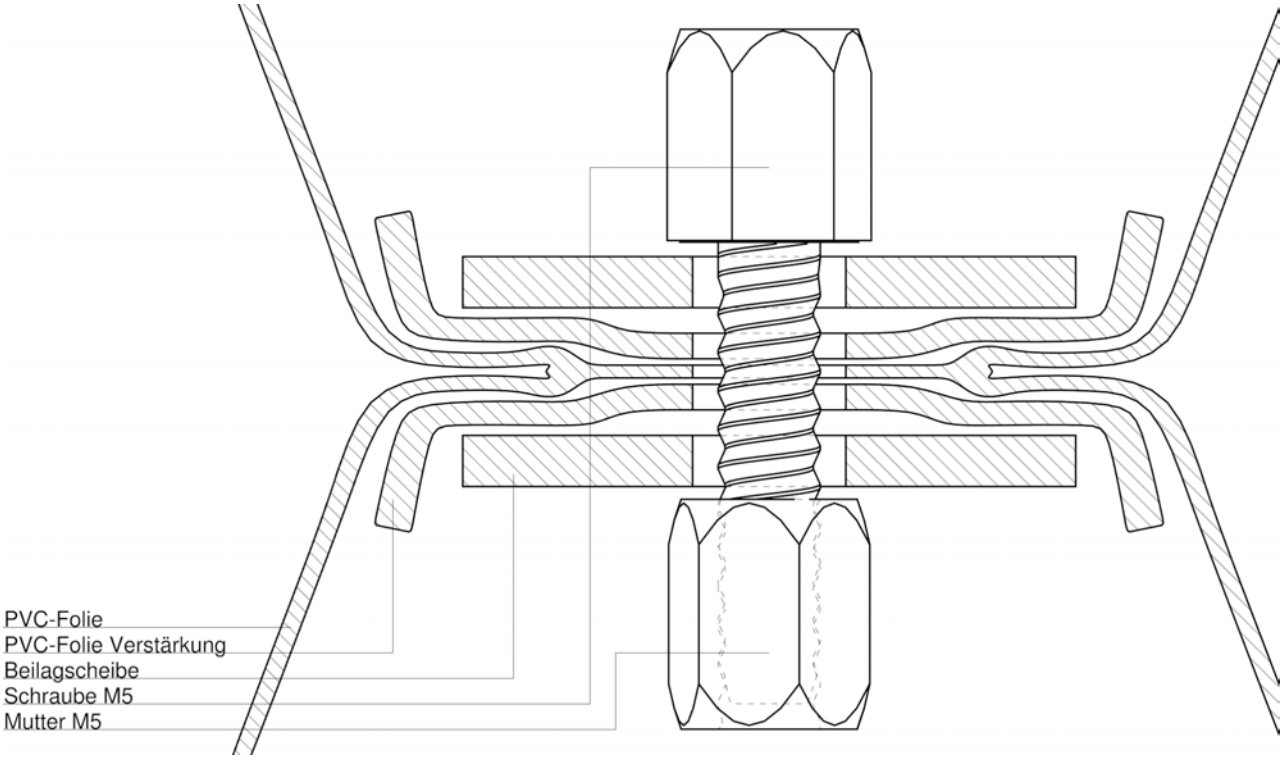


Zchng.3.5.4-5 Variante 1

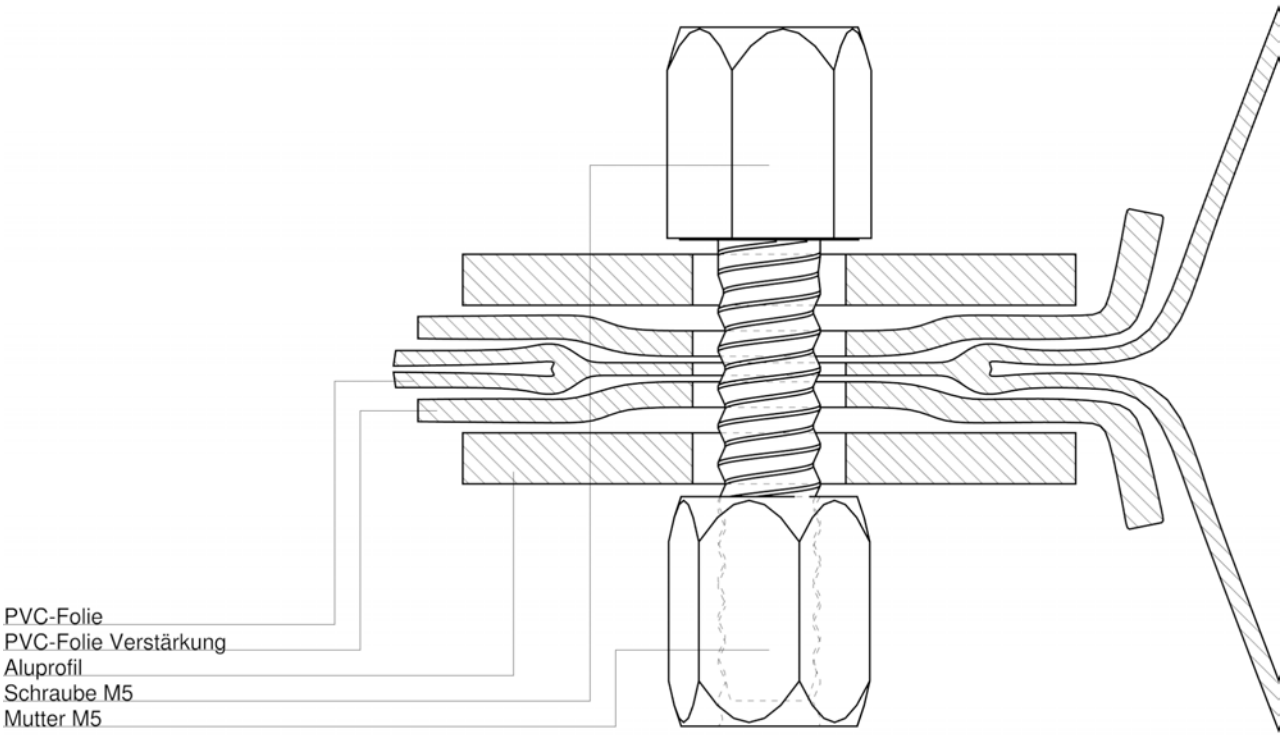


Zchng.3.5.4-6 Variante 2

Schemadetail Pneus
Fokus auf Schweißnähte
Entwicklung eines gewölbten Kissens



Zchng.3.5.4-7 Abspannung symbolisch gestreckt dargestellt



Zchng.3.5.4-8 Randbereich symbolisch gestreckt dargestellt

Aus den Erkenntnissen der vorausgegangenen Studien wurde schließlich ein Kissen für den Prototyp im Maßstab 1:1 entwickelt. Die Abwicklung zur Rechten zeigt eines von insgesamt drei Kissen pro bAm. Zu sehen ist die Position des Ventils, der Auflager, der Verbindung der Einzelstücke und den Abspannungen. Das digitale Modell links unten zeigt eine Simulation des Kissens im aufgeblasenen Zustand.

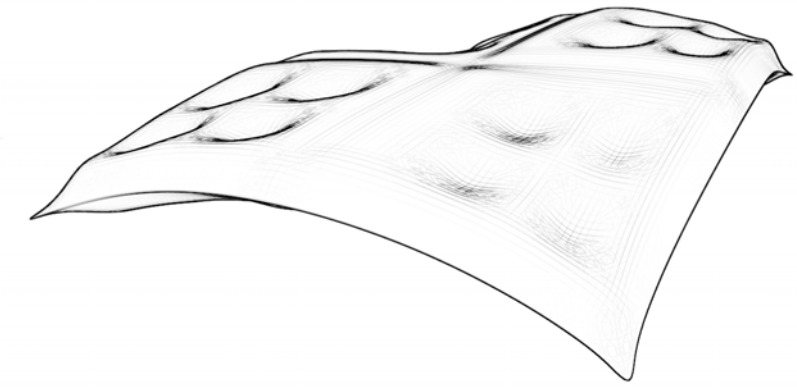
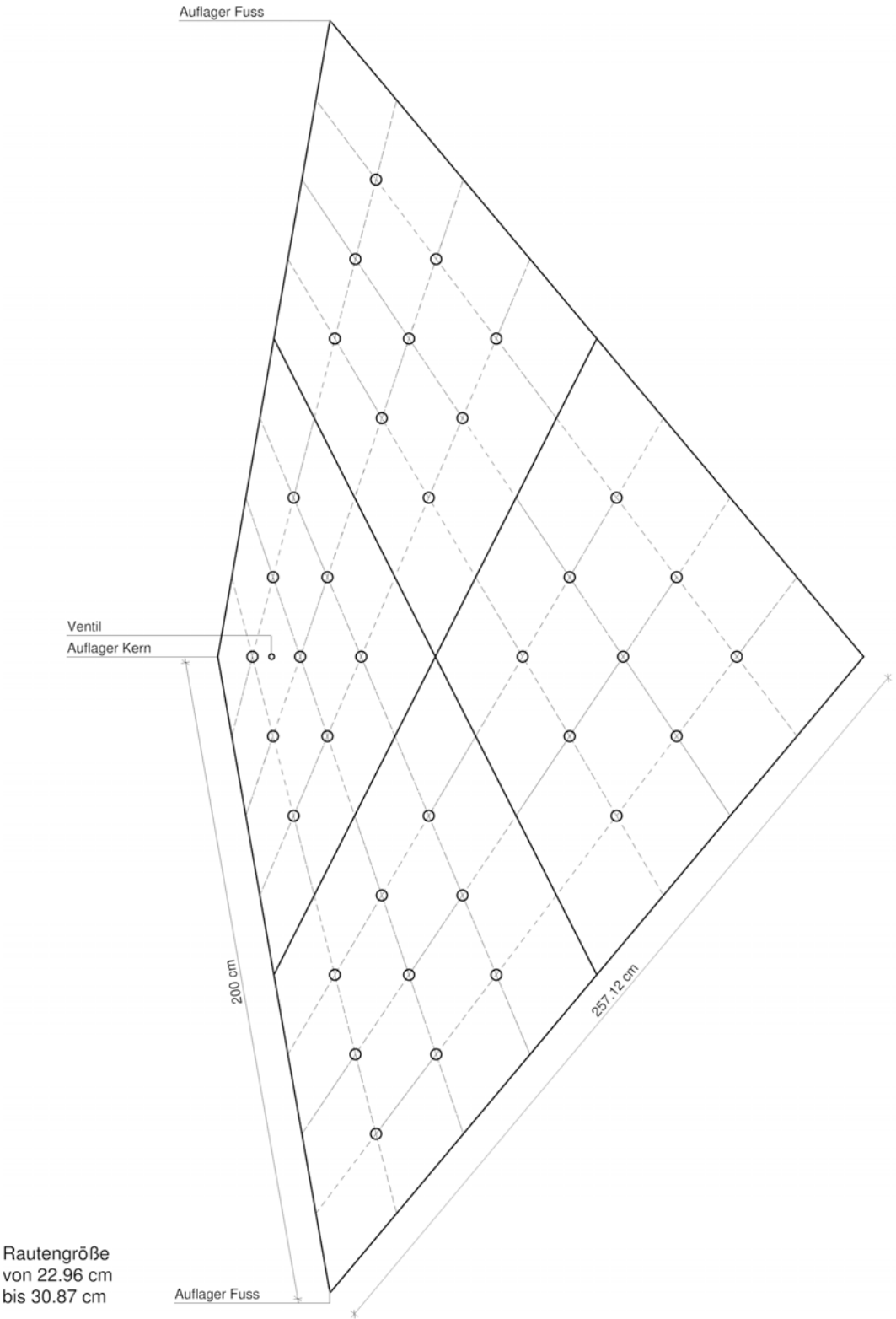


Abb.3.5.4-9 Simulation

Schemadetail Pneus
Ausführung im Prototyp
Zusammenfassung der vorausgegangenen Erkenntnisse



Zchng.3.5.4-9 Grundriss

Um die möglichen Verformungen des Pneus zu testen, wurde ein digitales Modell erstellt. Zu sehen ist die maximale Bewegung eines Fußes von mehreren Standpunkten aus. Die Folgeseite zeigt ein Rendering des Kissens im Ruhezustand eines bAm.

Folgeseite:
Abb.3.5.4-10 Rendering

Schemadetail Pneus
Ausführung im Prototyp
Simulation der Bewegungen

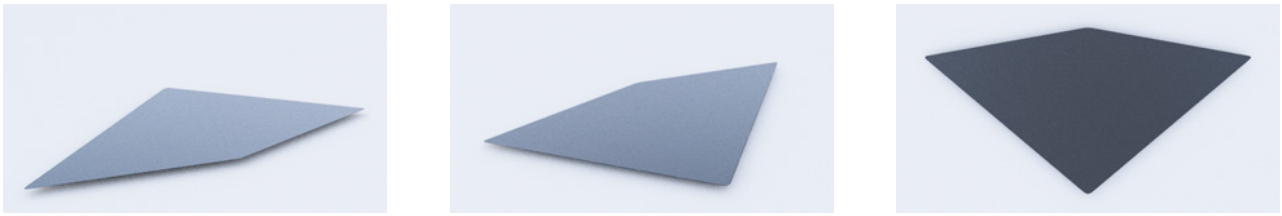


Abb.3.5.4-11 a,b,c Ausgangsposition

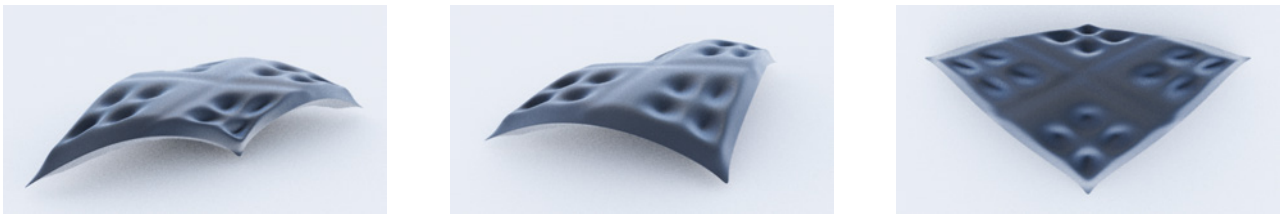


Abb.3.5.4-12 a,b,c 20 %

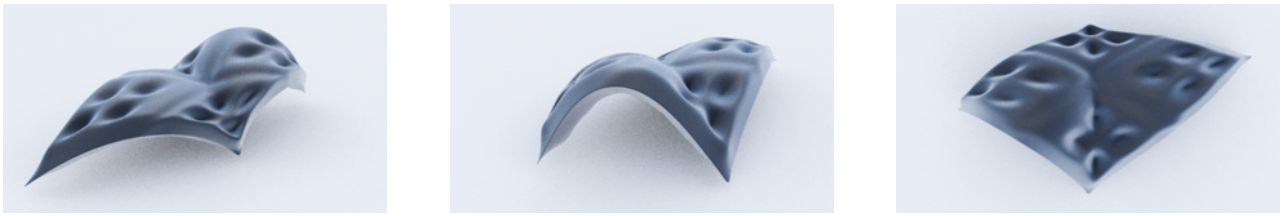


Abb.3.5.4-13 a,b,c 40 %

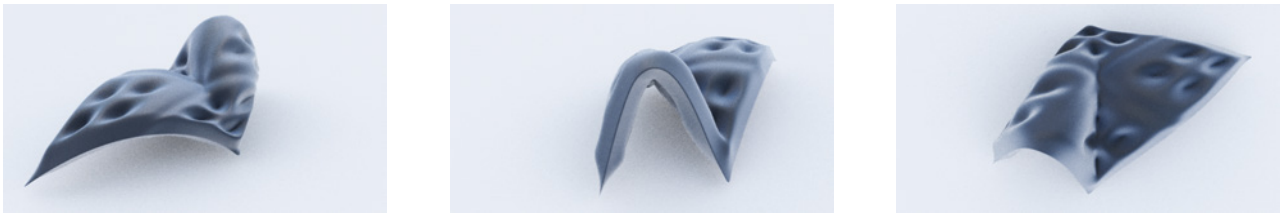


Abb.3.5.4-14 a,b,c 60 %

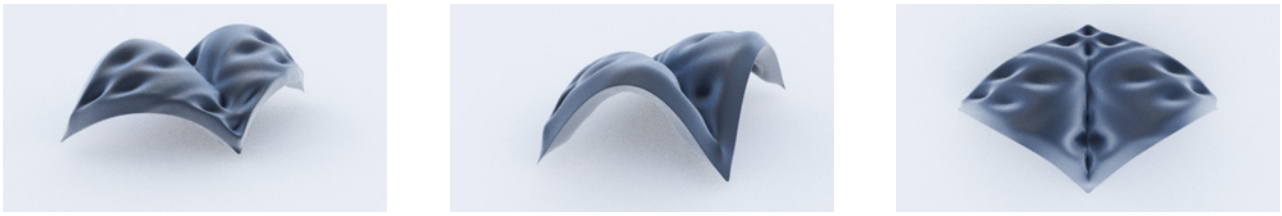


Abb.3.5.4-15 a,b,c 80 %

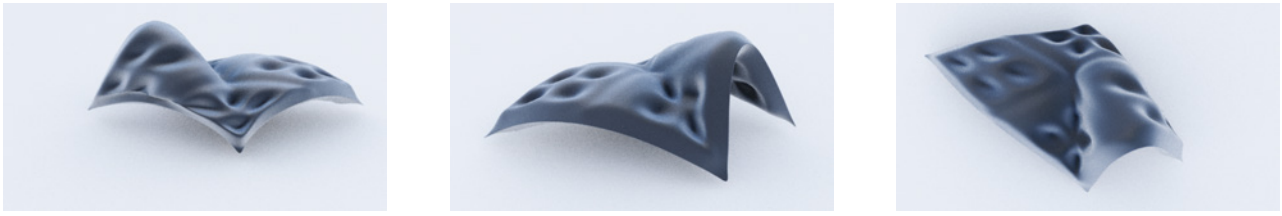
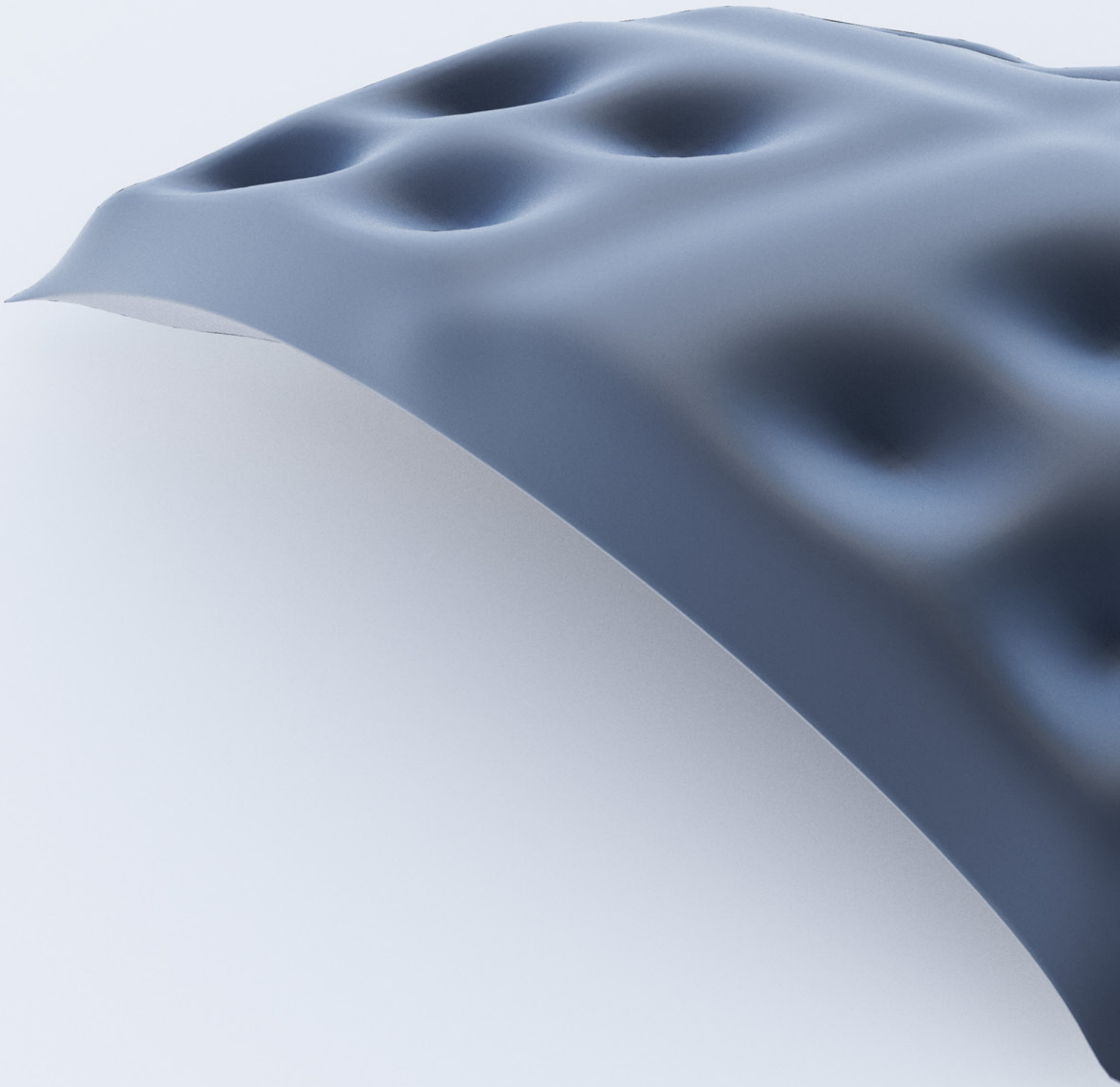
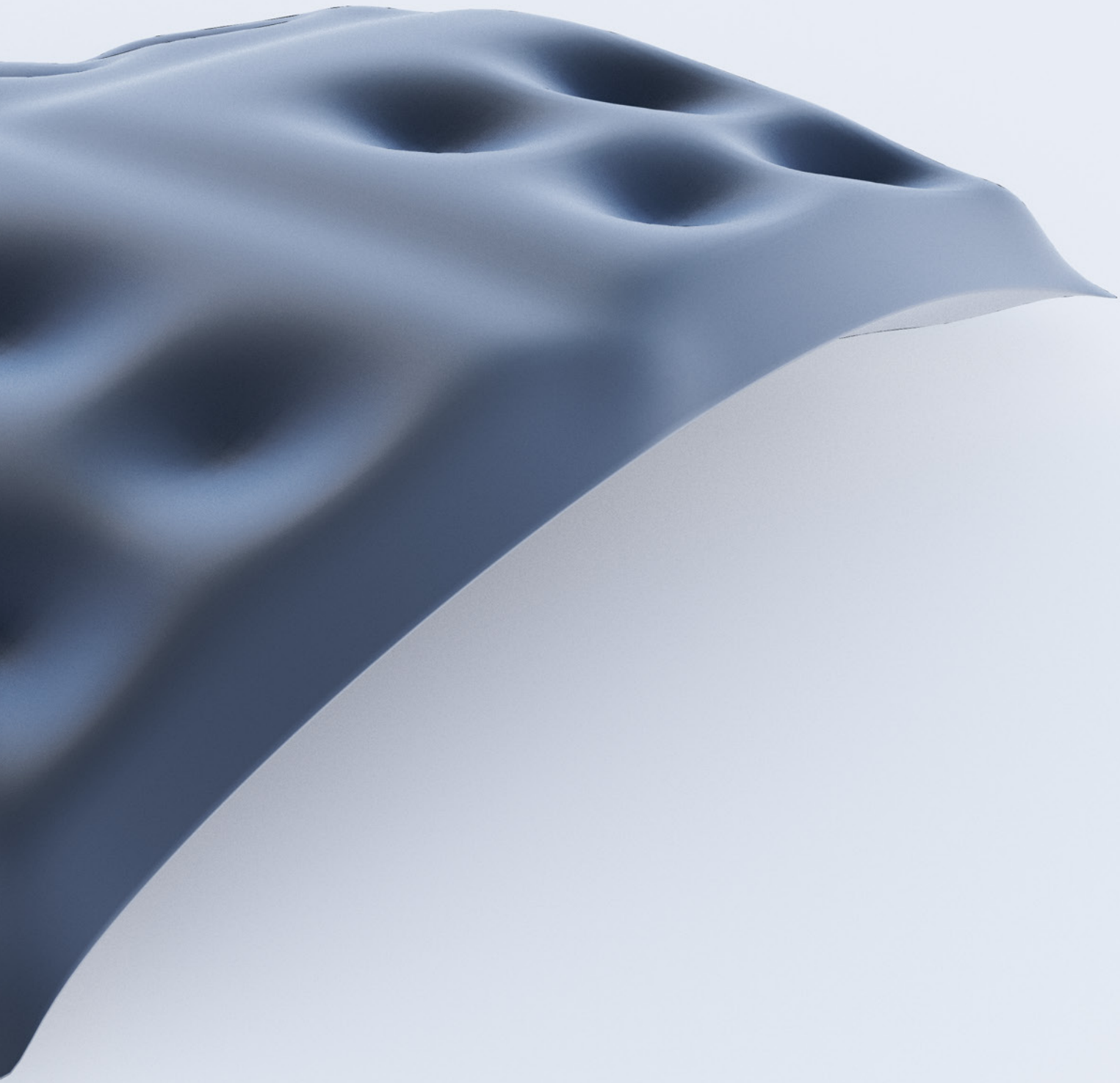


Abb.3.5.4-16 a,b,c Endposition





3.6 UMSETZUNG

3.6.1 EINZELTEILE

„... denn die Wahrheit einer Absicht
ist nur die Tat selbst“

Georg Wilhelm Friedrich Hegel,
„die Phänomenologie des Geistes“,
1807

Folgeseite:
Abb.3.6.1-1 Trennen mittels Winkelschleifer

Umsetzung Metallbau
Aufzeigen der Arbeitsschritte
Bohren, Schleifen, Schneiden

Abb.3.6.1-2 Bohren der Metallteile

Abb.3.6.1-3 Schleifen mittels Schleifbock

Abb.3.6.1-4 Einspannen der Teile zum Schneiden


EXPERTS
PROFESSIONAL

Folgeseite:
Abb.3.6.2-1 Aufgereihte Einzelteile

Umsetzung Einzelteile
Aufzeigen der Arbeitsschritte
Sortieren und montieren

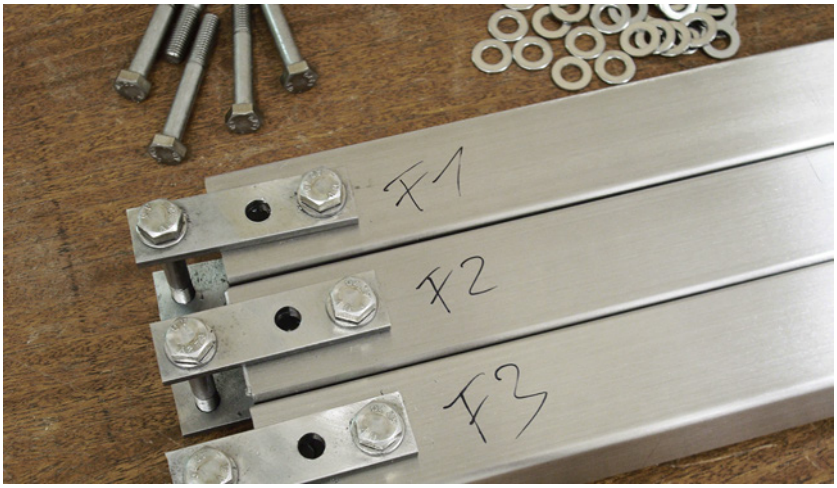


Abb.3.6.2-2 Vorbereitung der Gelenke

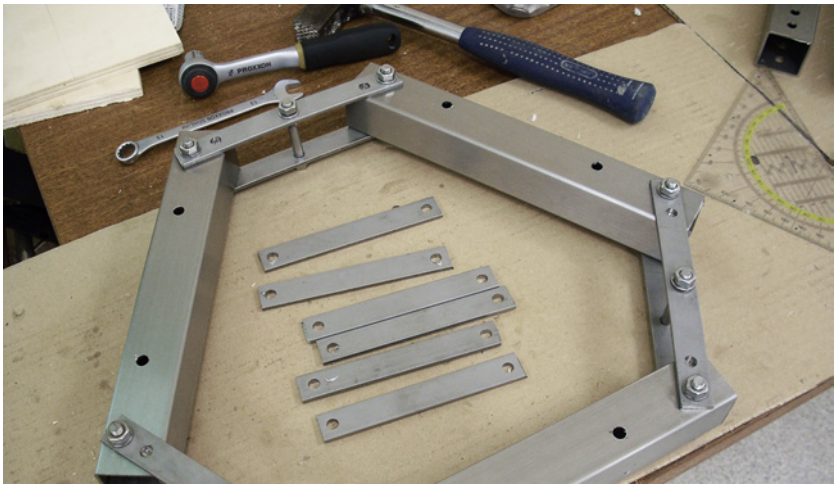
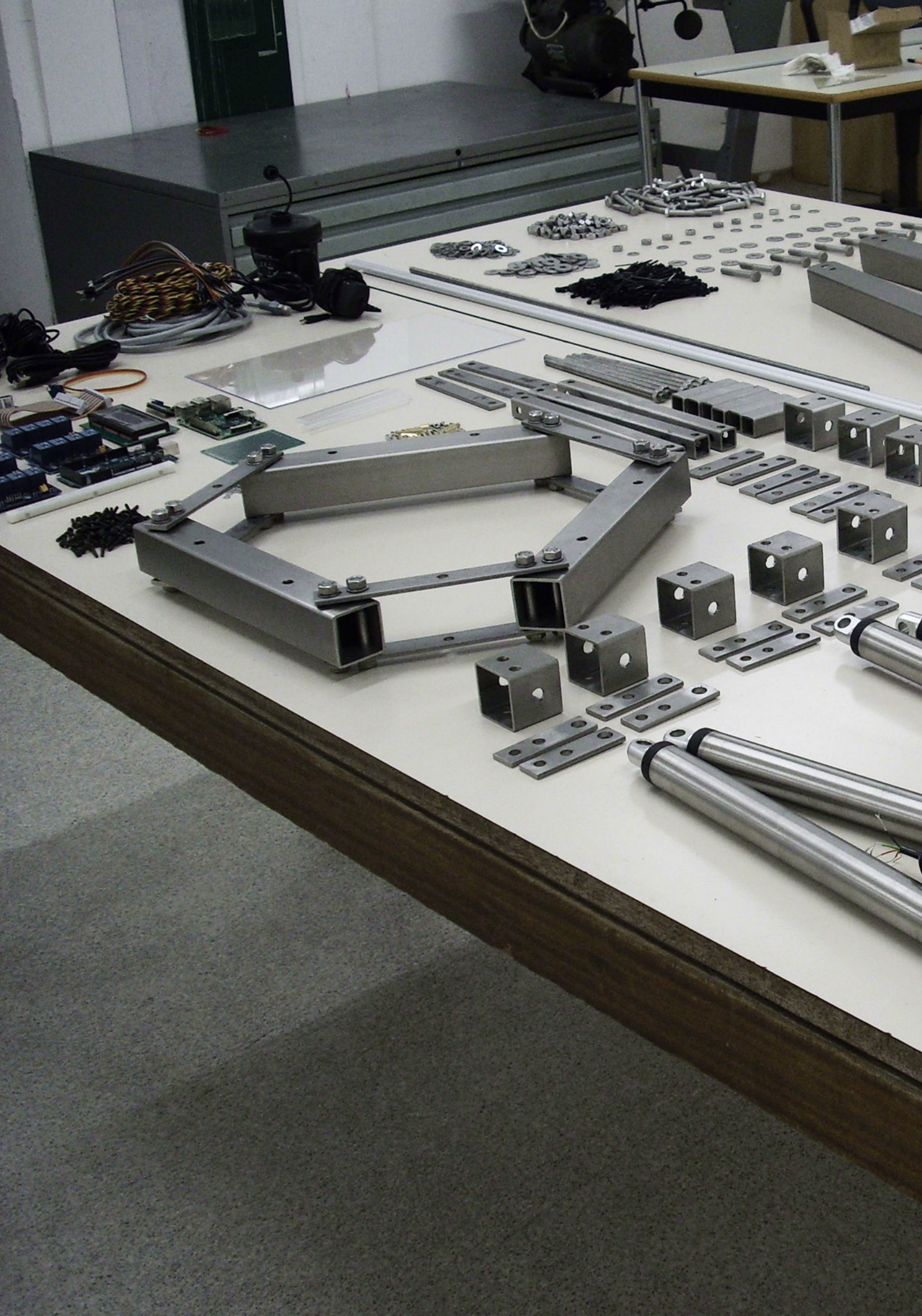


Abb.3.6.2-3 Montage des Kerns


Abb.3.6.2-4 Montage der Füße



Umsetzung Zusammenbau
Aufzeigen der Arbeitsschritte
Gleitlager und fixe Verbindungen

Abb.3.6.3-1 Detailansicht eines Gleitlagers

Abb.3.6.3-2 Zusammenbau eines Fußes

Abb.3.6.3-3 Verbindung der Füße mit dem Kern

Folgeseite:
Abb.3.6.4-1 Detailansicht Platine des bAm

Umsetzung Elektronik
Aufzeigen der Arbeitsschritte
Zusammensetzten der Platine

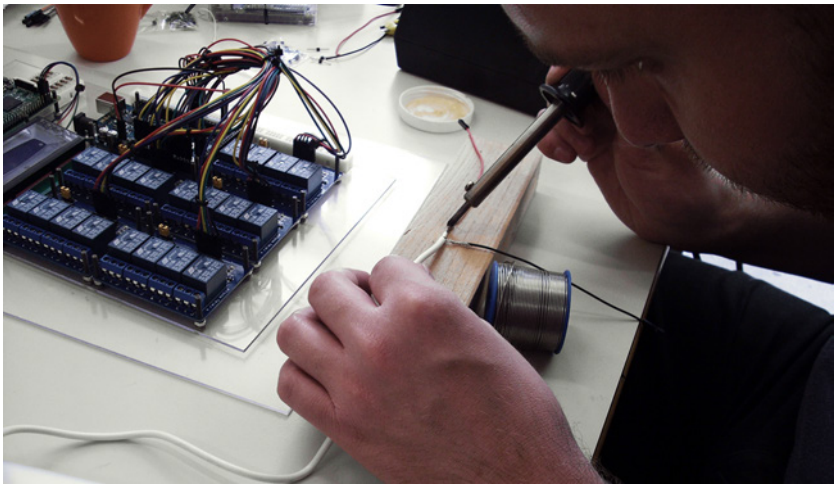


Abb.3.6.4-2 Löten der Stromversorgung

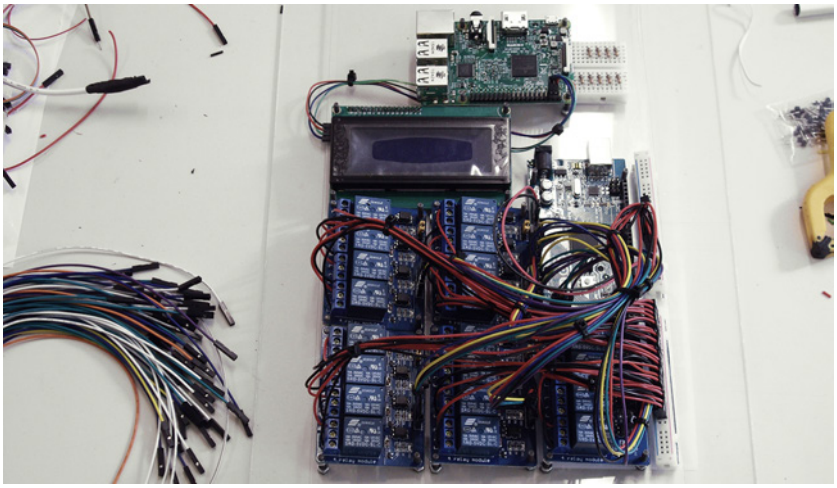


Abb.3.6.4-3 Bau der Platine

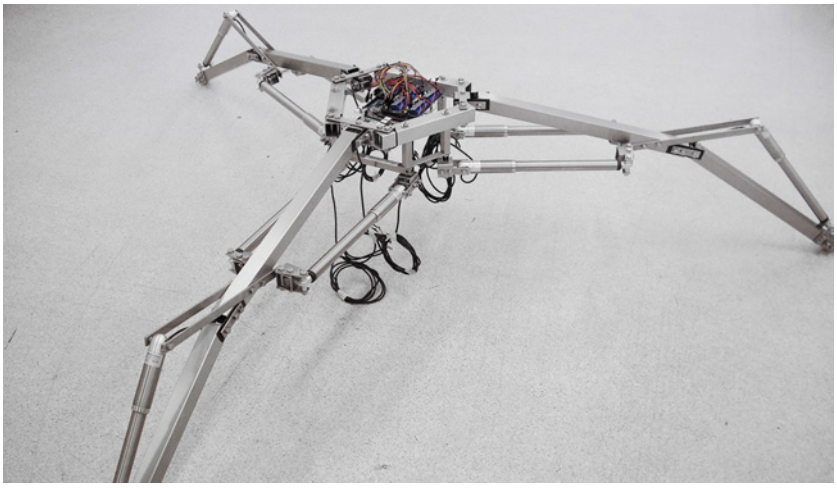
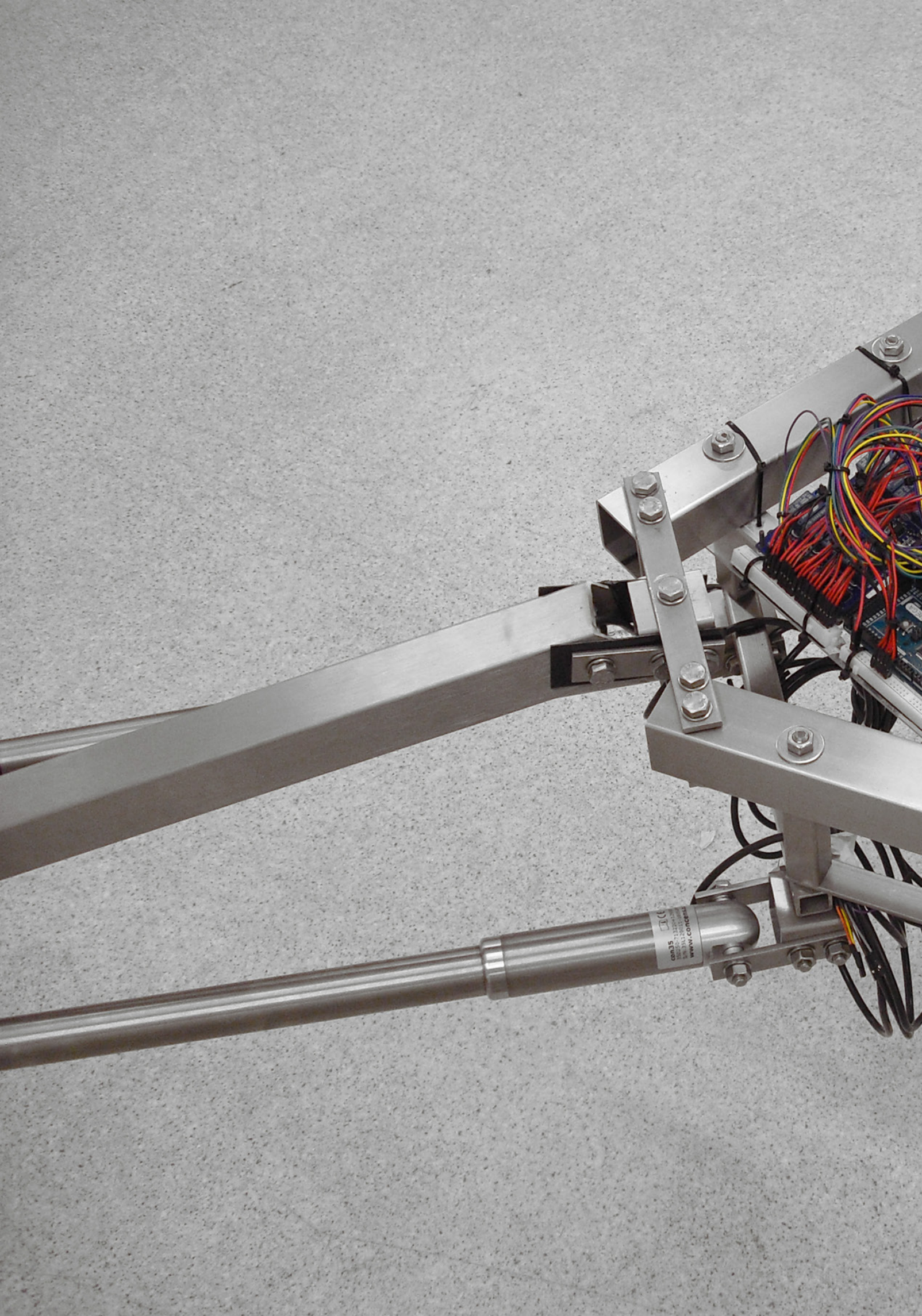
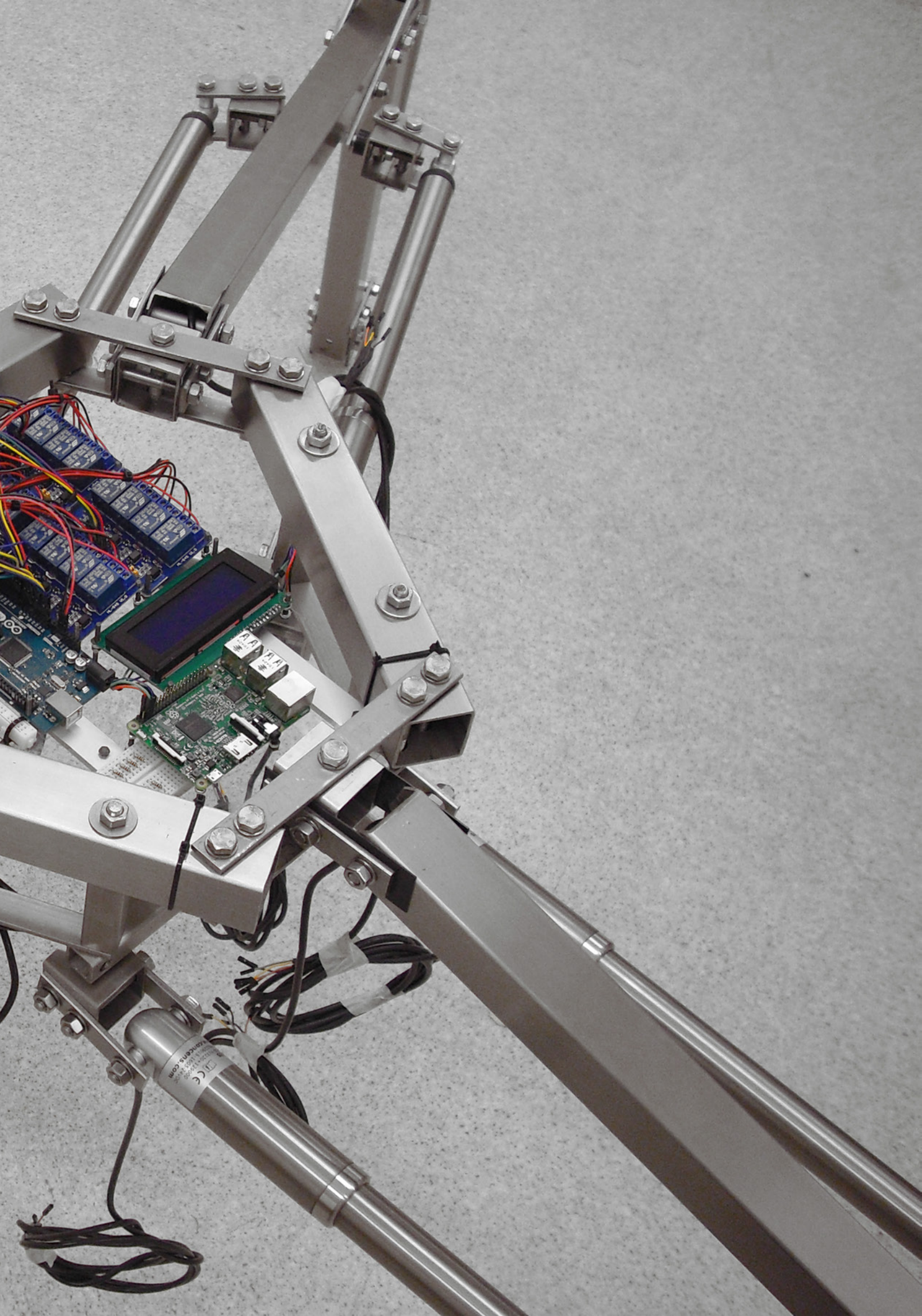


Abb.3.6.4-4 Platine am bAm





3.6.5 P N E U S

Folgeseite:
Abb.3.6.5-1 Befestigung der Pneus am bAm

Folgeseite danach:
Abb.3.6.5-2 Fast fertig, gestellter bAm

Umsetzung Pneus
Aufzeigen der Arbeitsschritte
Aufreißen, schweißen, zusammenfügen

Abb.3.6.5-3 Aufriss der Kissen

Abb.3.6.5-4 Verschweißen nach Zuschnitt

Abb.3.6.5-5 Verstärken der Schweißnähte

3.7 POTENZIAL

3.7.1 PERFORMATIV

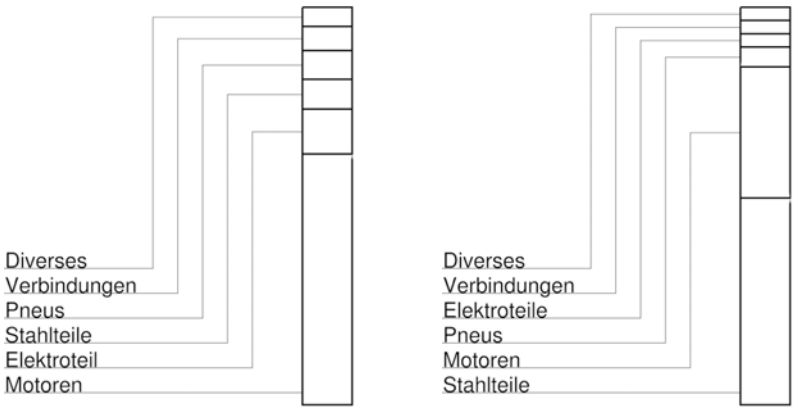
Die bAm eignen sich in erster Linie zur Überdachung von öffentlichen Plätzen. Der Fokus liegt auf Einsatzorten, die einem hohen Maß an Veränderung ausgesetzt sind. Denkbar sind beispielsweise Bahnhofsvorbereiche mit nicht gleichbleibender Besucherfrequentierung, temporäre Überdachungen von Veranstaltungen wie Konzerten, Bauplätzen mit schwankenden Windlasten und Bodentragfähigkeit.

Darüber hinaus bietet die Adaptivität, in Bezug auf die Interaktion zwischen Mensch und Raum sowie Mensch und Maschine, ein enormes Potenzial, das in den nächsten Schritten untersucht wird.

3.7.2 GEWICHT UND KOSTEN

Ein bAm deckt in etwa eine Fläche von 3,5 qm ab. Die reinen Materialkosten betragen in etwa 4.200,- Euro netto. Daraus ergibt sich ein Quadratmeterpreis von 1.200,- Euro netto. Eine Vorproduktion in Bausätzen erlaubt, dass Benutzer die Module eigenständig zusammen setzen. Der Aufbau erfolgt durch die bAm selbst. Das Gewicht eines Modules beträgt in eta 30 kg, woraus sich ein Gewicht von 8,6 kg/qm ergibt.

Folgeseite:
Abb.3.7.2-1 Beleuchteter bAm



Diagr.3.7.2-1 Kosten

Diagr.3.7.2-2 Gewicht

Aufstellung von Kosten und Gewicht
Preise sind netto angegeben
das Gewicht bezieht sich auf einen einzelnen bAm

Teil	Produkt	Einheit	Anzahl	Preis	Gesamt	kg Stück	Gesamt	Preis und Gewicht	
Füße	Formrohr QU 40 x 2 mm	0,60 m	x 6	34,06 Euro	122,62 Euro	1,89 kg/m	6,80 kg		
Kern	Formrohr QU 40 x 2 mm	0,28 m	x 3	34,06 Euro	28,61 Euro	1,89 kg/m	1,59 kg		
	Formrohr QU 20 x 2 mm	0,28 m	x 3	12,12 Euro	10,18 Euro	0,89 kg/m	0,75 kg		
	Formrohr QU 20 x 2 mm	0,07 m	x 6	12,12 Euro	5,09 Euro	0,89 kg/m	0,37 kg		
	Flachstange 20 x 4 mm	0,18 m	x 9	15,67 Euro	25,39 Euro	0,63 kg/m	1,02 kg		
Auflager	Formrohr QU 40 x 2 mm	0,04 m	x 15	34,06 Euro	20,44 Euro	1,89 kg/m	1,13 kg		
	Flachstange 20 x 4 mm	0,08 m	x 48	15,67 Euro	60,17 Euro	0,63 kg/m	2,42 kg		
	Flachstange 20 x 4 mm	0,40 m	x 6	15,67 Euro	37,61 Euro	0,63 kg/m	1,51 kg		
								7,49 % Stahlteile	52,00 %
Motoren	Linearaktuatoren	-	x 9	290,00 Euro	2610,00 Euro	1,10 kg	9,90 kg		
								63,08 % Motoren	33,00 %
Pneu	PVC-Folie	-	-	-	200,00 Euro	-	1,00 kg		
	GFK-Stange	-	-	-	100,00 Euro	-	0,50 kg		
								7,25 % Pneus	5,00 %
Verbindungen	-	-	-	-	250,00 Euro	-	1,00 kg		
								6,04 % Verbindungen	3,33 %
Raspberry Pi	-	-	x 1	60,75 Euro	60,75 Euro	-	-		
Arduino	-	-	x 1	36,83 Euro	36,83 Euro	-	-		
Webcams	-	-	x 3	16,66 Euro	49,98 Euro	-	-		
Luftpumpe	-	-	x 1	20,00 Euro	20,00 Euro	-	-		
Baromter	-	-	x 3	20,00 Euro	60,00 Euro	-	-		
Spannungsmesser	-	-	x 9	20,00 Euro	180,00 Euro	-	-		
Ultraschallsensor	-	-	x 3	20,00 Euro	60,00 Euro	-	1,00 kg		
								11,30 % Elektroteile	3,33 %
Diverses	-	-	-		200,00 Euro	-	1,00 kg		
								4,83 % Diverses	3,33 %
Summe	-	-	-		4137,66 Euro	-	30,00 kg	100,00 %	100,00 %

Tab.3.7.2-1 Aufstellung

3.7.3 S T A T I S C H

Die Analysen zeigen die Kräfteverteilung innerhalb eines bAm. Das Stabwerksprogramm Star2 war eine große Hilfe, die Module zu analysieren und schließlich systemintern zu optimieren. Stellvertretend wurden in Summe jeweils 100 N als Lastfall angenommen. Eine Erhöhung des Lastfalls führt zu einer Erhöhung der jeweiligen Belastung der Stäbe im gleichem Maße. Sämtliche Knoten wurden als gelenkig definiert. Biegesteife Elemente wurden in entsprechende gelenkige Dreiecke zerlegt. Berechnet wurde die Normalkraft, welche auf die einzelnen Stäbe wirkt. Moment und Querkraft aller Elemente des Stabwerks ergaben bei sämtlichen Berechnung Null. Als kritische Elemente werden die Motoren angenommen.

Der erste Lastfall ist das Bewegen der bAm. Wenn davon ausgegangen wird, dass der 30 kg schwere bAm die Last auf alle drei Füße gleichmäßig verteilt, können 100 N als Lastfall angenommen werden. Diese Belastung ist bei sämtlichen Motoren erfüllt. Wenn die Beiden Motoren S0 als kritischsten Punkt angenommen werden, könnte das Gewicht der bAm auf über 50 kg erhöht werden.

Verweis auf:
Gewicht, Seite 264

Das Heben stellt den kritischsten Lastfall dar. Wenn ebenfalls die Motoren S0 als kritischsten Punkt angenommen werden, kann der Lastfall in Etwa 45 kg betragen. Es könnten also eineinhalb bAm gestemmt werden. In einer Kettenlinie teilt sich die Last auf zwei Punkte auf. Jedoch wird die Last zu den Seiten hin, wie im Lastfall Heben beschrieben, erhöht. Angenommen wird eine Spannweite von drei Modulen, also eine Länge von 8,7 m die sich in Abhängigkeit der Bogenform verringert.

Verweis auf:
Aufrichten der bAm, Seite 186

Welche Tragfähigkeit sich daraus für ein einzelnes Modul ergibt, ist mit einem Statiker in den nächsten Schritten abzuklären und zu verfeinern. Sollte eine Erhöhung von Querschnitt oder Wandstärke der Edelstahlprofile erforderlich sein, wirkt sich das unmittelbar auf das Gewicht und somit auf die gesamte Gleichung aus.

Verweis auf:
Star2, Seite 275

Stab	S0	S1	S2	S3	S4	S5
Art	Motor x 2	Träger	Hilfsstab	Träger	Träger	Motor
Bewegen	+896	-262	-584	+622	+105	+184
Heben	-1035	+250	+729	-628	-214	+226
Spannen	+24	-116	-20	+13	-104	-9

Diagr.3.7.3-1 Kosten

Potenzial aus statischer Sicht
mögliche Lastfälle eines bAm
Unterschieden in Motoren und Träger

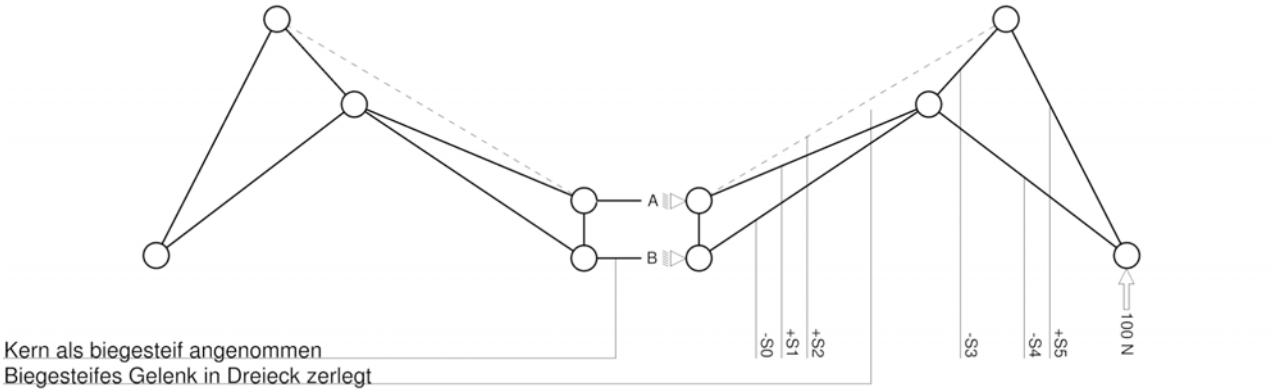


Abb.3.7.3-1 Bewegen

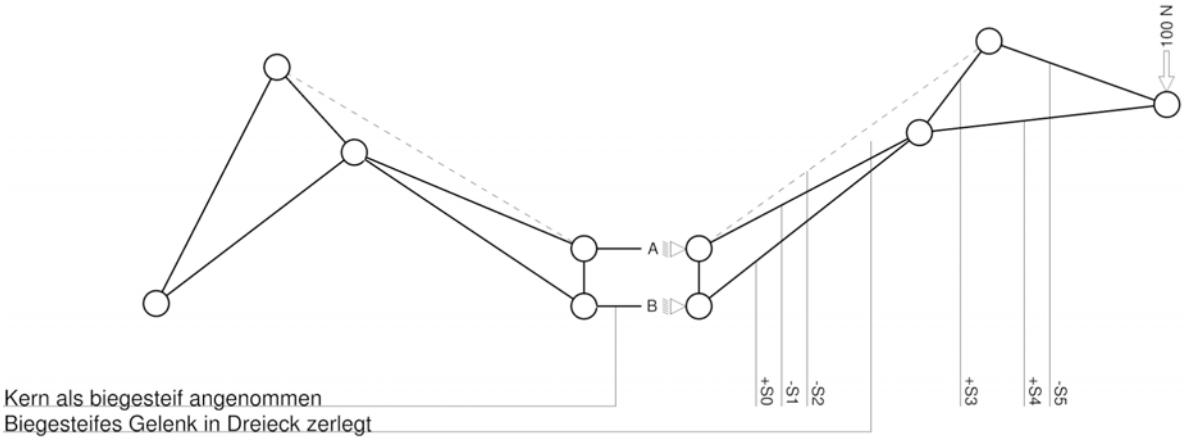


Abb.3.7.3-2 Heben

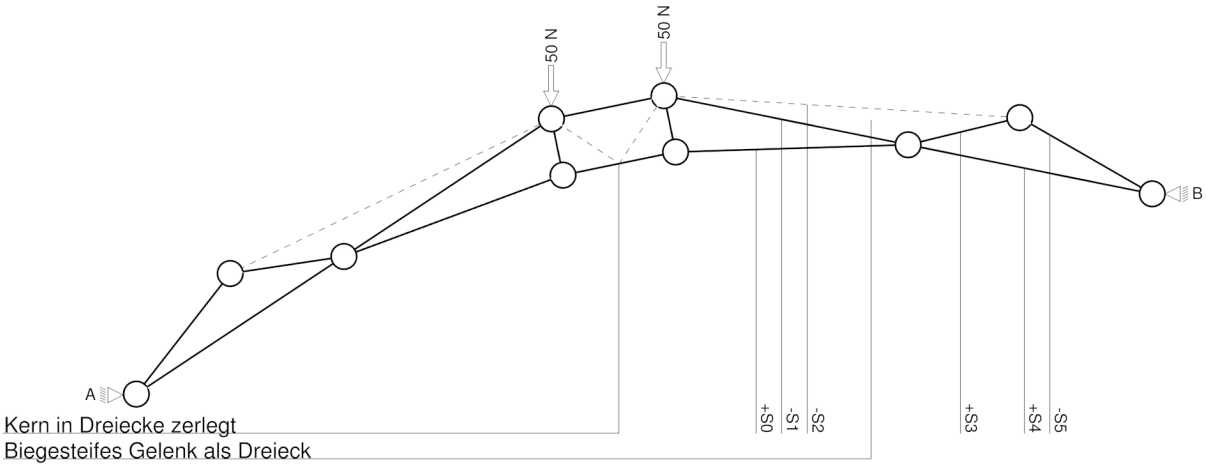


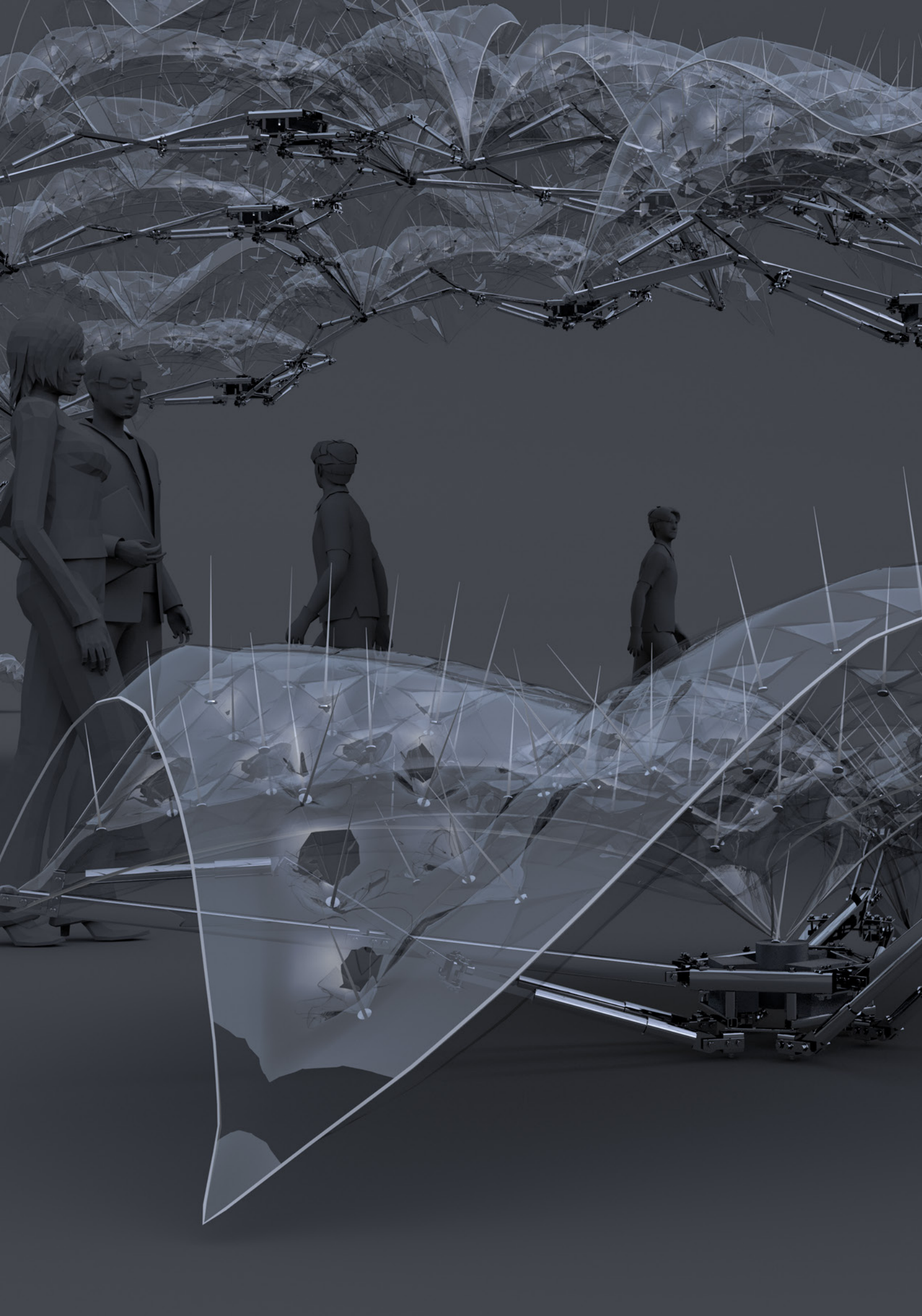
Abb.3.7.3-3 Spannen

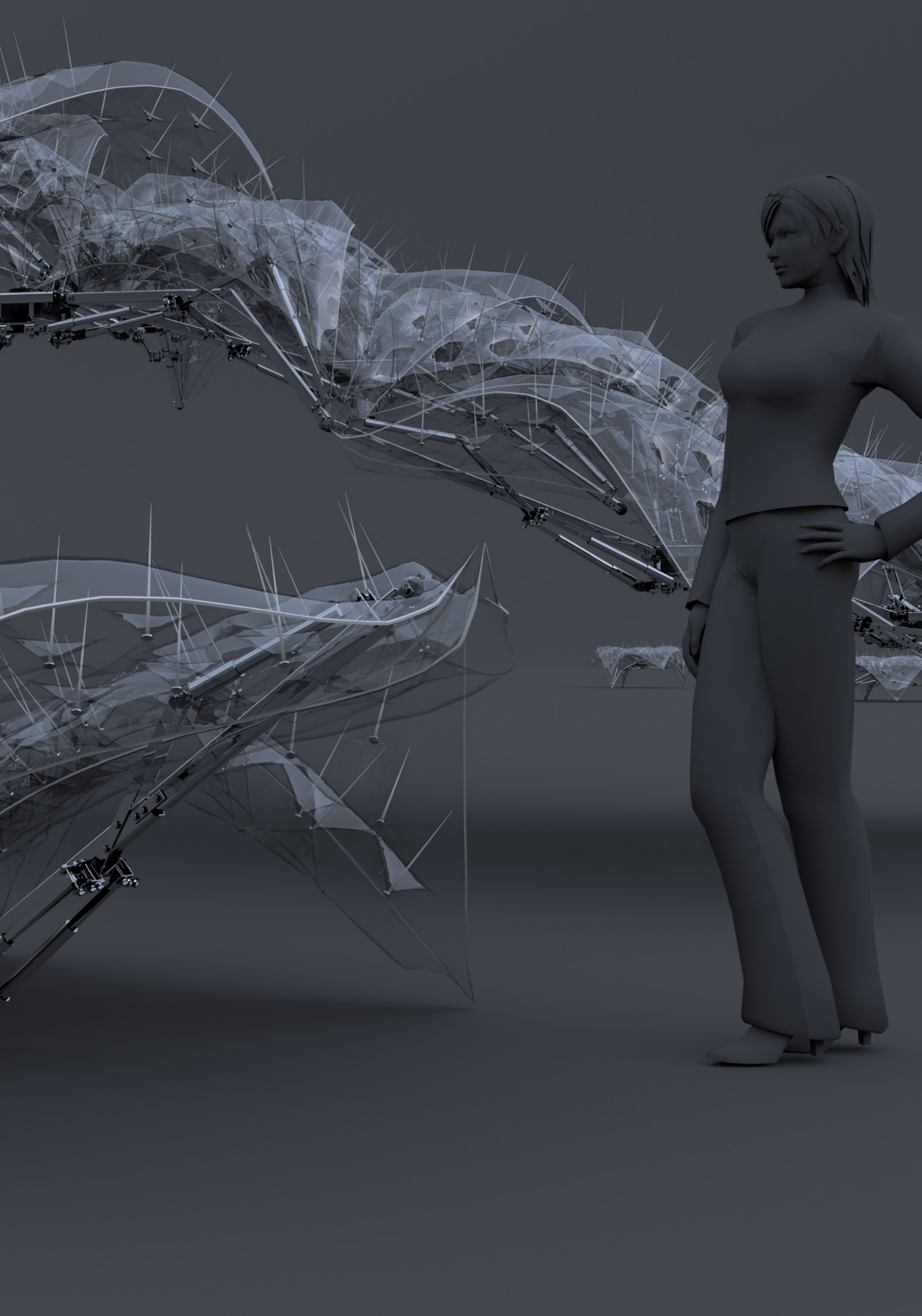
4 ABSCHLIESSEND

„Die Ewigkeit dauert lange, besonders gegen Ende“

Woody Allen

Folgeseite:
Abb.4.0.0-1 Rendering





4.1 BRENNWEITE 35 MM

Die Erforschung interaktiver Architektur verleitet dazu, den Blick über viele architekturtheoretische Themen schweifen zu lassen. Wie werden Räume, die sich intelligent anpassen können wahrgenommen? Verändert sich dadurch die Beziehung zwischen Mensch und Raum? Ist Architektur dann Maschine oder Bauwerk? Geht sämtlicher Ortsbezug verloren, oder entsteht durch Bewegung eine neue Form von Vernetzung? Was bedeutet Nutzerpartizipation, wenn das Gebaute nach Fertigstellung editierbar bleibt? Ein Feld schier endloser Weite spannt sich auf. Es läge nahe zu behaupten: Im Rahmen der Entwicklung eines einzelnen Projektes können diese Fragen nicht beantwortet werden. Tatsache ist: Jene Fragen müssen bei der Entwicklung permanent beantwortet werden. Teils bewusst, teils unbewusst, manchmal angebracht, oft auch fehlerhaft, aber meist wortlos, in Form von entstandener Geometrie. Akademische Forschung ist eine ideale Voraussetzung, jene Fragen anhand von konkreten Referenzmodellen zu ergründen und auszutesten.

4.2 MEHRWERT

Entstandener Mehrwert ist, dass die Arbeit als Referenzmodell für die Entwicklung interaktiver Architektur dienen kann. Der Bau von kinematischen Prototypen, die Simulation, die Steuerung und weitere Themen werden aus Sicht der Architektur beleuchtet. Diese Erkenntnisse wurden für den Fachbereich aufgearbeitet und sind auf die Erforschung anderer kinematischer Systeme übertragbar. In weiterer Folge soll ein Download-Bereich die gewonnenen Erkenntnisse einer breiteren Masse einfacher zur Verfügung stellen.

Besuchen Sie:
www.mueller-christoph.com

4.3 ZUSATZ

4.3.1 SOFTWARE

- Blender, Digital Content Creation, www.blender.org
- Arduino, Steuerung Micro-Controller, www.arduino.org
- Maxima, Computeralgebrasystem, maxima.sourceforge.net
- ToPy, Topologie-Optimierung, github.com/williamhunter/topy
- BESO2D, Topologie-Optimierung, www.rmit.edu.au
- Star2, Stabwerks-Programm, www.ibb.uni-stuttgart.de
- Orange, Programm für maschinelles Lernen, orange.biolab.si
- ParaView, Anzeigen von 3D-Modellen, www.paraview.org
- LaTex, Layout und Textsatz, www.latex-project.org

Für die im Projekt animierten Menschen wurde Makehuman sowie Makewalk verwendet. Die BVH-Dateien zur Steuerung werden freundlicherweise vom ACCAD der Ohio State University zur kostenlosen Verwendung zur Verfügung gestellt. accad.osu.edu/research/mocap

4.3.2 ABKÜRZUNGEN UND BEGRIFFE

- bAm, bewegendes Architekturmodul, Eigenname der im Rahmen dieser Arbeit erzeugten Roboter
- DCC, Digital Content Creation, Programme zum Erstellen, Verändern und Animieren von digitalen Inhalten
- VGT, Variable Geometry Truss, von Koryo Miura und Hiroshi Furuya entwickeltes Tragsystem für Kragarme im Weltraum

4.4 VERZEICHNISSE

4.4.1 ABBILDUNGEN

4.4.2 DIAGRAMME

4.4.3 TABELLEN

4.4.4 TEXTE

James Cameron, „Avatar“, 2009, zu finden auf der DVD: „Avatar - Aufbruch nach Pandora“, 20th Century Fox, April 2010, Deutsch und Englisch, EAN: 4010232049841, Zwischen 14:10 und 14:15 15

Ludwig Hilberseimer, „Konstruktion und Form“, in: „G - Zeitschrift für elementare Gestaltung“, hrsg. Hans Richter, Berlin, Juni 1924, Seite 14 bis 16 - übernommen aus: Vittorio Magnago Lampugnani, Ruth Hanisch, Ulrich Maximilian Schumann, Wolfgang Sonne, „Architekturtheorie 20. Jahrhundert: Positionen, Programme, Manifeste“, Hatje Cantz Verlag, Ostfildern-Ruit, 2004, ISBN: 3-7757-1375-1, Seite 94 15

Michael Hansmeyer und Benjamin Dillenburger, „Digital Grotesque“, 2013, www.michael-hansmeyer.com, unter: „Projects/Digital Grotesque“, Zugriff: 20. September 2016 16

Le Corbusier und Pierre Jeanneret, „Fünf Punkte zu einer neuen Architektur“, in: Alfred Roth, „Zwei Wohnhäuser von Le Corbusier und Pierre Jeanneret“, Akademischer Verlag Dr. Fr. Wedekind & Co., Stuttgart, 1927, Seite 5 bis 7, übernommen aus: Vittorio Magnago Lampugnani, Ruth Hanisch, Ulrich Maximilian Schumann, Wolfgang Sonne, „Architekturtheorie 20. Jahrhundert: Positionen, Programme, Manifeste“, Hatje Cantz Verlag, Ostfildern-Ruit, 2004, ISBN:3-7757-1375-1, Seite 112 29

Koryo Miura, Hiroshi Furuya und Kenichi Suzuki, „Variable geometry truss and its application to deployable truss and space crane arm“, in: „Acta Astronautica“, Band 12, Ausgabe 7, Juli - August, 1985, Seite 599 32

Alvaro Ferrán Cifuentes, github.com/alvaroferran, unter: „Blender Controller“, Zugriff: 20. September 2016 40

AMAXeu.amaxshop.com, unter: „Servo AMAX G09AP Micro“, Zugriff: 20. September 2016 40

Lothar Papula, „Mathematische Formelsammlung für Ingenieure und Naturwissenschaftler“, Auflage 10, Vieweg+Teubner | GWV Fachverlage GmbH, Wiesbaden, 2009, ISBN:978-3-8348-0757-1, Seite 26 und 28 40

ELRA Antriebstechnik Vertriebs GmbH, „Con 35“, in: „DC-LINEARANTRIEBE“, 48 / 2014, Seite 15 bis 17 46

Robert Van Deest, „Motor Control using Relays“, www.bpesolutions.com, unter: „technical“, „Hardware References“, 2007, Zugriff: 20. September 2016 46

wiki.blender.org, „Geometry“, wiki.blender.org, unter: „Manual“, „Extensions“, „Python“, „Geometry“, 2013, Zugriff: 20. September 2016 50

Schema Hydraulik GmbH, „Normzylinder“, www.hydraulik-webshop.com, unter: „Kataloge“, „Zylinderkatalog“, „Hydraulikzylinder S10060“, Zugriff: 20. September 2016 62

Parker Hannifin GmbH, „Lightraulics Composite Hydraulic Cylinders“, Katalog HY07-1410/UK, unter: „Lightraulics C-Series Cylinders“, 01 / 2016, Seite 13 68

Ein spannender Weblink mit Infos über Ron Resch, www.ronresch.org, Zugriff: 20. September 2016 86

Adolf Loos, „Das Princip der Bekleidung“, in: „Neue Freie Presse“, Wien, 4.9.1898, Seite 6, übernommen aus: Vittorio Magnago Lampugnani, Ruth Hanisch, Ulrich Maximilian Schumann, Wolfgang Sonne, „Architekturtheorie 20. Jahrhundert: Positionen, Programme, Manifeste“, Hatje Cantz Verlag, Ostfildern-Ruit, 2004, ISBN:3-7757-1375-1, Seite 25 bis 26 90

Xiaodong Huang und Mike Xie, „Evolutionary Topology Optimization of Continuum Structures: Methods and Applications“, John Wiley & Sons Ltd, Chichester, 2010, ISBN: 978-0-470-74653-0, Seite 189 bis217 114

William Hunter, „Predominantly solid-void three-dimensional topology optimization using open source software“, Department of Mechanical and Mechatronic Engineering, University of Stellenbosch, Republic of South Africa, betreut von: Professor Albert Groenwold, März 2009, Seite 30 bis 34 120

Mark Elling Rosheim, „Leonardo's Lost Robots“, Springer-Verlag, Berlin / Heidelberg, 2006, ISBN: 978-3-540-28440-6, Seite 69 128

Noriaki Kurokawa, „Capsule Declaration“, in: „Space Design“, Tokio, 52, März 1969, Seite 50, deutsche Übersetzung: Chihaya Koyama Luethi, übernommen aus: Vittorio Magnago Lampugnani, Ruth Hanisch, Ulrich Maximilian Schumann, Wolfgang Sonne, „Architekturtheorie 20. Jahrhundert: Positionen, Programme, Manifeste“, Hatje Cantz Verlag, Ostfildern-Ruit, 2004, ISBN:3-7757-1375-1, Seite 237 130

Eric Clapton, „Let it grow“, 1974, zu finden auf der CD: „461 Ocean Boulevard“, Universal Music, September 1996, EAN: 0731453182127, Zeit 0:00 bis 0:12 143

Gerardo Beni und Jing Wang, „Swarm Intelligence in Cellular Robotic Systems“, in: „Robots and Biological Systems: Towards a New Bionics?“Band 102, aus der Serie, „NATO ASI Series“, 1993, ISBN: 978-3-642-63461-1, Seite 703 144

Craig Reynolds, „Flocks, Herds, and Schools: A Distributed Behavioral Model“, in: „SIGGRAPH '87 Proceedings of the 14th annual conference on Computer graphics and interactive techniques“, ACM, New York, 1987, ISBN:0-89791-227-6, Seite 25 145

Eurostemcell, www.eurostemcell.org, unter: „Factsheets“, „Regeneration: Was ist das und wie funktioniert es?“, Oktober 2011, Zugriff: 19 September 2016 174

Martin Kompf, www.kompf.de, unter: „Wetter“, „Barometer“, „Barometer mit dem Raspberry Pi und dem I2C Luftdrucksensor BMP085“, Zugriff: 20. September 2016 174

Adafruit Industries LLC, github.com/adafruit, unter: „Adafruit Python BMP“, Zugriff: 20. September 2016 174

Pascal van Kooten, github.com/kootenpv, unter: „yagmail“, Zugriff: 20. September 2016 174

Dejan Nedelkovski, howtomechatronics.com, unter: „Tutorials“, „Arduino“, „Ultrasonic Sensor HC-SR04 and Arduino“, Zugriff: 20. September 2016 176

Tim Eckel, playground.arduino.cc, unter: „User Code Library“, „Libraries“, „New Ping Example“, Release 1.8, 30. Julie 2016, Zugriff: 20. September 2016 176

Anthony Oliver, gist.github.com/xamox, unter: „balltrack.py“, Zugriff: 20. September 2016 178

Pink Floyd, „Hey You“, 1979, zu finden auf der CD: „The Wall“, EMI, Oktober 1994, EAN:0724383124329, Zeit: 4:25 bis 4:35 184

Jesper Mosegaard, cg.alexandra.dk, „Mosegaards Cloth Simulation Coding“, Juni 2009, Zugriff: 20. September 2016 186

Ronnie James Dio, 1983, zu finden auf der CD: „Holy Diver“, Mercury Records Limited, März 2012, ASIN: B007B0LXAI, Zeit: 0:15 bis 0:25 192

Anthony Zhang, pypi.python.org/pypi, unter: „Speech Recognition“, Version 3.4.6, 2016 192

Jonathan Duddington, „eSpeak text to speech“, espeak.sourceforge.net, Version 1.48.0.4, April 2014, Zugriff: 20. September 2016 192

Georg Fenady, Winrich Kolbe, Bernard L. Kowalski, Bob Bralver, Sidney Hayers, „Knight Rider“, 1982, zu finden auf der DVD: „Knight Rider-Season 1 “, Universal Pictures Customer Service Deutschland/Österreich, März 2012, EAN: 5050582890389, Zeit: Im Vorspann von 0:15 bis 0:25 194

Carlo Mascellani, rpihome.blogspot.co.at, unter: „Face detection with Raspberry Pi“, März 2015, Zugriff: 20. September 2016 194

Orange, docs.orange.biolab.si, unter: „Tutorials“, 2015, Zugriff: 20. September 2016 202

The Smashing Pumpkins, „Bullet with butterfly wings“, 1995, zu finden auf der Doppel-CD: The Smashing Pumpkins, „Mellon Collie And The Infinite Sadness“, Virgin (Universal Music), Oktober 1995, ASIN: B000024JHZ, Zeit: 0:50 bis 0:55 214

Georg Wilhelm Friedrich Hegel, „System der Wissenschaft“, „Erster Theil, die Phänomenologie des Geistes“, Bamberg und Würzburg, bey Joseph Anton Goebhardt, 1807, Seite 91 244

Woody Allen, „Die Ewigkeit dauert lange, besonders gegen Ende“, www.zitate-online.de, unter: „Woody Allen“, Zugriff: 20. September 2016 271

4.4.5 SCRIPTE

Scp.3.4.2-1 Umsetzung des maschinellen Lernens der bAm mittels Orange in Python, Christoph Müller, „Bewegende Architektur“, 2016, basierend auf den Tutorials: Orange, docs.orange.biolab.si, unter: „Tutorials“, 2015, Zugriff: 20. September 2016 203

4.4.6 ZEICHNUNGEN

4.5 LEBENSLAUF

4.5.1 ÜBERBLICK

Mein Interessengebiet sind digitale Formfindungsmethoden, sowie Interaktivität und Robotik in Architektur. Seit 2013 unterrichte ich am Institut für Architekturtheorie der TU-Wien. Neben Lehre und Forschung arbeitete ich als freier Architekturschaffender in diversen Büros und verfolge immer wieder eigene Projekte.

- Geboren in Zwiesel, Niederbayern, 1985
- Architekturstudium an der FH-Erfurt von 2005 bis 2007
- Architekturstudium an der TU-Wien von 2007 bis 2012
- Diplom mit Auszeichnung bei Dörte Kuhlmann, 2012
- Hybrid aus Lehrender, Forscher und Architekturschaffender seit 2013

4.5.2 LEHRE

Im Zeitraum der Arbeit sind folgende Lehrveranstaltungen entstanden:

- Entwerfen, 8 h, „Zentrum für Architektur“ , 2013S
  Kristina Schinegger, Stefan Rutzinger, Christoph Müller
- Entwerfen, 8 h, „The Art of Re-Creation“, 2013W
  Kristina Schinegger, Stefan Rutzinger, Christoph Müller
- Entwerfen, 8 h, „Operations of the Formless“, 2014S
  Kristina Schinegger, Stefan Rutzinger, Christoph Müller
- Entwerfen, 4 h, „Atlas of Material Formations“, 2014S
  Kristina Schinegger, Stefan Rutzinger, Christoph Müller
- Entwerfen, 8 h, „Fibrosity“, 2014W
  Kristina Schinegger, Stefan Rutzinger, Christoph Müller
- Entwerfen, 4 h, „Beautifully Grotesque“, 2014W
  Michael Hansmeyer, Christoph Müller
- Entwerfen, 8 h, „The Alchemy of Simulation“, 2015S
  Kristina Schinegger, Stefan Rutzinger, Christoph Müller
- Entwerfen, 8 h, „Die Transfiguration des Gewöhnlichen“, 2015W
  Dörte Kuhlmann, Heimo Schimek, Christoph Müller
- Entwerfen, 8 h, „Resilient Matter“, 2015W
  Kristina Schinegger, Stefan Rutzinger, Christoph Müller
- Entwerfen, 8 h, „Frankly Architecture“, 2016S
  Dörte Kuhlmann, Heimo Schimek, Christoph Müller
- Entwerfen, 8 h, „Beyond Accidentism“, 2016S
  Kristina Schinegger, Stefan Rutzinger, Christoph Müller
- Entwerfen, 4 h, „Before Sunrise“, 2016S
  Kristian Faschingeder, Christoph Müller
- Modul, 2 h, „Präfabrikation“, 2016S
  Manfred Berthold, Christoph Müller

4.5.3 PROJEKTE

Insgesamt habe ich im Rahmen der letzten 10 Jahre für diverse Büros über 40 Architekturwettbewerbe abgeschlossen und an diversen Direktaufträgen mitgewirkt. Eine kleine Auswahl von eigenen Projekten ist:

- Widerlager, Wettbewerbsgewinn des TU-weiten Wettbewerbs Seilbahn Feuerkogel, 2008
- Biotech, Wettbewerbsgewinn des international CAD-Wettbewerbs Architects Jury, 2010
- Panta Rhei, Wettbewerbsbeitrag für die Documenta in Kassel, 2011
- New Beletage, Wettbewerbsbeitrag zum Schindler Award, 2012

4.5.4 PUBLIKATIONEN UND KONFERENZEN

Die folgenden Publikationen wurden veröffentlicht und ich nahm an den genannten Konferenzen als Vortragender teil:

- advanced building skins, Proceedings of the International Conference, On Building Envelope Design and Technology, 23-24 April 2015, Graz, Austria, ISBN: 978-3-85125-397-9, Seite 24 bis 25
- Operationen des Formlosen, Pneumatische Installation, ISBN-978-3-902844-50-7, Seite 24 bis 25
- New Beletage, Diplomarbeit an der TU-Wien 2012

- advanced building skins, Internationale Konferenz in Graz, 2014
- 1. Wiener DoktorandInnen Symposium der Architektur, Wien, 2015
- Rese Arch, Internationale Konferenz in Bratislava, 2015

Abb.4.5.4-1 Foto mit Christoph Müller

4.6 DANKSAGUNG

Herzlich bedanken möchte ich mich bei meiner Familie, für die vielseitige Unterstützung in den letzten Jahren.

Bei Thomas, Michael und Johannes Kropatschek, für die vielen Informationen und tatkräftige Hilfe beim Metallbau.

Bei Carolin Lotz, P. Michael Schultes und Ilina Kokaleska vom Verein experimonde | die Welt des Experiments e.V., für das Erstellen der Pneukissen und die Bereitstellung von Werkzeug und Räumlichkeiten.

Bei den Firmen ELRA Antriebstechnik GmbH und Fix Metall GmbH, für die Beratung und ein großes Entgegenkommen bei den Materialkosten.

Bei Kristina Schinegger und Stefan Rutzinger, Oliver Schürer, Kristian Faschingeder, Mathias Mitteregger, Sebastian Leschhorn, Thomas Pachner, Christoph Kolbeck und weiteren Freunden und Kollegen, für die anregenden Gespräche und vielseitige Hilfe.

Beim Dekanat für Architektur und Raumplanung der TU-Wien, für die administrative, organisatorische und auch finanzielle Unterstützung in Form von drei Stipendien.

Ganz besonderer Dank geht an Manfred Berthold und Margit Gföhler, für die außerordentliche Betreuung.

Bedanken möchte ich mich zudem bei Ihnen, für ihr Interesse und das Lesen der Arbeit.