DIPLOMARBEIT

VON DER FOTOKOPIE ZUR TOPOKOPIE

TRANSFORMATION EINES ZWEIDIMENSIONALEN PROZESSES IN DIE DRITTE DIMENSION

ausgeführt zum Zwecke der Erlangung des akademischen Grades eines Diplom-Ingenieurs unter der Leitung von

Univ.Prof.Arch.Dipl. Ing. Christian Kern
E264|2
Institut für Kunst und Gestaltung - Dreidimensionales Gestalten und Modellbau

eingereicht an der Technischen Universität Wien Fakultät für Architektur und Raumplanung

von David Paul Schwärzler
0426985

Wien, am

INHALTSVERZEICHNIS

VON DER FOTOKOPIE ZUR TOPOKOPIE

TRANSFORMATION EINES ZWEIDIMENSIONALEN PROZESSESIN DIE DRITTE DIMENSION

Die vorliegende Arbeit beschäftigt sich mit Verfahren zur Digitalisierung dreidimensionaler Oberflächen sowie computergesteuerter Produktionsmethoden.

Aufbauend auf eine Analyse und Evaluierung aktueller 3D-Scann- und Rapid-Manufacturing-Verfahren, wurde ein eigener Prozess entwickelt. Dieser ist mit dem Prozess des Fotokopierens vergleichbar, wobei eine weitere Dimension bei der Ein- und Ausgabe hinzugefügt wurde, Farbinformationen jedoch entfallen.

Physische Objekte werden arrangiert und das Relief der Objekte digital erfasst. In einer eigens entwickelten Software wird der Scann als Graustufenbild angezeigt. Die jeweiligen Grauwerte geben die Höheninformation wieder.
Das Bild kann bei Bedarf sowohl direkt in der Software, als auch mit externen Bildbearbeitungsprogrammen manipuliert werden.
Mit speziell konzipierten „Vakuumschnitzwerkzeugen“ wird eine Negativform des fertigen Entwurfs aus Gusssand geschnitzt und mit Kunststein ausgegossen.

Das Verfahren eignet sich zur Anwendung in unterschiedlichen Bereichen der Architektur und Kunst.

FROM PHOTOCOPYING TO TOPOCOPYING

MOVING A TWO DIMENSIONAL PROCESS TO THE THIRD DIMENSION

The following paper deals with different methods of digitization for complex surfaces as well as computer numerically controlled production methods.

Based on an analysis and evaluation of current technologies in the field of 3D-Scanning and Rapid-Manufacturing a unique process was developed. This process bares resemblance to the action of photocopying but featuring an additional spatial dimension for process in- and output, with the omission of color.

Initially the user arranges physical objects. Subsequently the object relief is digitally recorded. The scan result is displayed as a grayscale image within a specifically developed software solution. Within this grayscale image the individual shades of grey represent the relief height information. According to the users preference the image can be modified within the scan-software or using an external image manipulation programm. Using specifically developed „vacuum-carving-tools“ a mold is carved into a bed of foundry sand and the final object can be cast.

Possible use cases for this process include various applications in the field of Architecture and Art.

I EINLEITUNG

Rapid Prototyping Verfahren sind in den letzten Jahren immer populärer geworden. Sie machen es möglich Kleinserien bis zur Losgröße „eins“ relativ einfach zu verwirklichen. Als 3D-Drucker ist ihnen der Durchbruch in den Mainstream gelungen. Hauptgründe dafür sind niedriger Preis, einfache Handhabung und die Möglichkeit komplexe Geometrien in verhältnismäßig kurzer Zeit herzustellen. Diese Maschinen haben es zwar noch nicht in den Durchschnittshaushalt geschafft, sie sind jedoch bereits als Gesprächs- und Diskussionsthema präsent. Dabei ähneln die Diskussionsthemen denen eines 2D Druckers. Was kann, darf oder darf nicht gedruckt werden? Wo sind die Grenzen des Formates und der Haltbarkeit? Wie schaut es mit dem Urheberrecht aus? Woher bekommt man die Daten zum Drucken und wie kann man diese selbst generieren?

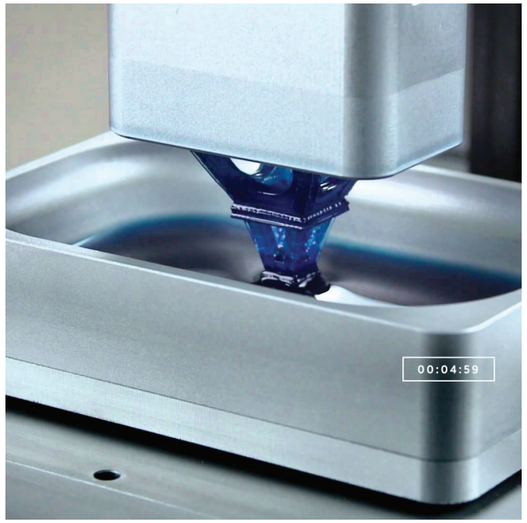
Mit eine große Triebkraft für die Verbreitung und die Aufmerksamkeit, die besonders FDM also 3D Kunststoffdruckern zuteil wird, ist die „Maker-Subkultur“. Unter diesem Begriff werden Personen zusammengefasst, die Objekte oder auch Maschinen selbst herstellen wollen und häufig gewonnenes Wissen sowie Baupläne und Software der Allgemeinheit zur Verfügung stellen. Firmen können so einen Teil der Entwicklungsarbeit einsparen und noch günstigere Geräte anbieten – Geräte, die für noch mehr Endverbraucher erschwinglich sind. Durch das gesteigerte Interesse für Rapid Prototyping Verfahren werden bestehende Technologien schneller weiterentwickelt und neue Anwendungsgebiete erforscht.

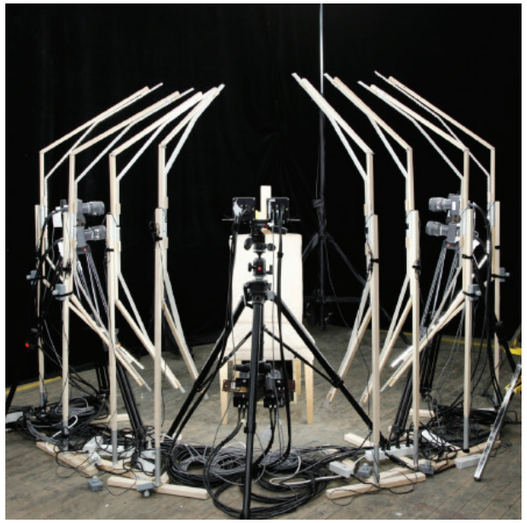
Auch in der Architektur und Kunst werden Einsatzmöglichkeiten für 3D-Druck und andere Rapid Prototyping Verfahren erforscht. Beispiele dafür reichen von der Herstellung 3D-gedruckter Miniaturabbilder von Ausstellungsbesuchern[1], über die computergesteuerte Anordnung von Ziegelsteinen[2] und das CNC-Fräsen von Betongussformen bis hin zu Konzepten für vollständig gedruckte Häuser.

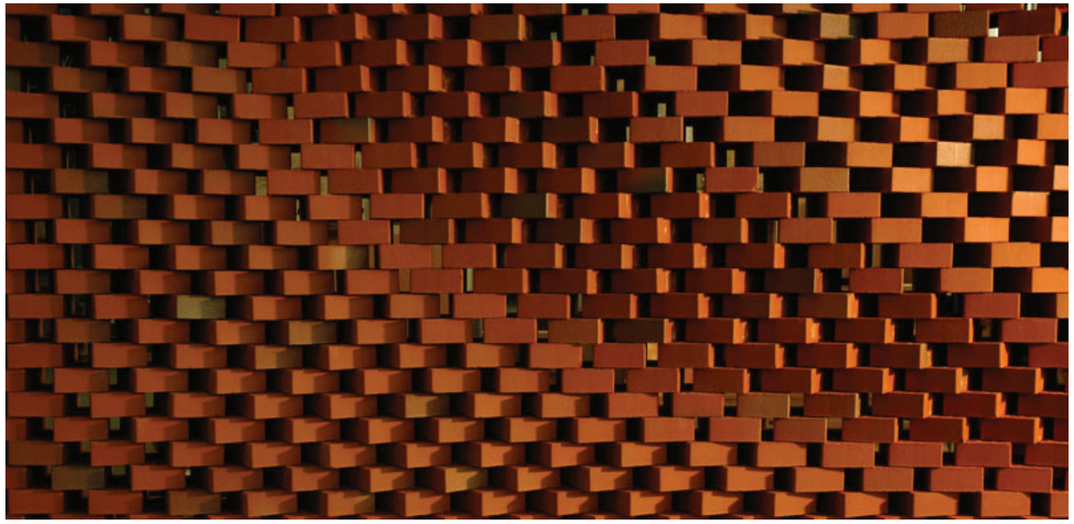
Auch wenn es technisch und finanziell möglich ist, beinahe jede beliebige Form zu produzieren, stellt der Schritt von der Idee, also einer erdachten Form, zur Herstellung eines physischen Produktes mit eben jener Form eine wesentliche Hürde dar. Die Herausforderung ist es, eine digitale Beschreibung der gewünschten Form zu erstellen und in ein für die produzierende Maschine verständliches Format umzuwandeln.

Hierzu können einerseits Digitalisierungswerkzeuge wie z.b. 3D-Scanner oder Messarme, andererseits CAD- und CAM-Software (computer aided design / computer aided manufacturing) eingesetzt werden. So kommt es bei der Gestaltung zwangsläufig zu Kompromissen, bedingt durch die Fähigkeiten des Benutzers, die eingesetzten Digitalisierungs-Werkzeuge und die Grenzen der eingesetzten Software.

Die vorliegende Arbeit beschäftigt sich mit der Frage, wie ein Verfahren zur Herstellung von dreidimensionalen Formen und Elementen aussehen könnte, das sowohl in der Architektur, als auch in der Kunst eingesetzt werden kann. Dabei spielen Faktoren wie Größe, Haltbarkeit, ökologische Nachhaltigkeit des Prozesses und nicht zuletzt Gestaltungsfreiheit eine wesentliche Rolle. Einen weiteren Schwerpunkt bildet die Benutzerfreundlichkeit: Ähnlich wie der







analoge gestalterische Prozess, soll auch die digitale Weiterbearbeitung ohne lange Einarbeitung in komplexe Computerprogramme möglich sein. So soll gewährleistet werden, dass durch intuitives Experimentieren das Potenzial des Verfahrens ausgeschöpft werden kann. Das Verfahren soll für ein möglichst breites Publikum zugänglich und benutzbar sein, gleichzeitig aber auch für komplexere Anwendung in der Architektur und Kunst eingesetzt werden können.

Die Ergebnisse dieser Fragen dienen als Grundlage für einen neuen Prozess der vom Beginn des Entwurfs bis zum fertigen Objekt reicht und im Zuge dieser Diplomarbeit zur Umsetzung kommt. Den technischen Schwerpunkt der Arbeit bildet die Entwicklung eines 3D-Scanners, der Steuerungs- und Interaktionssoftware sowie eines passenden Gussverfahrens.

1 Siehe dazu die Ausstellung MUSEUM Eduard Spörri trifft... Ruth Maria Obrist „Schichtungen“ 16.05. -13-12-2015, 5430 Wettingen. http://www.eduardspoerri.ch/, Zugriff am 28.05.2015.

2 Siehe dazu: Gramazio Kohler Research, ETH Zürich, 2006. http://www.gramaziokohler.arch.ethz.ch/web/d/lehre/81.html, Zugriff am 28.05.2015-

Abb. 01 CLIP 3D Printing Technology, Carbon3D
Abb. 02 Medusa Performance Capture System, Disney Research
Abb. 03 The Programmed Wall, Gramazio Kohler Research ETH

II RECHERCHE

Der Arbeit liegt eine ausführliche Recherche zu den Themen Rapid Prototyping, Rapid Tooling und Rapid Manufacturing sowie zu Digitalisierung und Scannverfahren zugrunde. Durch die Analyse und Evaluierung aktueller Technologien konnten geeignete Prozesse bereits bestehender Verfahren identifiziert und für den Topokopierer durch eigene ergänzt beziehungsweise angepasst werden. Auf den folgenden Seiten werden die unterschiedlichen Verfahren kurz besprochen, deren Potenzial und Einschränkungen erläutert und exemplarisch bereits bestehende Maschienen vorgestellt.

1 TECHNOLOGIEN

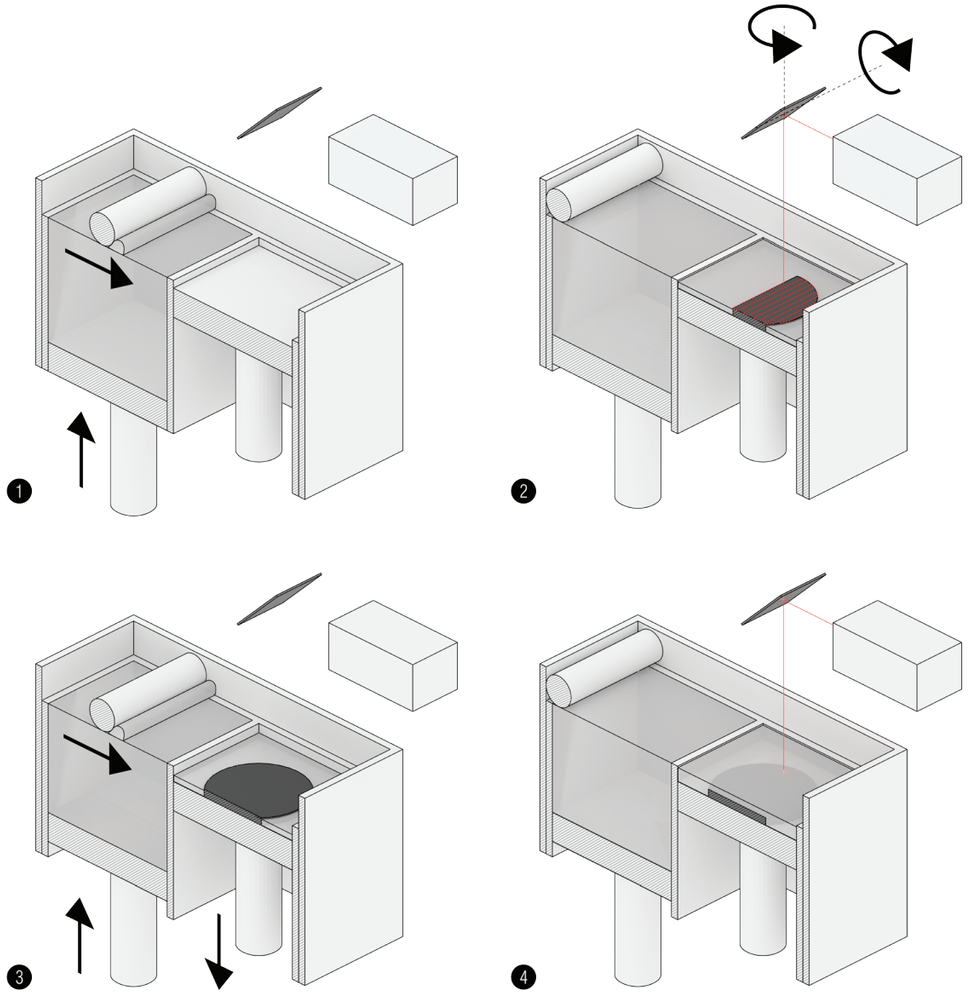
Zunächst war es notwendig relevante Ausgabe-Verfahren zu analysieren, um die Möglichkeiten auszuloten und den Scann-Prozess an die Anforderungen anzupassen.

1.01 RAPID PROTOTYPING | TOOLING | MANUFACTURING VERFAHREN

Bei den Rapid Prototyping/Tooling/Manufacturing Verfahren wird zwischen additiven und subtraktiven Herstellungsmethoden unterschieden. Wie der Name schon sagt, werden bei additiven Verfahren Materialien aufgebaut, während bei den subtraktiven Verfahren Substanz abgetragen wird.

Additive Verfahren
Der Großteil der additiven Rapid-Manufacturing Verfahren kann drei Hauptgruppen zugeordnet werden:

1. Verfahren mit pulverförmigem Grundmaterial, das mit Bindemittel oder durch Sintern selektiv verbunden wird.
2. Verfahren mit flüssigem Baumaterial, das durch Düsen aufgetragen wird und rasch erhärtet.
3. Belichtungsverfahren bei denen das Baumaterial in flüssiger Form vorliegt und durch gezieltes Belichten selektiv erhärtet.



1. Verfahren mit pulverförmigem Grundmaterial das mit Bindemittel oder durch Sintern selektiv verbunden wird.
Alle Verfahren, die dieser Kategorie angehören, basieren auf dem Prinzip des schichtweisen Verbindens eines pulverförmigen Grundmaterials im Bauraum der jeweiligen Maschine. Die Schichtdicke entspricht dabei der Auflösung des 3D-Drucks in Z Richtung.

Bei Verfahren mit Bindemitteln wird mit einem Druckkopf das Bindemittel an jenen Stellen auf dem Pulver aufgetragen, an denen das zu druckende Objekt die Schicht durchdringt. Bei Selektiven-Laser-Sinter-Maschinen wird die Pulverschicht mittels Laser selektiv erhitzt, wobei die einzelnen Körner an diesen Stellen verkleben. Bei Selektiven-Laser-Schmelz-Verfahren werden die Körner sogar komplett zum Schmelzen gebracht.

Ist eine Schicht fertig verbunden, wird eine neue Pulverschicht aufgebracht und das Pulver nach dem gleichen Prinzip wie zuvor bearbeitet. Die behandelten Körnchen verbinden sich nicht nur innerhalb ihrer Schicht, sondern auch mit der Schicht darunter. So wird Lage für Lage das Objekt hergestellt. Nachdem die letzte Schicht eingebracht und bearbeitet ist, muss das Objekt vom ungenutzten Pulver, in das es eingebettet ist, befreit werden. Hierbei ist zu beachten, dass alle Hohlräume des Objekts Öffnungen benötigen, um eingeschlossenes, ungenutztes Pulver wieder entfernen zu können.

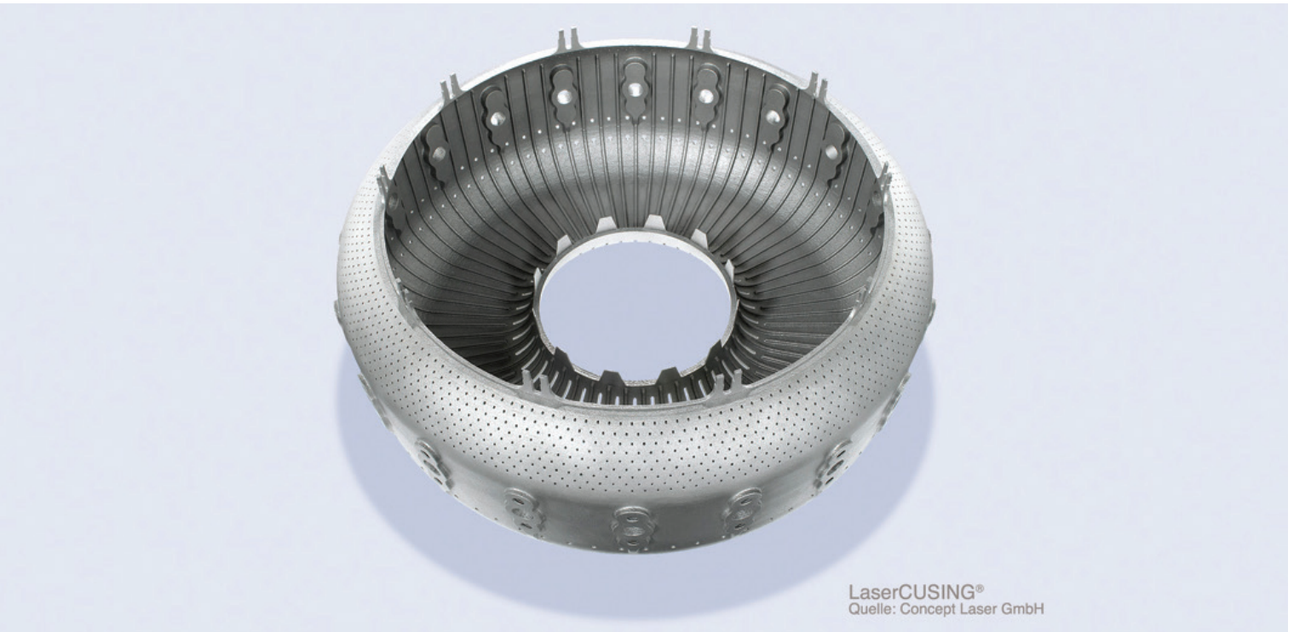
Bei einigen dieser Kategorie angehörenden Verfahren sind gedruckte Objekte, wenn sie frisch aus dem Drucker kommen, noch sehr zerbrechlich. Sie müssen erst infiltriert oder stark erhitzt werden, um ihre Endfestigkeit zu erlangen.

Abb. 04 Systemdarstellung Selektives Lasersintern

Bei Verfahren mit Bindemittel ist es zusätzlich möglich, durch Verwendung mehrerer farbiger Bindemittel Farbdrucke zu erstellen. Die feinste erreichbare Auflösung des Drucks hängt von der Körnung des Pulvers ab.

Beim Selektiven-Laser-Schmelzen können mit geeignetem Baumaterial besonders haltbare und mechanisch belastbare Teile hergestellt werden. Formen, die mit auf Pulver mit Bindemittel basierenden Verfahren bearbeitet werden, lassen sich auch in einem größeren Maßstab realisieren, wie die Firma „voxeljet“ beweist.

Besondere Vertreter dieser Gruppe: (Stand April 2015)
_ Voxeljet - VX4000
_ Concept Laser - X Line 2000r



Voxeljet:

Verfahren:	Pulver mit Bindemittel
Maschine:	VX4000
Maximaler	Bauraum: 4 x 2 x 1m
Druckkopf	Auflösung: 600 dpi
Schichtdicke:	Kunststoff (PMMA) 0.08 - 0.2 mm
	Quarzsand 0,2 - 0,3 mm
Geschwindigkeit:	75 Sekunden/Schicht, ca. 70 Stunden für den kompletten Bauraum bei 0,3 mm Schichtdicke
Haltbarkeit:	abhängig vom verwendeten Pulver und Bindemitte

Concept Laser:

Verfahren:	Verwendbare Materialien:
Maschine:	Selektives Laser Schmelzen
Maximaler Bauraum:	X Line 2000R
Laser Focus Durchmesser:	0.8 x 0.4 x 0.5 m
Schichtdicke:	0.1 - 0.5 mm
Geschwindigkeit:	0.03 - 0.2mm
Haltbarkeit:	materialabhängig 10 -100 cm^3 pro Stunde

Produzierte Teile sind mechanisch so belastbar, wie vergleichbare Teile aus dem selben Material

Abb. 05 Anwendungsbeispiel Voxeljet
Abb. 06 Anwendungsbeispiel Concept Laser

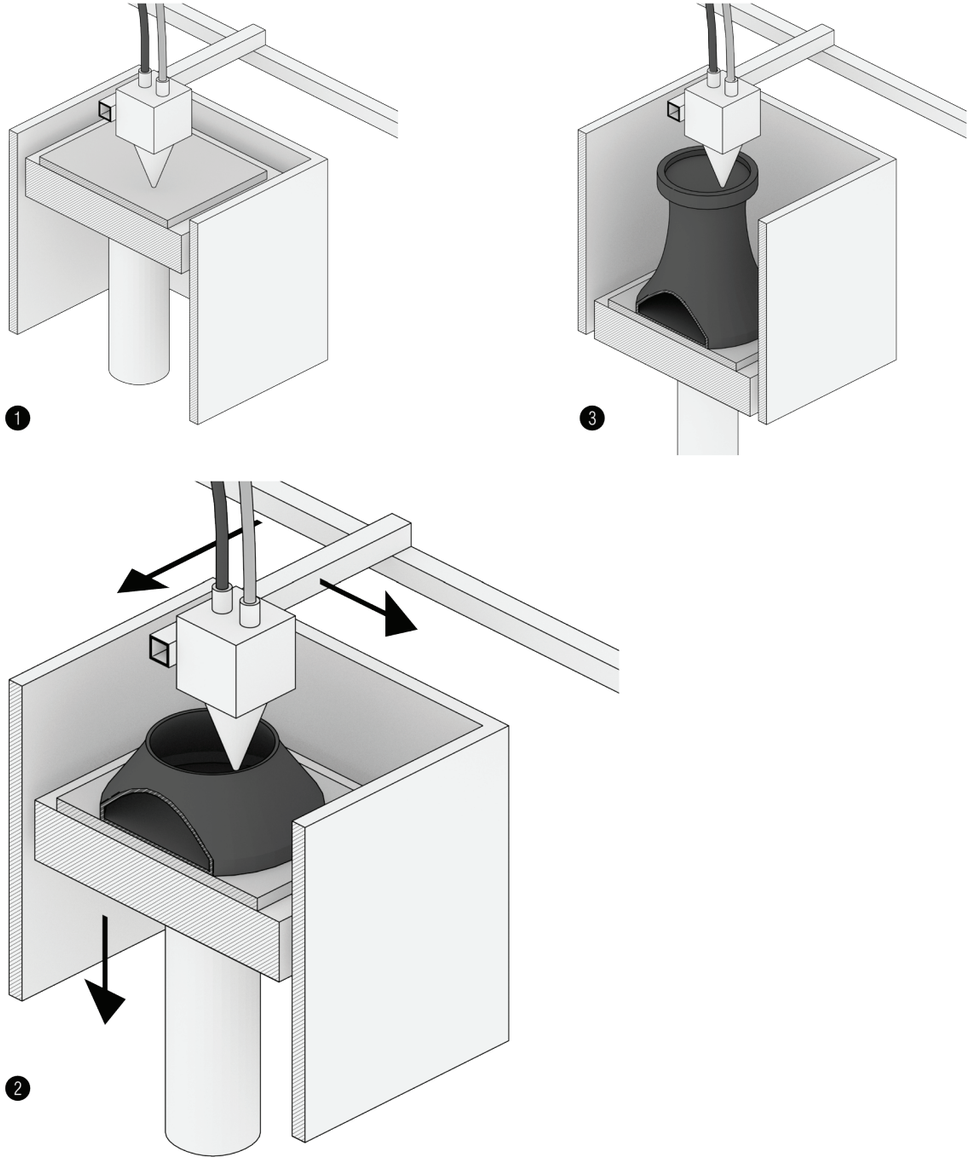


Abb. 07 Systemdarstellung FDM

2. Verfahren mit flüssigem Baumaterial, das durch Düsen aufgetragen wird und rasch erhärtet.

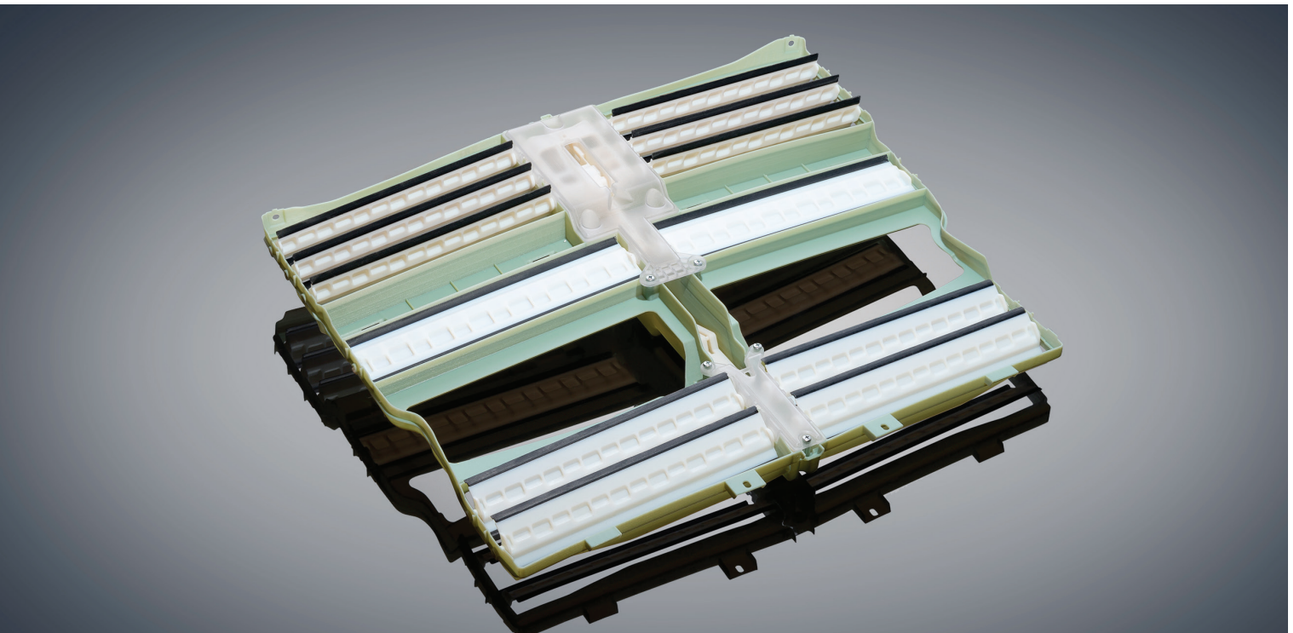
Auch bei diesen Verfahren wird das Objekt schichtweise aufgebaut. Allerdings kann bei jeder Schicht nur auf das Material der darunter liegenden Schicht aufgebaut werden. Auskragungen sind bei einem Druck mit einem Material nur begrenzt möglich. Um dieses Problem zu lösen, wird ein zweites Material als Stützstruktur mitgedruckt. Die Stützstruktur muss jedoch im Nachhinein mechanisch oder chemisch entfernt werden. Als Materialien kommen hauptsächlich thermoplastische Kunststoffe zum Einsatz. Diese werden durch Erhitzen in einen flüssigen Zustand gebracht und kühlen nach dem Verlassen der Düse wieder ab und erhärten dabei.

Andere Verfahren dieser Kategorie nutzen punktuelle UV-Strahlung oder chemische Reaktionen, um das Erhärten des Baumaterials gezielt zu erreichen. Mittlerweile gibt es eine Vielzahl an Materialien, mit denen auf diese Art gedruckt wird. Auch die Herstellung von Maschinen mit größeren Bauräumen ist möglich. Beim Skalieren des Bauraums wird meistens auch die Schichtdicke erhöht, um so die Druckgeschwindigkeit zu steigern. Sollen gedruckte Teile mechanisch belastet werden, ist zu beachten, dass die gedruckten Schichten in sich stabiler sind als untereinander.

Besondere Vertreter dieser Gruppe: (Stand April 2015)

_ Stratasys - Fortus 900mc

_ Stratasys - Objet 1000 Plus

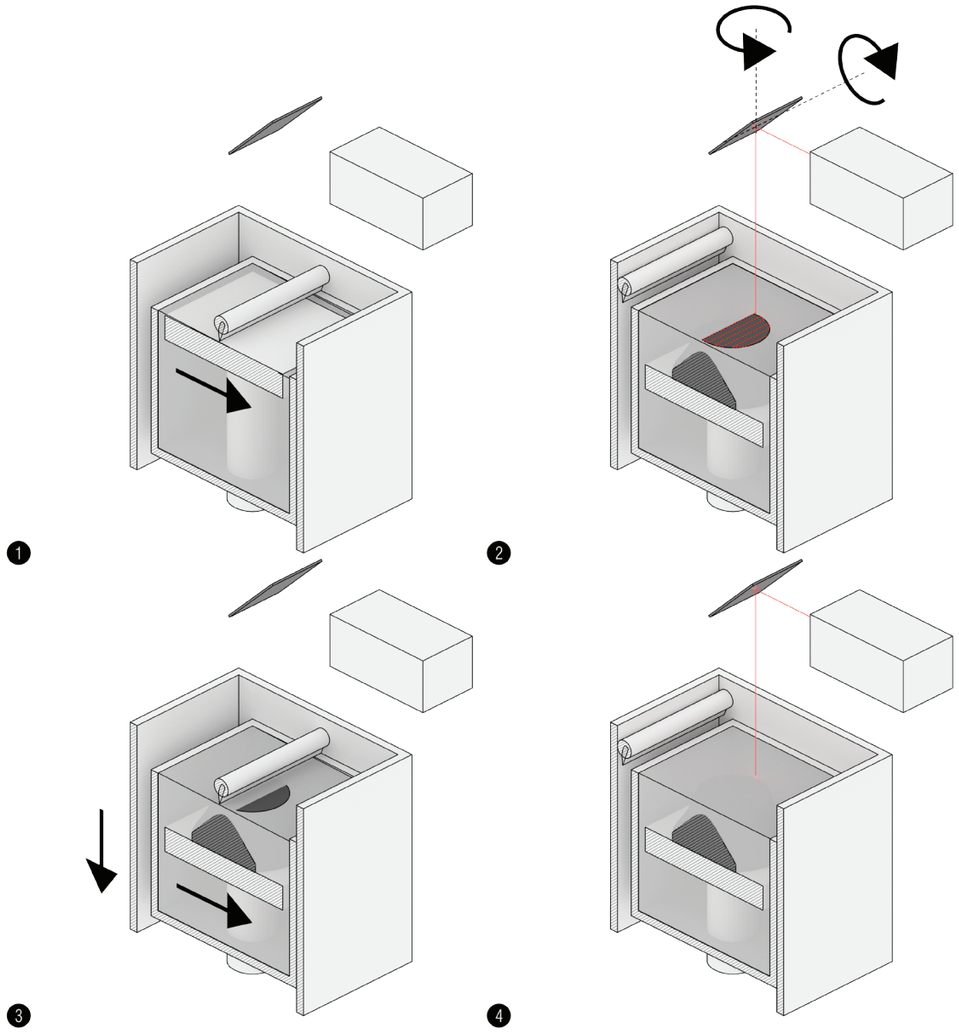


Stratasys:

Verfahren:	FDM (Fused Deposition Modeling)
Maschine:	Fortus 900mc
Maximaler Bauraum:	0.91 x 0.61 x 0.91 m
Schichtdicke:	0.13 - 0.33 mm
Haltbarkeit:	die produzierten Teile sind mechanisch so belastbar, wie vergleichbare Teile aus dem selben Material, nur begrenzt witterungs- und lichtbeständig
Verwendbare Materialien:	ABS, PC, Nylon

Verfahren:	PolyJet (Mischen mehrerer Materialien möglich, wodurch Materialeigenschaften,wie Farbe und Elastizität kombiniert werden können.)
Maschine:	Objet 1000 Plus
Maximaler Bauraum:	1 x 0.8 x 0.5 m
Schichtdicke:	ab 0.016 mm
Druckkopfauflösung:	300 dpi
Haltbarkeit:	Vom verwendeten Material abhängig, empfindlich auf UV Strahlen (Teile werden spröde)

Abb. 08 Anwendungsbeispiel FDM
Abb. 09 Anwendungsbeispiel PolyJet



3. Belichtungsverfahren bei denen das Baumaterial in flüssiger Form vorliegt und durch gezieltes belichten selektiv erhärtet

Für diese Verfahren werden spezielle Flüssigkeiten benötigt, die unter Einfluss von UV Belichtung erhärten. Die Bauplattform wird zunächst mit einer Schicht der Flüssigkeit bedeckt und in den Bereichen der jeweiligen Objektschicht, entweder mit einem UV-Laser abgefahren oder mittels eines UV-Projektors belichtet. Es folgt ein Absenken der Bauplattform und das Benetzen der bereits gedruckten Schicht mit weiterer Flüssigkeit.

Bei anderen Geräten erfolgt die Belichtung von unten. Die Bauplattform wird nach jeder Schicht angehoben, damit kontinuierlich Flüssigkeit nachrinnen kann. Diese Methode hat den Vorteil, dass weniger zusätzliche Flüssigkeit für den Druck benötigt wird. Das zu druckende Teil muss nicht komplett in der Flüssigkeit versenkt werden, sondern scheint aus der Flüssigkeit herauszuwachsen. Beim Belichten der nächsten Schicht verbindet sich diese mit der vorherigen Schicht. Der Prozess wird so oft wiederholt, bis das komplette Objekt aufgebaut wurde.

Auch bei diesem Verfahren ist das Drucken einer Stützstruktur für Auskragungen notwendig. Die Stützstruktur besteht dabei aus dem selben Material, wie das gedruckte Objekt und muss später mechanisch entfernt werden.

Nachdem der Druckprozess beendet ist, wird das Druckstück chemisch gereinigt, um Flüssigkeitsrückstände zu entfernen. Je nach verwendetem Material kann es notwendig sein, das Teil in einem UV Ofen fertig aushärten zu lassen.

Abb. 10 Systemdarstellung Stereolithographiev

Hauptvorteil dieser Verfahren ist die rasche Produktionsgeschwindigkeit und die Herstellung sehr feiner Schichtdicken, wodurch größte Präzision möglich ist. Als Nachteile sind hohe Kosten für das flüssige Baumaterial und die Empfindlichkeit gegenüber UV-Strahlung zu nennen.

Besondere Vertreter dieser Gruppe: (Stand April 2015)

_ Formlabs - Form 1 +
_ 3DSYSTEMS - ProX950

Formlabs:

Verfahren:	Stereolithographie
Maschine:	Form 1+
Maximaler Bauraum:	0.125 x 0.125 x 0.165 m
Genauigkeit (Laserposition):	0.01 mm
Feinste druckbare Strukturen:	0.3 mm
Schichtdicke:	0.025 - 0.2mm
Geschwindigkeit:	abhängig von der Schichtdicke (1-3 cm pro Stunde bei einer Schichtdicke von 0.1mm)
Haltbarkeit:	Produzierte Teile sind mechanisch so belastbar, wie vergleichbare Teile aus dem selben Material
Verwendbare Materialien:	Photopolymerharz in Schwarz, Grau, Transparent und Weiß, rückstandslos ausbrennend für Gussteile oder elastisch

3DSYSTEMS:

Verfahren:	Stereolithographie
Maschine:	ProX 950
Maximaler Bauraum:	1.5 x 0.75 x 0.55 m
Auflösung Laser Brennpunkt:	0.0013 mm
Schichtdicke:	0.05 - 0.15 mm
Geschwindigkeit:	abhängig von der Schichtdicke und dem Material

Abb. 11 Anwendungsbeispiel Form1 +
Abb. 12 Anwendungsbeispiel ProX 950

Subtraktive-Verfahren

Bei subtraktiven Verfahren wird die gewünschte Form mit Werkzeugen aus einem Rohling herausgearbeitet. Der Rohling muss dabei etwas größer als das fertige Teil sein. Meistens kommen hierfür CNC-Fräsen oder Drehbänke zum Einsatz. Das Spektrum der auf diese Weise bearbeitbaren Materialien deckt den Großteil gängiger Werkstoffe ab. Leitende Metalle können zusätzlich auch durch Elektroerodieren bearbeitet werden.

Was die Komplexität der erreichbaren Geometrien angeht, sind die additiven 3D-Druckverfahren im Vorteil, da sie durch das schichtweise Aufbauen jeden Punkt der gewünschten Form problemlos erreichen können. Bei zerspanenden Verfahren hängen die Grenzen der herstellbaren Geometrien davon ab, ob alle Bereiche der gewünschten Form mit geeigneten Werkzeugen erreichbar sind. Ist dies nicht der Fall, muss das Objekt in mehreren einzelnen Segmenten gefertigt und zusammengefügt werden.

Beim Elektroerodieren wird das Metallstück in eine nicht leitenden Flüssigkeit getaucht. Danach wird ein Werkzeug mit einer, dem Werkstück gegenüber negativen elektronischen Spannung, angenähert. An der Stelle an der sich Werkzeug und Werkstück beinahe berühren, kommt es zu Entladungen, die am Werkstück Material abtragen. Da keine mechanischen Belastungen auf das Werkzeug einwirken, können mit diesem Verfahren sehr feine und tiefe Strukturen mit hervorragender Oberflächenbeschaffenheit herausgearbeitet werden. Nachteile sind unter anderem, dass nur elektrisch leitende Materialien bearbeitet werden können und, dass das Abtragen von Material viel Zeit in Anspruch nimmt.

Gussverfahren

Oftmals ist es einfacher, ein Objekt zuerst aus einem einfach zu bearbeitenden Material herzustellen und damit eine Gussform zu erstellen. So kann ein Objekt aus einem weniger haltbaren Material durch einen stabileren, gießbaren Werkstoff ersetzt werden. Ein 3D-Druck, der aus einem rückstandslos verbrennenden Material hergestellt wurde, kann beispielsweise zur Herstellung einer Form für den Metallguss herangezogen werden. Andere Möglichkeiten sind das direkte Herstellen von Negativformen mittels zuvor erwähnter additiver- oder subtraktiver- Verfahren.

Auch zur Herstellung komplexerer Elemente in der Architektur werden Gussformen häufig eingesetzt. Sollen außergewöhnliche Formen aus Beton gegossen werden, ist die Herstellung passender Schalungen notwendig. Da es sich oftmals um Einzelanfertigungen handelt, werden die Schalungen händisch hergestellt oder bei aufwendigeren Formen gefräst. Nicht nur der Zeitaufwand für die Herstellung der Form und des Gusses, sondern auch der Materialaufwand für aufwendigere Bauwerke ist dabei enorm.

Kurzzusammenfassung der Herstellungsverfahren

3D-Druckverfahren ermöglichen die Herstellung beinahe jeder Form. Das Spektrum der verwendbaren Materialien und deren Eigenschaften ist sehr vielfältig.

Hauptproblem ist, dass die Kosten für eine Anwendung im großen Maßstab bei der Verwendung haltbarer Materialien (noch) sehr hoch sind. Günstigere Materialien sind aufgrund ihrer geringeren Witterungsbeständigkeit für eine dauerhafte Verwendung in der Architektur meist ungeeignet.

Werden weniger kostspielige Verfahren und Materialien zur Herstellung von Gussformen eingesetzt, um beispielsweise Betonelemente zu produzieren, stellt sich die Frage nach der Lagerung oder Entsorgung der Form nach dem Guss.

Auch mit CNC-Fräsen können Betongussformen relativ schnell und preiswert hergestellt werden. Allerdings muss hierbei nicht nur die Einlagerung oder Entsorgung der Form beachtet werden, sondern auch der Müll der während der Produktion der Form abfällt.

Einer der Schwerpunkte der vorliegenden Arbeit war es, die positiven Eigenschaften der jeweiligen Verfahren herauszufiltern und darauf aufbauend einen neuen Prozess zu entwickeln. So soll ein neuer Lösungsansatz für bestehende Probleme gezeigt werden.

Trotz der vielfältigen Möglichkeiten, die 3D-Druckverfahren mittlerweile bieten, erschien die Verwendung eines Gussverfahrens zur Herstellung von Objekten für den architektonischen Bereich am vorteilhaftesten.

Schlussfolgerung

Um negative Aspekte des Gussformherstellens mittels CNC-Fräse zu mindern, wurde von mir ein eigenes Verfahren entwickelt. Es basiert auf dem Prinzip des Sandgusses. Im Gegensatz zu gängigen Verfahren wird jedoch nicht ein Modell abgeformt, sondern ein digitaler Entwurf mit einem Werkzeug aus Formsand geschnitzt.

Das CNC geführte Schnitzwerkzeug saugt abgetragenen Sand bereits während der Bearbeitung ein und fängt ihn in einem Behälter auf. Auf diese Weise kann sehr sauber gearbeitet und der gesamte Gusssand für die nächste Form wiederverwendet werden. Die Geschwindigkeit mit der Sandformen mit diesem Verfahren hergestellt werden können übertrifft 3D-Druckverfahren, dafür wird momentan noch ein Kompromiss was die Komplexität der Form angeht eingegangen.

1.02 DIGITALISIERUNGS / 3D-SCANN VERFAHREN

Es gibt zahlreiche unterschiedliche Möglichkeiten, die Oberflächenbeschaffenheit eines Objektes digital zu erfassen. Grob können die Verfahren in berührungslose und abtastende Methoden eingeteilt werden. Im Folgenden sollen die wichtigsten Verfahren kurz besprochen werden.

Berührungslose Verfahren

Bei der Triangulation wird mit Hilfe von Geometrie die Position eines Punktes ermittelt. Dabei ist der zu messende Punkt ein Punkt eines Dreiecks, bei dem die Position der anderen beiden Punkte, die an der Messung beteiligt sind, in Relation zum Messsystem bekannt sind. Durch Ermittelung der Winkel des Dreiecks kann die Position des zu messenden Punktes berechnet werden.

Allgemein wird zwischen passiver und aktiver Triangulation unterschieden.

Bei passiver Triangulation wird ein Punkt von zwei bekannten Standorten aus aufgenommen. Durch Ermittlung der Sehwinkel wird die Position des Punktes berechnet. Schwierigkeit dieses Verfahrens ist, dass der Punkt von beiden Standorten aus sichtbar

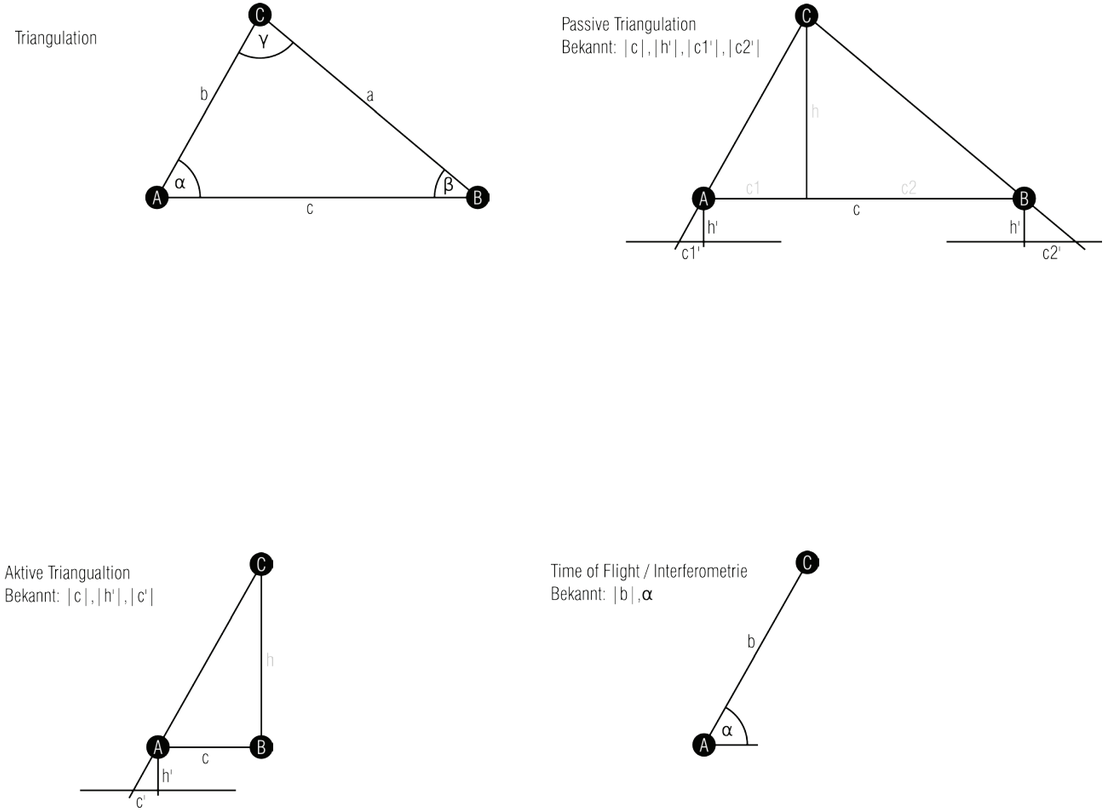


Abb. 13 Grundlage Triangulationsverfahren

und identifizierbar sein muss. Dies setzt kontrastreiche Bilder voraus und eignet sich nur für markante Punkte. Auf die Dichte und Verteilung der erfassten Punkte kann nur schwer Einfluss genommen werden.

Bei aktiver Triangulation wird mit einer Lichtquelle auf das zu vermessende Objekt beispielsweise ein Punkt projiziert. Solange die Ausrichtung und Position der Projektionsquelle im Verhältnis zur Kamera und deren optische Eingenschaften bekannt sind, kann über den Sehwinkel der Punkt ermittelt werden. Schwierig wird das Erfassen von Punkten, wenn keine freie Sicht auf die Projektion vom Kamerastandpunkt aus möglich ist oder die Oberfläche des zu erfassenden Objekts zu viel Licht schluckt bzw. zu stark reflektiert. Durch synchrones Aufnehmen der Projektion von unterschiedlichen Positionen aus, kann das Problem der verdeckten Sicht abgemindert werden. Eine Alternative zur Verwendung mehrerer Kameras ist eine Kamera deren Position gezielt verändert wird.

Als Alternative zu Triangulationsverfahren kann auch per Interferometrie oder mit Time-of-Flight-Verfahren die Position einzelner Punkte erfasst werden. Dazu wird eine Welle auf den zu messenden Punkt geschickt.

Bei der Interferometire werden zwei mit gleicher Frequenz allerdings unterschiedlicher Phase schwingende Wellen ausgesandt. Eine Welle wird auf den Punkt der eingemessen werden soll gerichtet, dort reflektiert und beim Zurückkommen vom Detektor erfasst. Die andere wird auf einen Spiegel mit bekannter Entfernung umgelenkt und kommt von dort zurück zum Detektor. Über die vom Detektor gemessene Interferenz kann die Entfernung berechnet werden. Dies funktioniert allerdings nur bis zu einer Entfernung von der Hälfte der Wellenlänge da sich die Interferenzen danach wiederholen.

Beim Time-of-Flight Verfahren wird die Zeitdifferenz zwischen dem Aussenden der Welle und dem Erfassen der Reflexion gemessen. Da die Ausbreitungsgeschwindigkeit der Welle bekannt ist, kann die Entfernung errechnet werden.

Für eine Bestimmung der räumlichen Lage des Punktes muss bei beiden Verfahren die Richtung, in der die Welle ausgesandt wurde, bekannt sein.

Der Vorteil dieser Verfahren gegenüber den Triangulationsverfahren ist, dass Emitter und Detektor kollinear angeordnet werden können und das Problem der verstellten Sicht entfällt.

3D Scannverfahren Basierend auf Triangulation

Bei der Photogrammetrie wird ein Objekt nach dem Prinzip der passiven Triangulation dreidimensional rekonstruiert. Es werden dazu von unterschiedlichen Standpunkten aufgenommene Bilder analysiert. Werden dazu Einzelbilder eines Videos verwendet, bei dem entweder das Objekt oder die Kamera bewegt wird, bezeichnet man das Verfahren „Structure from Motion“ Dies wird in der Filmindustrie angewendet, um im Nachhinein Filme dreidimensional erscheinen zu lassen und um Spezialeffekte realistischer in Filme zu integrieren. Es treten die gleichen Schwierigkeiten und Probleme wie bei der passiven Triangulation auf. (autodesk123d, smartphone Apps, Medusa Performance Capture System)

Structured Light Verfahren nutzen das Prinzip der aktiven Triangulation. Bei ihnen wird ein zuvor definiertes Muster auf die zu scannende Oberfläche projiziert. Von einer Kamera, deren Position im Verhältnis zum Projektor bekannt ist, wird das von der Oberfläche des Objekts deformierte Muster aufgenommen. Durch Analyse der Deformation des Musters kann die Oberfläche des Objekts digital rekonstruiert werden.

Eine der einfachsten Varianten dieses Verfahrens nutzt einen Linienlaser als Projektionsquelle. Da so allerdings nur eine Konturlinie pro aufgenommenem Bild rekonstruiert werden kann, muss die Position von Projektionsquelle im Verhältnis zum Objekt verändert werden und die Art der Positionsveränderung muss bekannt sein.

Bei der Projektion aufwendigerer Muster können Oberflächen auch ohne Positionsänderung erfasst werden.
(Kinect, David-3D, Faro ScanArm, Nikon ModelMaker MMCx, mein Scanner)

Auf Fokus bzw. Unschärfe basierende Verfahren nutzen über eine Kamera bekannte Parameter. Durch Verändern des Fokus Bereichs und Beobachten der Zunahme von Schärfe bzw. Unschärfe in Verbindung mit der Richtung der Sehstrahlen, kann die Lage von Punkten ermittelt werden. Ähnlich der Photogrametrie funktioniert dieses Verfahren nur für kontrastreiche Aufnahmen und gemusterte Oberflächen.
(Lytro light field Camera)

Auf Interferometrie oder dem Time-of-Flight Prinzip basierende 3d-Scanner digitalisieren Oberflächen in dem Punkt für Punkt, zeilenweise Oberflächen eingemessen werden. Der Strahl mit dem gemessen wird, wird dazu über Spiegel ausgerichtet.
(Zoller+Fröhlich Z+F IMAGER, RIEGL VZ-2000)

Abtastende Verfahren
Mittels maschinell oder händisch geführter Messtaster können ebenfalls Oberflächengeometrien ermittelt werden. Dazu muss der Messtaster an die Oberfläche herangeführt werden, bis er diese berührt. Mitunter kann auf diese Weise die Lage von Punkten genauer ermittelt werden als mit optischen Verfahren, da keine Reflexionen oder transparente Bereiche die Messung stören. Wenn für die Erfassung der Geometrie einer Oberfläche viele Punkte notwendig sind, ist das Abtasten jedoch wesentlich zeitaufwändiger als berührungsloses Einmessen.
(Renishaw OMP400)

Schlussfolgerung
Mit abtastenden Verfahren benötigt die Digitalisierung einer Oberfläche zu lange, um für ein interaktives Arbeiten mit Oberflächen in Frage zu kommen. Interferometrie und Time-of-Flight basierende Scannverfahren könnten die Anforderungen an Präzision und Geschwindigkeit erfüllen, sind allerdings für dieses Diplomprojekt zu teuer in der Anschaffung und zu aufwändig selbst herzustellen. Mit auf Photogrametrie und Focus basierenden Verfahren ist eine hohe Dichte und gleichmäßige Verteilung eingemessener Punkte eine Herausforderung. Ein auf Structured-Light Verfahren basierender 3D-Scanner zeichnet sich durch die damit erreichbare Auflösung, Geschwindigkeit, und den relativ einfachen Aufbau aus.

Eine der größten Hürden bei der Nutzung eines 3D-Scanners ist, dass das Ergebnis des Scanns im Normalfall eine sogenannte 3D-Punktewolke ist. Auch wenn die Punktewolke von der Scannsoftware in ein Mesh mit reduzierter Punkteanzahl umgewandelt wird, ist für das Weiterarbeiten mit den erfassten Daten die Benutzung einer 3d Modellier-Software unumgänglich.

1.03 SCHLUSSFOLGERUNGEN TECHNOLOGIEN

Wie könnte nun also ein Prozess aussehen der 3D-Scann, digitale Nachbearbeitung und CNC-Fertigung verbindet, mit Focus auf benutzerfreundlicher Bedienbarkeit? Ein Prozess der die gesuchten Anforderungen erfüllt, mit dem Unterschied, dass er nur für den zweidimensionalen Bereich ausgelegt ist, ist das Fotokopieren.

Der naheliegende Lösungsansatz wäre also der Versuch, einen 3D Kopierer als Kombination von 3D-Scanner und -Drucker herzustellen. Nachteil dieser Lösung ist hauptsächlich, dass die digitale Nachbearbeitung und Modifikation nur über 3D-Modellier-Software für Menschen mit fortgeschrittenen 3D-Modellier-Fähigkeiten nutzbar wäre, denen das herkömmliche arbeiten mit 3D-Scanner und 3D-Drucker ohnehin zuzutrauen ist.

Mein Vorschlag für einen Lösungsansatz ist, sich als einen ersten Schritt in die dritte Dimension mit Reliefs auseinander zu setzen. Auf diese Weise verliert man zwar Hinterschneidungen, dafür kann mit einfachen Mitteln viel erreicht werden. Ein Relief kann als Graustufenbild dargestellt und in Folge auch mit Bildbearbeitungsprogrammen bearbeitet werden. Für die Herstellung einer Gussform für ein Relief reicht eine drei-achsige Bearbeitung aus. Somit qualifizieren sich mehrere Maschinen für das Produzieren der Form. Zusätzlich lässt sich ein 3D-Scannverfahren zur raschen Erfassung von Reliefs verhältnismäßig einfach umsetzen.

Angestrebt wird also ein Prozess, der in der Bedienung dem Prozess des Fotokopierens ähnelt, da dieser sich bereits bewährt hat, wie im nächsten Kapitel genauer beschrieben.

2 DAS FOTOKOPIEREN ALS VORBILD FÜR EINEN INTEGRIERTEN PROZESS

Die Fähigkeiten eines Kopierers sind nicht nur auf das Vervielfältigen von Dokumenten beschränkt, sondern werden durch das Verbinden von analoger Eingabe sowie der Möglichkeit zur direkten Manipulation und analoger Ausgabe erweitert.

Der Fotokopierer war Werkzeug einer eigenen Kunstrichtung, welche durch die Verfügbarkeit von Kopierern in normalen Haushalten endete. „Auf der Biennale in Venedig wurden 1970 erstmals Computergrafiken und Copy-Art-Arbeiten im Rahmen einer etablierten Kunstausstellung gezeigt.“(Copy Art, S.175) Nichts desto trotz ist der Fotokopierer immer noch ein wichtiges Instrument für viele Bereiche der Kunst – besonders in der Grafik. Durch fortlaufende Evolution der ursprünglichen Erfindung bietet das Fotokopieren mittlerweile besonders vielseitige Arbeitsmöglichkeiten. Skalierung, Kontrast, Collagen, Bewegung, Unschärfe durch Anheben, Farbanpassung, Verfremdung durch wiederholtes Kopieren oder die Ausgabe auf besondere Papiere sind nur ein paar der Möglichkeiten, sich an einem Kopierer kreativ zu betätigen und „Originale“ herzustellen. Das Anwendungsgebiet wird in Verbindung mit einem PC noch erweitert. Hierbei werden Scann- und Druckfunktion separat von einander genutzt.

Abb. 14 Angewandte Technik: Copy Motion

2.01 DIE TECHNIKEN DER COPY ART

„In der Copy Art werden die technischen Vorgaben und Möglichkeiten, mit denen die Konstrukteure ihre Kopierer ausgestattet haben, sowohl genutzt als auch transzendiert. Die Künstler überwinden mit ihrer Kreativität die meist starren, zweckorientierten Programme der Automaten oder lassen diese in einem veränderten Kontext ablaufen. So wundert es nicht, dass drei der vier grundlegenden Copy-Art-Techniken an genau diesen Punkten ansetzen, die Sicht der Konstrukteure als Schwachstelle ihrer Maschinen erscheinen.“ S.181

„Die Beherrschung aller künstlerischen Kopiertechniken ist nun zwar keine unabdingbare Voraussetzung dafür, ein perfektes Copy-Kunstwerk schaffen zu können – das zeigen die oft hervorragenden Bilder von Amateuren -, wohl aber für jede ernsthafte künstlerische Auseinandersetzung mit dem Medium. Ohne ein tiefes Verständnis der Prozesse, die der Fotokopie zugrundeliegen, und ihrer verschiedenen Verfahren läßt sich so manche gute Idee nämlich einfach nicht verwirklichen.“ S.180

Realkopie – Kopierer als Kamera

Die Vorgehensweise bei der Realkopie ist wohl die ursprünglichste aller Kopiertechniken. Objekte werden auf dem Vorlagenglas arrangiert, bei Bedarf mit einem weißen Tuch abgedeckt und die Copytaste gedrückt. Die Tiefenschärfe ist je nach Gerät auf 0.5 bis 5cm begrenzt. Dies verleiht Realkopien dreidimensionaler Objeke ein charakteristisches Erscheinungsbild. Wird das resultierende Bild vertikal betrachtet, wirken die auf dem horizontalen Vorlagenglas angeordneten Objekte fast schwerelos.

Abb. 15 Angewandte Technik: Copy Motion

Copy Motion – Kopie in Bewegung

Bei der Copy Motion wird die Eigenheit des Fotokopierprozesses genutzt, dass das Aufnehmen des Vorlagenbereichs entlang einer Zeitachse geschieht. Die Länge der Zeitachse hängt davon ab, wie lange die Maschine braucht, um den Vorlagenbereich Zeile für Zeile zu erfassen. Wird eine Vorlage während des Scannprozesses entgegen der Fahrtrichtung der Scannleiste bewegt, so erscheint die Vorlage im resultierenden Bild gestaucht. Bewegt man die Vorlage langsam in Fahrtrichtung mit, so erscheint die Vorlage gestreckt. Wird man jedoch Schneller als die Scannleiste, erhält man ein gespiegeltes Abbild der Vorlage. Durch diagonales Bewegen der Vorlage können Perspektiveneindrücke erzielt werden. Andere Effekte die auch die Zeitachse der Aufnahme nutzen und deshalb in den Bereich der Copy Motion gehören, sind das schnelle Auswechseln der Vorlage oder das Folgen der Scannleiste mit Fingern und anderen Gegenständen als eine Art Pinsel.

Overlay – Überlagerungstechnik

Sogenannte Overlays beruhen auf dem Prinzip, das Ergebnis eines Kopierdurchganges mit einem weiteren zu überlagern. Dafür wird ein bereits bedrucktes Blatt wieder in die Papierkassette bzw. das Papiereinzugsfach eingelegt. Bei einer Überlagerung nach diesem Verfahren spricht man von einer Copy-on-Copy.
Beim Kopieren mit einem (älteren) Gerät, welches die Grundfarben einzeln erfasst und dafür drei bis vier mal den Vorlagenbereich abtasten muss, kann zwischen den einzelnen Scan-Durchgängen die Bildvorlage getauscht werden. Dies ermöglicht die Kombination von bis zu vier Bildvorlagen. Durch den Wechsel der Reihenfolge der Bildvorlagen ergeben sich bis zu 16 Farbvarianten von einem Bild.

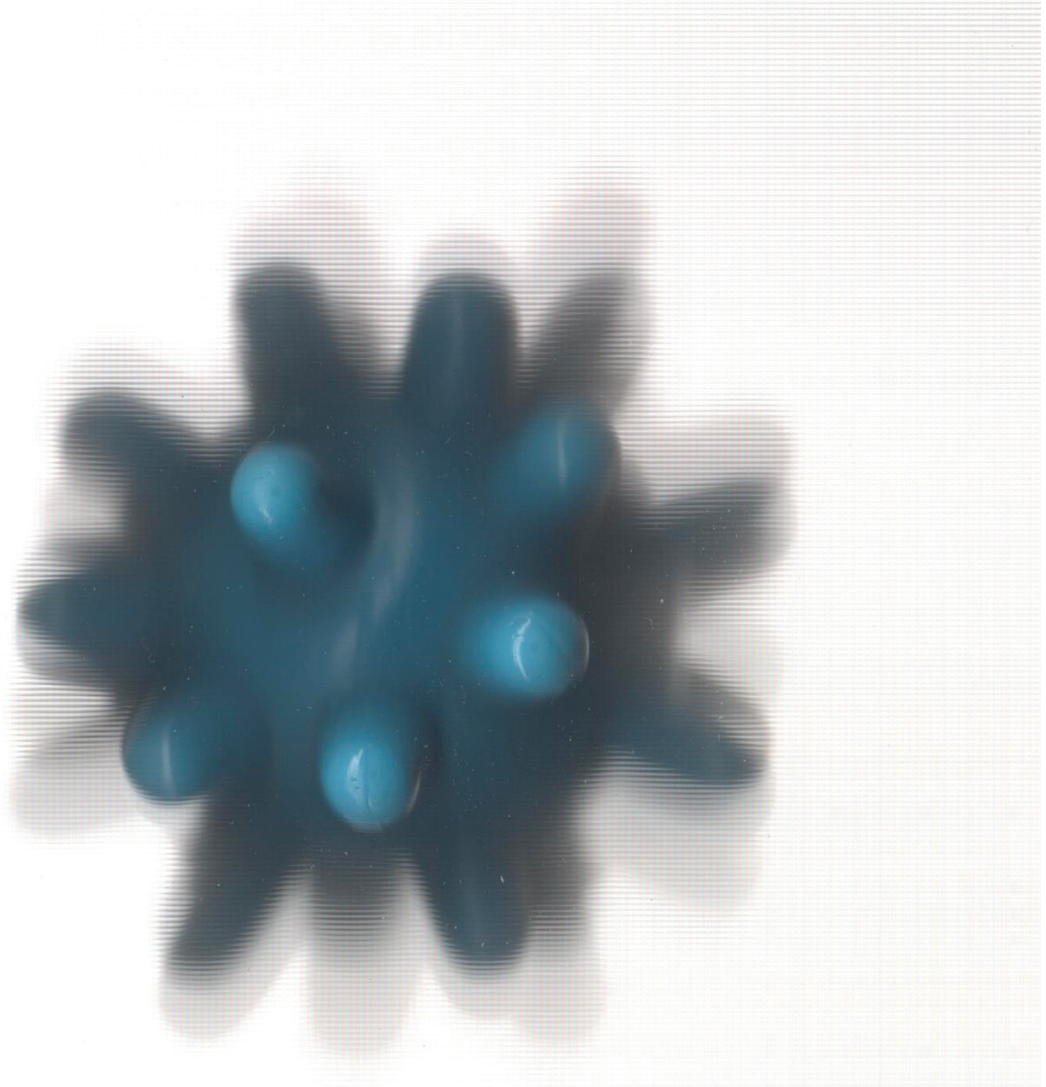


Abb. 16 Angewandte Technik: Realkopie Blow-Up

BlowUps – Vergrößerung

Digitale Kopierer ermöglichen das Vergrößern einer Vorlage auf das mehrfache der ursprünglichen Größe. Dabei wird das Ergebnis passgenau auf mehreren einzelnen Blättern ausgegeben. Die Fotokopie wird immer noch häufig in der Street-Art-Szene eingesetzt, als günstige und schnelle Möglichkleit zur Herstellung von Plakaten in allen Abmessungen.

Body Art

Die Body Art ist eine der Urformen der kreativen Betätigung am Kopierer. Es handelt sich dabei generell um Realkopien von Körperteilen oder Tieren. Besonders beliebte Motive sind dabei Hände, Köpfe und nicht zuletzt Gesäße. Für Ganzkörper Kopiegrafien sollte die Abdeckung des Vorlagenglases abmontiert und um den Kopierer ein Podest aufgebaut werden, um die benötigte Akrobatik zu erleichtern. Mit sieben bis acht Realkopien auf DIN-A3 im Querformat kann ein durchschnittlicher Mensch abgebildet werden. Body Art wurde auch gerne für künstlerische Aktionen und Performances angewandt.

Collage / Variage

Fotokopierer wurden von Mitgliedern der Mail-Art-Bewegung eingesetzt, um Collagen zu vervielfältigen. Wird die Belichtung richtig angepasst, kann ein Verschmelzen der Bildelemente erreicht werden. Durch gezielte Unterbelichtung wird der Zitatcharakter der einzelnen Elemente betont werden.
Bei der Variage werden die einzelnen Bildelemente nicht wie bei der Collage fix montiert, sondern direkt auf dem Vorlagenglas arrangiert. So kann mit Serien, Sequenzen und Bewegungen gearbeitet werden.

Hardcopy

Bei der Hardcopy setzen sich Künstler mit den Ausgabeverfahren auseinander und wie diese beeinflusst oder mit alternativen Quellen genutzt werden können. Werden für die Herstellung einer Hardcopy Transferverfahren genutzt, so eignen sich deren Zwischenprodukte für den Pirate-Art-Prozess.[1]

Installationen

Installationen sind meistens speziell für einen Ort gestaltete temporäre räumliche Arbeiten, oder entstehen im Rahmen künstlerischer Aktionen, bei denen es um den Werkprozess geht. Das Kopiergerät ist also ein wesentlicher Teil des Kunstwerks und wird in den Raum eingebunden. Das Papier lässt sich einfach mit Klebeband, Nadeln oder Nägeln befestigen. Für dauerhaftere Montage können auch Kleister, Buchbinderleim, Sprühkleber oder Klebefolien benutzt werden.

Negativtechnik

Schwarzweißkopierer besitzen üblicherweise eine Funktion zum Invertieren des Farbraumes. Bei Farbkopierern kann eine Funktion zur Umwandlung von Farbnegativen für die Herstellung von Dias integriert sein. Alle modernen Kopierer haben mittlerweile einen Netzwerkanschluss und können daher mit Hilfe eines Computers zur Invertierung von Vorlagen genutzt werden.

Portraits

Fast jeder der schon einmal mit einem Kopierer gearbeitet hat, hat (mehr oder weniger bewusst) eine Fotokopie seiner Hand erstellt. Bei einer spielerischen Auseinandersetzung mit der Maschine ist es nur eine Frage der Zeit bevor Köpfe und andere Körperteile auf dem Vorlagenglas landen. Entscheidend für das Aussehen von mit dem Kopierer aufgenommenen Portaits ist die begrenzte Tiefenschärfe. Alles, was sich direkt auf der Vorlagenscheibe befindet, wird präzise wiedergegeben. Mit zunehmender Entfernung wird alles verschwommener bis, es ganz im diffusen Nichts verschwindet.

Wird das Gesicht auf das Vorlagenglas gedrückt werden Ohren, Backen oder Nasen, dem entsprechend deutlich allerdings platt gedrückt wiedergegeben.

Auf den folgenden Seiten werden Beispiele der jeweiligen Techniken angeführt.

1 Unter Pirate-Art versteht man im Zusammenhang mit Copy Art Verfahren, bei denen technische Abfall-Produkte fremder Kopien zu einem neuen Kunstwerk verschmolzen werden. Siehe dazu: Urbons, Klaus: Copy Art, Kunst und Design mit dem Fotokopierer. Köln 1991, S. 128.

Abb. 17 Angewandte Technik: Realkopie Portrait

Abb. 18 Helen Chadwick, „The Oval Court“
1984-1986, Technik: Realkopie, Body Art

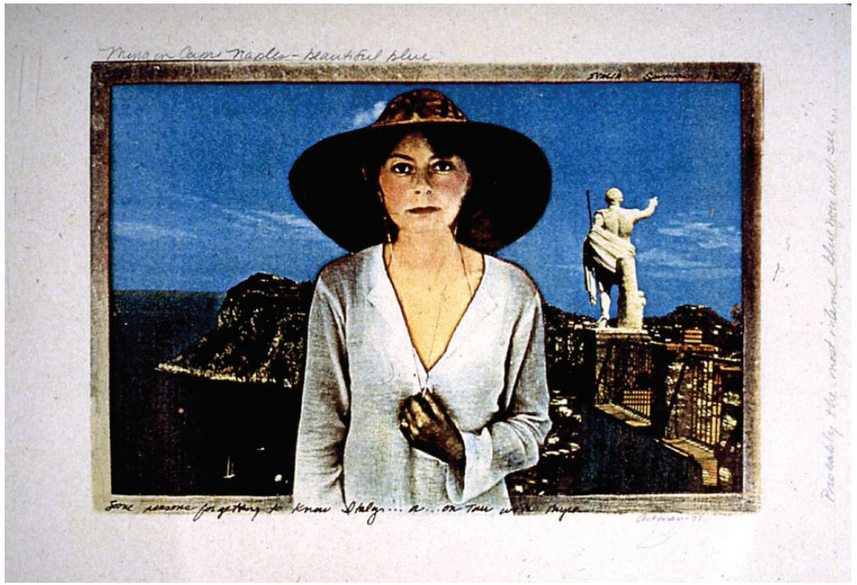


Abb. 19 + Abb.20 Barbara Astman, „On tour with Myra..."
1976; Technik: Variage, Farbxerographie

Abb. 21 + Abb. 22 Shepard Fairey, „André the Giant Has a Posse“
1989, „OBEY Giant“ 1994 – 2015, Technik: Blow-Up

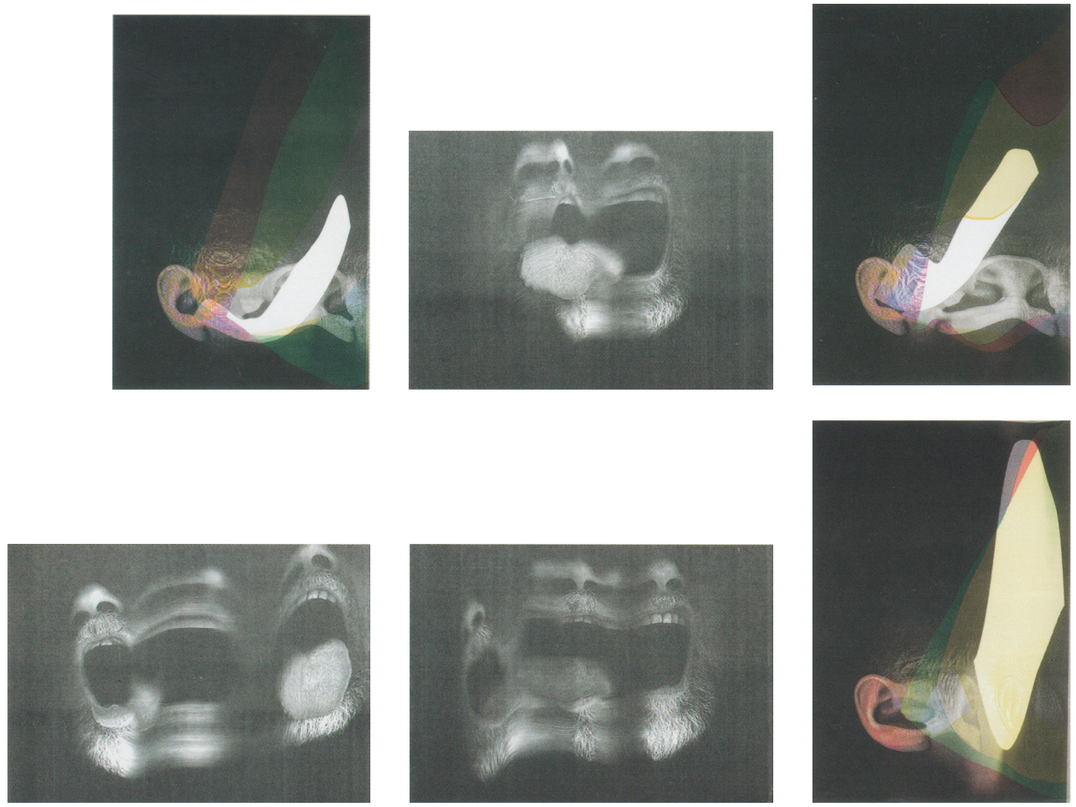


Abb. 23 Peter Huemer, „der Schrei"
2001, Technik: Realkopie, Copy-Motion

Abb. 24 Marc Mer, „data-date“
1993/94, Rauminstallation mit diversen Materialien (Kopiergeräte und Spiegel)

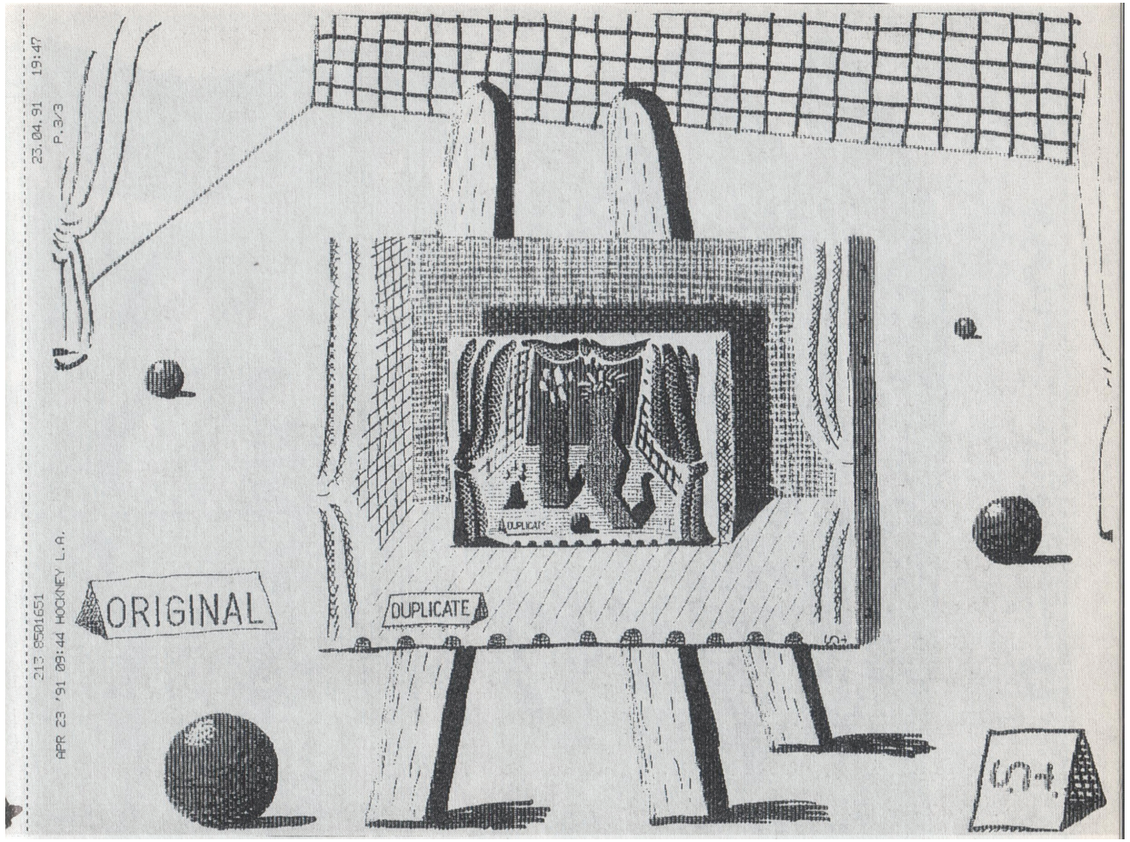


Abb. 25 David Hockney, „Original/Duplicate"
1991, Technik: Faxarbeiten

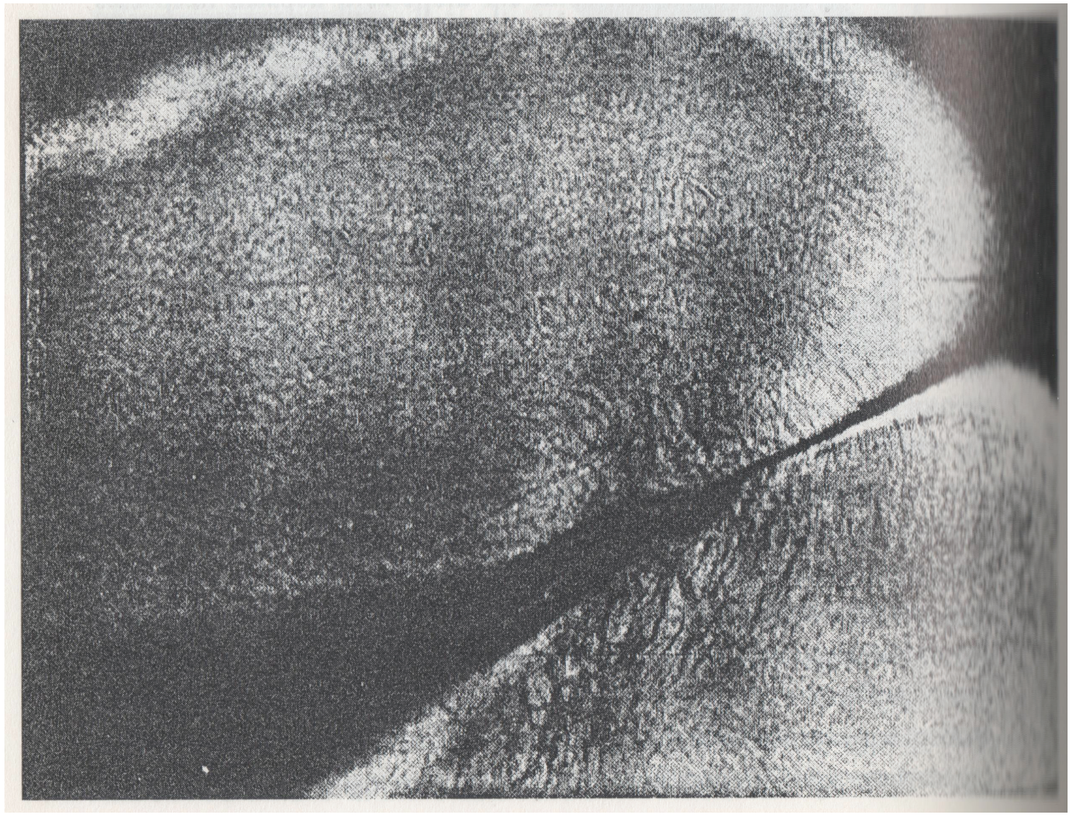


Abb. 26 Anonym, „Bürokopie"
o.J., Technik: Realkopie, Body Art

2.02 DAS RELIEF

Um den Prozess der „Topokopie“, sowohl von der technischen Seite her, als auch vom Blickwinkel der Benutzerfreundlichkeit aus, möglichst unkompliziert zu gestalten, erwies sich das Relief als geeignetstes Ausgabemedium.

Eines der Hauptkriterien war es, ein Gerät zu entwickeln, das ähnlich wie ein Kopierer, nicht nur von eingeschultem Fachpersonal bedient werden kann. Obwohl die Verwendung des Topokopierers – wie geplant – auch „unvorbereitet“ zu Ergebnissen führt, kann das volle Potenzial nur ausgeschöpft werden, wenn man sich zuvor mit den Möglichkeiten vertraut macht. Ähnlich wie bei der Copy Art ist es also für das erfolgreiche Benutzen des Prozesses nicht unbedingt notwendig, sich mit dem Thema Relief auszukennen, für eine ernsthafte künstlerische Auseinandersetzung ist dies jedoch unabdingbar. So können bei der Arbeit mit Topokopierer erzielte Effekte aus dem Kontext der Reliefkunst, wiedererkannt, gezielt herbeigeführt beziehungsweise nachgeahmt werden. Erst durch das Wissen über die klassische Vorgehensweise zur Herstellung eines Reliefs können Chancen sowie Einschränkungen, die das neue Verfahren mit sich bringt, als solche identifiziert werden.

Reliefs sind dem Bereich der Plastik zugeordnet, nehmen allerdings eine Zwischenposition zwischen Malerei und Plastik ein. Manche Relief-Künstler bevorzugen es sogar, wenn ihre Werke im Kontext der Malerei betrachtet werden.

Grundlegende Eigenschaft eines Reliefs ist das Vorhandensein einer Bezugsebene. Unter der Bezugsebene versteht man im Allge

meinen die mehr oder weniger ausgearbeitete Hintergrundebene, auf der die Figuren oder Objekte angeordnet sind. Diese kann durch Hineinarbeiten oder durch Auftragen während dem Arbeiten versetzt werden. Bei modernen Reliefs kann die Bezugsebene auch nur eine gedachte Ebene sein, entlang derer diverse Elemente arrangiert sind.

Das Relief hat seinen Namen vom Abheben vom Hintergrund. Tatsächlich werden allerdings beim in Stein gearbeiteten Relief nicht Formen angehoben sondern die Grundfläche abgesenkt/nach hinten verschoben.

Die übliche Kategorisierung von Reliefs erfolgt über die Intensität der Plastizität. Es wird dabei zwischen Flachrelief (Basrelief), Halbrelief und Hochrelief unterschieden. Eine Sonderstellung hat das aus der ägyptischen Kunst bekannte

versenkte Relief. Hier sind die Figuren in die Bezugsebene hineingearbeitet und mit eingeritzten Umrisslinien von ihr abgegrenzt. Bei vielen Werken ist eine konkrete Zuordnung zu einer dieser Kategorien nicht möglich, da sie Eigenschaften mehrerer Kategorien kombinieren. Beispielsweise wurde bei Reliefs in der Renaissance der Hintergrund als Flachrelief, der Vordergrund als Hochrelief ausgeführt. Im Manierismus hingegen wurde innerhalb figuraler Darstellungen häufig der aussagekräftigere obere Bereich – also

Abb.27 Beispiel für Hintergrund als Flachrelief, Vordergrund als Hochrelief

Gesichter und Oberkörper – als Hochrelief gearbeitet, während weniger wichtige Bereiche wie die Beine in der Technik des Flachreliefs erscheinen.

Traditionell kann die Herstellung eines Reliefs in zwei grundlegend unterschiedliche Methoden eingeteilt werden: Methoden, bei denen das Relief durch Abtragen entsteht (Bildhauer) oder durch auftragen und modellieren (Plastiker) gestaltet wird.

Bildhauerische Vorgehensweise

Die Wahl des Grundmaterials, des sogenannten Rohlings (z.B. Steinplatte), gibt bereits die Rahmenbedingungen für die Gestaltung des Reliefs vor. Naturgemäß kann kein Teil des fertigen Reliefs über die vordere Seite hinaus stehen. Die Rückseite dient dem Bildhauer während der Arbeit als Referenzebene.

Bei der traditionellen Vorgehensweise werden Skizzen und modellierte Vorlagen erarbeitet, bevor der Bildhauer mit dem Abtragen von Material beginnt. Diese dienen dem Bildhauer beim Arbeiten am Bossen zur Orientierung. Zunächst werden die Umrisse der Figuren mit einem Stift vorgezeichnet und fast senkrecht in die Tiefe geschlagen. Nach den Umrissen wird die Binnengliederung eingemeisselt, wobei das überflüssige Material schichtweise abgetragen wird, um ein unerwünschtes Abbrechen von Elementen zu verhindern. Natürlich ist es auch möglich, weitere Zwischenebenen herauszuarbeiten. Nach dem Ausmeisseln wird damit begonnen die Kanten zu runden und die Plastizität der Figuren herauszuarbeiten. Es gibt dabei unterschiedliche Grade, wie detailliert beziehungsweise nuanciert die Figurenflächen weiterbearbeitet werden. Die einfachste Möglichkeit ist, nur die Abstechkanten zu verrunden und die Figurflächen eben zu belassen. Der nächstfeinere Schritt ist es, die Figurenflächen mehr oder weniger stark zu runden, es bleibt allerdings noch etwas von der Kantentiefe übrig. Eine aufwendigere Variante stellt dier Herstellung einer halben Vollplastik dar. Die komplexeste Ausführungsvariante ist, die Figurenflächen bis zur Grundplatte so zu runden, dass auf Grund der Berücksichtigung der perspektivischen Verkürzung der Eindruck einer Vollrundung entsteht.

Unterläuft dem Bildhauer bei der Arbeit am Bossen ein Fehler, kann er nur versuchen das gesamte Relief noch tiefer in den Stein zu arbeiten. Dies ist allerdings nur möglich, sofern noch genug Material, auch Fleisch genannt, vorhanden ist.

Modellieren eines Reliefs

Auch beim Modellieren ist es hilfreich, zuerst Skizzen und Vorstudien herzustellen. Im Gegensatz zur bildhauerischen Vorgehensweise wird mit einer Unterlagsplatte als Hintergrundebene gestartet und Material aufgebaut. Diese Arbeitsweise hat den großen Vorteil, dass beinahe beliebig viel Material aufgetragen, geformt, umgeformt und wieder weggeschnitten werden kann, bis man mit der Form zufrieden ist. Linien zur Orientierung für Änderungen können direkt in die Modeliermasse eingeritzt werden. Da durch das Material keine Grenzen vorgegeben werden, wo sich die Bezugsebene und Zwischenebenen befinden, muss der Plastiker sich diszipliniert darauf konzentrieren.

Abb. 28 Beispiel Egyptisches Relief

Üblicherweise dienen die hergestellten Modelle als Positivformen bei Gussverfahren. So ist es möglich, die Bildnisse in haltbarere Materialien wie Metall oder Gusssteine zu „übersetzen“.

Bei der Topokopie werden einzelne Vorgehensweisen beider Methoden verbunden. Beim Entwerfen des Reliefs wird wie beim Modellieren vorgegangen: es können Formen nach Belieben hinzugefügt oder wieder entfernt werden. Dies kann entweder analog durch Ergänzung oder Umstrukturierung der Objekte unter dem Scannen erfolgen oder digital durch Veränderungen am Graustufenbild erreicht werden. Im Gegensatz zu den traditionellen Methoden ist es bei der Topokopie möglich, unterschiedliche Ebenen unabhängig von einander zu gestalten und später digital zusammenzufügen. Auch die Skalierung und Plastizität der einzelnen Ebenen sowie die Position der Ebenen und Objekte kann frei gewählt werden. Der Entwurf der Hintergrundebene ist meist einfacher und kommt mit weniger Niveausprüngen aus, als komplexe Formen im Vordergrund. Daher ist es leichter möglich, den Hintergrund als Graustufenbild zu entwerfen. Durch den momentanen Aufbau des Gerätes ist der Modellierbereich auf 60 x 60 x 20 cm beschränkt. Dies erinnert an die Begrenzungen, die bei der Bildhauerei gegeben sind. Auch die Produktion des fertigen Entwurfs ähnelt dem bildhauerischen Prozess. Wie beim Herausarbeiten der einzelnen Ebenen, wird auch bei der Herstellung der Negativform der Sand schichtweise abgetragen, damit möglichst keine Bereiche abbrechen. Ebenso wird mit unterschiedlichen Werkzeugen von grob nach fein gearbeitet.

Abb. 29 Beispiel Übergang zur Vollplastik

III UMSETZUNG

1 ENTWURF

Für die Erstellung eines Entwurfs können sowohl physische Objekte als auch digitale Elemente frei kombiniert werden.

1.01 ANALOGES ARBEITEN MIT DEM SCANNER

Die zu scannenden Objekte werden im Scannbereich positioniert und der Scannvorgang gestartet. Bei der Wahl der Objekte, die mit dem Scanner digitalisiert werden sollen, ist auf die Oberflächenbeschaffenheit zu achten. Zu glatte Oberflächen reflektieren den Laserstrahl zu stark, was in unkontrollierbaren hellen Werten resultiert, während zu dunkle Objekte den Laserstrahl quasi schlucken, wodurch schwarze Stellen entstehen. Die Unterlage des Scannbereichs ist mit einem Raster versehen, der bei der Anordnung der Objekte hilft. Speziell wenn mehrere Scanns kombiniert werden sollen, erleichtert er das Positionieren weiterer Objekte.
Die Standardabmessungen des Scannbereichs betragen 40 cm x 40 cm x 20 cm. Für diese Abmessungen und eine Auflösung von 0,5 mm beträgt die Scanndauer ca. 4 Minuten. Die maximalen Abmessungen des Scannbereichs betragen beim momentanen Aufbau des Gerätes 60 cm x 150 cm x 20 cm. Die momentan maximal erreichbare Auflösung liegt bei 0,33 mm.
Der Scanner erfasst das Relief der Objekte innerhalb des Scannbereichs, wobei Hinterschneidungen bewusst ignoriert werden. Bis zu einem gewissen Grad könnten Hinterschneidungen vom Scanner zwar erfasst werden, jedoch würde das Integrieren dieser Elemente sowohl die digitale Weiterbearbeitung für den Benutzer, als auch die Produktion des Gusselements erheblich erschweren.

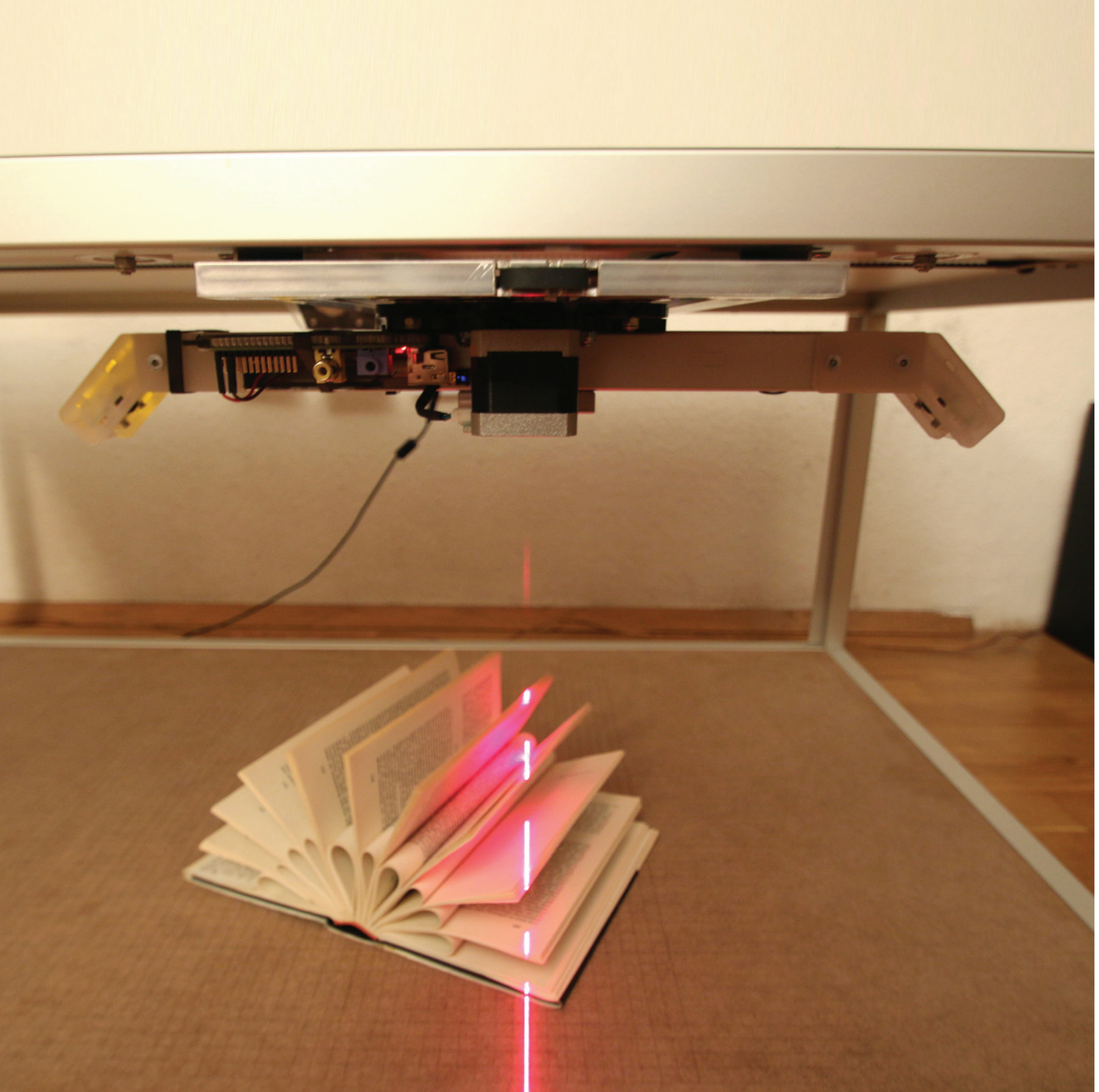
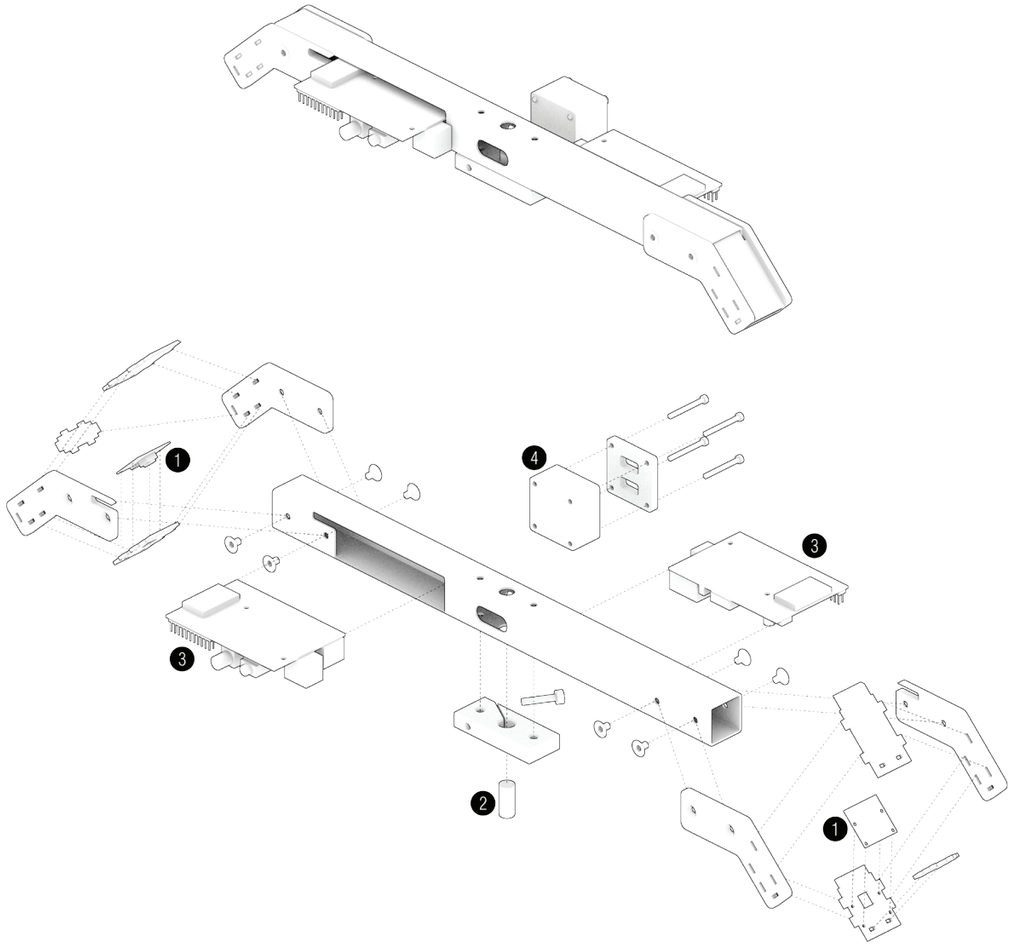


Abb. 30 Scanntisch

1.02 DIGITALISIERUNG

Da beim Topokopierer unterschiedliche Prozesse miteinander verknüpft wurden, war es notwendig, die einzelnen Elemente aufeinander abzustimmen, um das gewünschte Ergebnis zu erreichen. Für die Digitalisierung erwies sich die Konstruktion einer eigenen Scanneinheit, die an die Möglichkeiten des späteren Formenbaus angepasst ist, als notwendig.



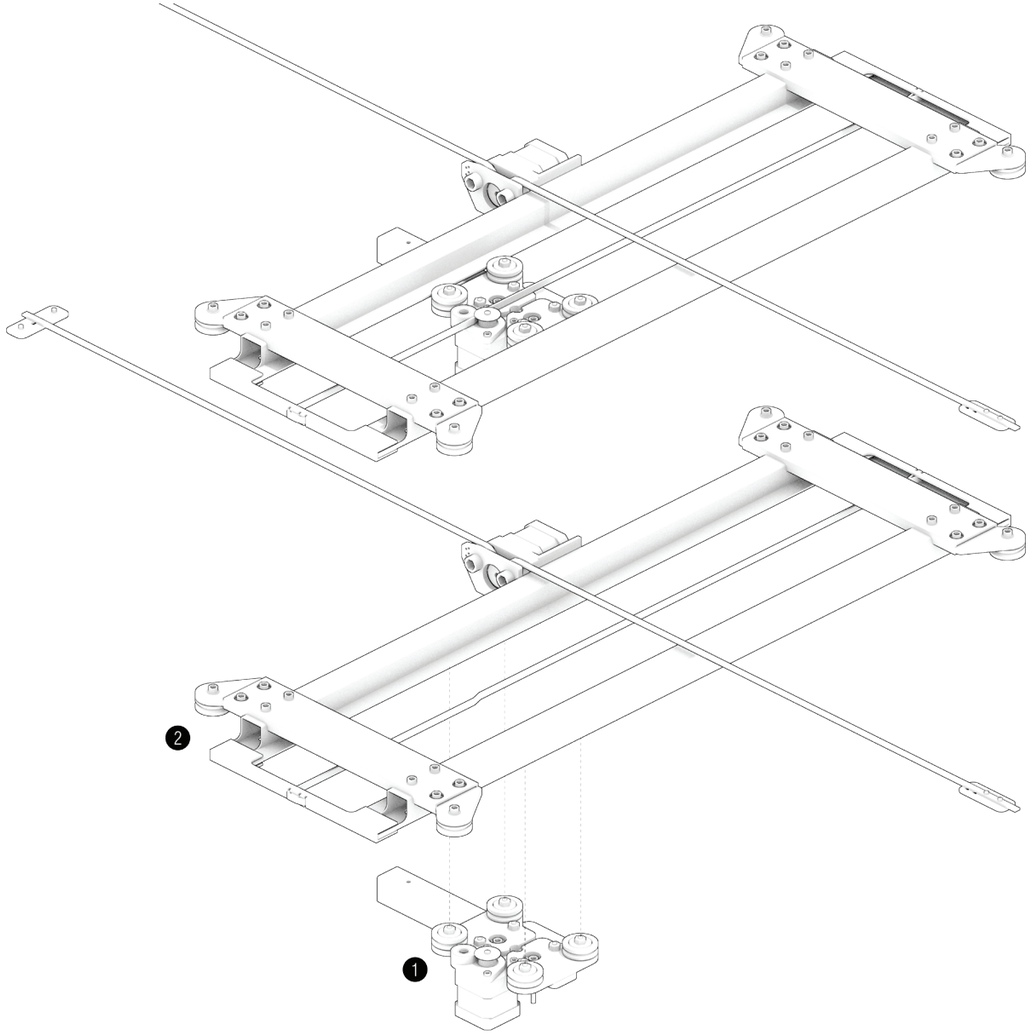
1 Kameramodul
2 Linienlaser
3 Raspberry Pi
4 12V auf 5V Spannungswandler

Abb. 31 Explosionszeichnung Scan-Einheit

Bauteile des Scanners

Die Scanneinheit setzt sich aus folgenden Bauteilen zusammen:

_ ein Linienlasermodul
_zwei Kameras
_zwei Raspberry Pi‘s (Modell B)
_zwei USB-WiFi-Dongles
_ein 12V auf 5V Spannungswandler
_selbst entworfene und hergestellte Teile, um alles zu einer soliden Einheit zu verbinden

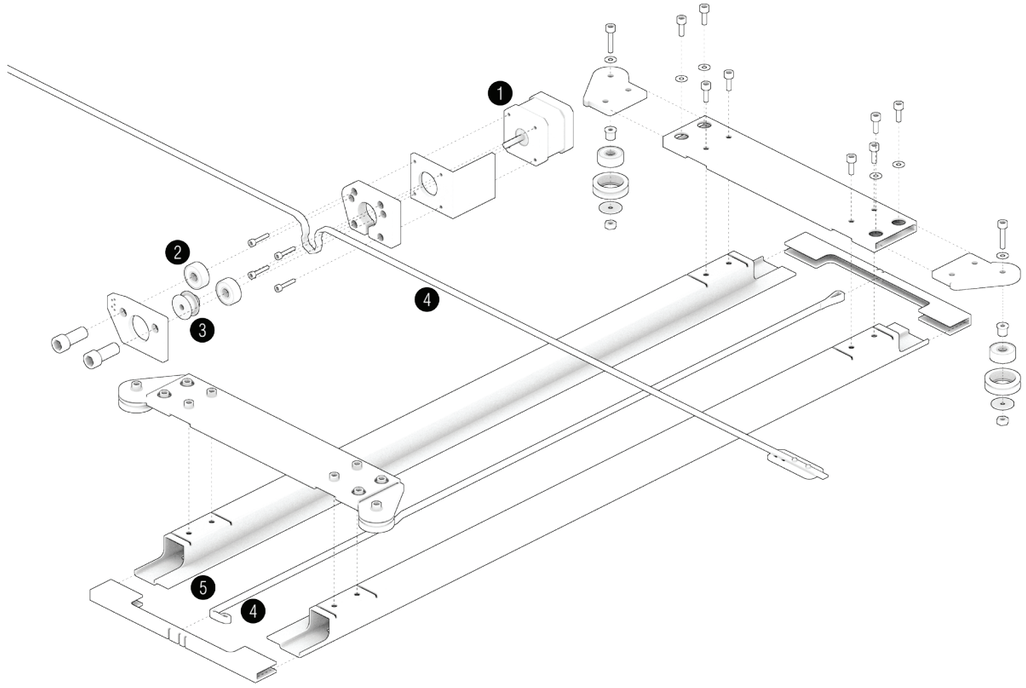


1 Bewegungseinheit x-Achse
2 Bewegungseinheit y-Achse

Abb. 32 Explosionszeichnung Bewegungs-Einheit

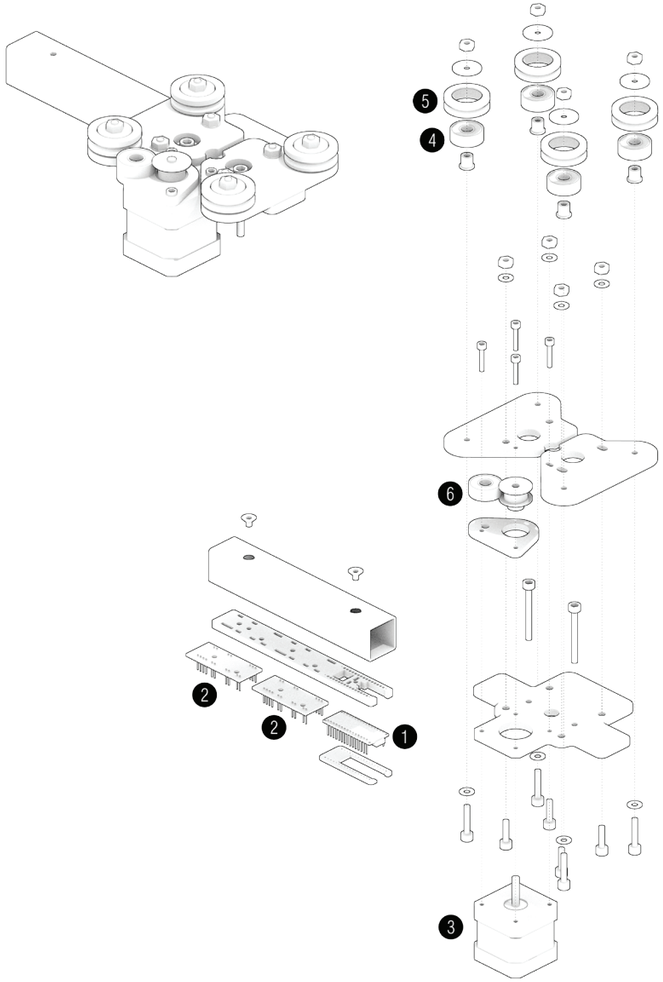
Die ergänzende Bewegungseinheit besteht aus:

_ einem Arduino (Nano)
_ zwei Schrittmotoren mit passenden Endstufen(SparkFun EasyDriver)
_ einem 12V Netzteil
_ 11 Kugellagern
_ 2 Zahnriemen
_ 8 Alusteck Steckverbinder und Profile



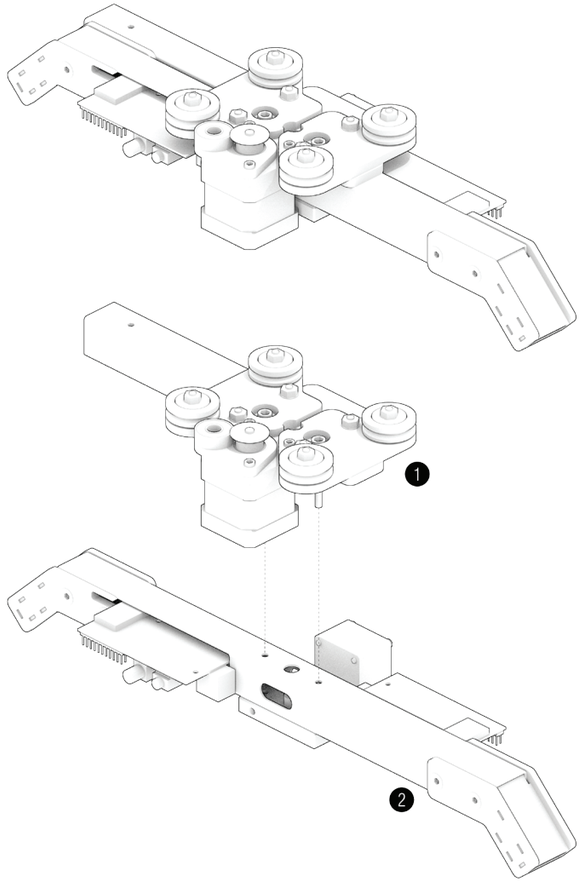
1 Steppermotor
2 Kugellager
3 Zahnscheibe
4 Zahnriemen
5 Alu-Profil 1-Steg

Abb. 33 Explosionszeichnung Bewegungseinheit Y-Achse



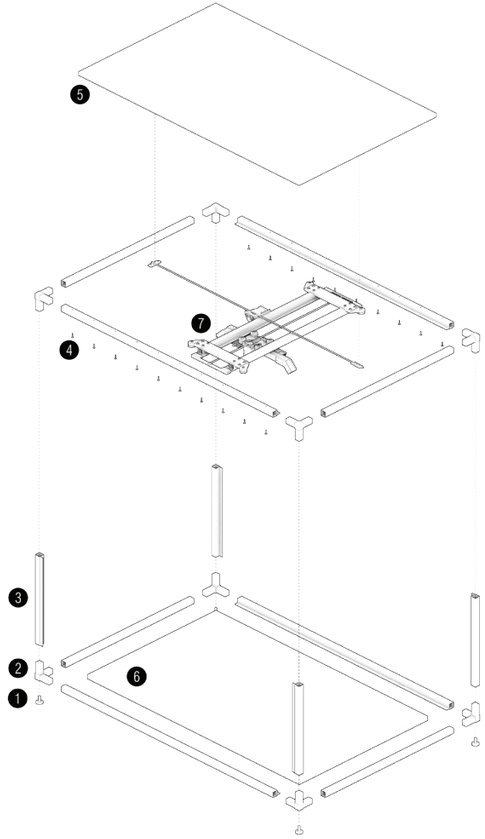
1 Arduino Micro
2 Stepper-Endstufe
3 Stepper-Motor
4 Kugellager
5 Rollen
6 Zahnscheibe

Abb. 34 Explosionszeichnung Bewegungseinheit x-Achse



1 Antriebseinheit x-Achse
2 Scan-Modul

Abb. 35 Explosionszeichnung Verbindung Antriebseinheit zu Scan-Einheit



1 Fuß
2 Steckverbinder
3 Alu-Profil 1-Steg
4 Schrauben
5 Tischplatte
6 Vorlagenplatte
7 Bewegungseinheit mit Scanner

Abb. 36 Explosionszeichnung Scann-Tisch

Abb. 37 Veranschaulichung Laser-Ebene
Abb. 38 Laser-Linie aus Kameraperspektive

Funktion des Scanners

Der Linienlaser projiziert mit einem Öffnungswinkel von 90 Grad eine rote Linie und spannt dabei eine Ebene auf. Wenn ein Objekt diese Ebene durchdringt, wird der Laser als Konturlinie auf der Oberfläche des Objekts sichtbar. Wird die Linie allerdings von einem Punkt innerhalb der Laserebene betrachtet, erscheint sie als Gerade. Um so größer der Winkel zwischen Blickachse und Laserebene ist, umso deutlicher wird das Relief sichtbar. Bei einem Winkel von 90 Grad erscheint die Relieflinie unverzerrt.

Würde man eine Kamera mit der Blickachse orthogonal zur Laserebene positionieren, müsste man die von ihr aufgenommene Laserlinie nur noch skalieren, um die tatsächliche Konturlinie zu erhalten. Solange sich die Position vom Laser zur Kamera nicht verändert, ändert sich auch der Skalierfaktor nicht. Somit ist es nur einmal notwendig, den Skalierfaktor zu ermittel. Aus allen danach von der Kamera aufgenommenen Konturlinien können die genauen Maße ermittelt werden. Es gibt mehrere Möglichkeiten den Skalierfaktor zu ermitteln. Eine einfache Variante ist es, die Kamera ein Objekt, dessen Dimensionen bekannt sind, aufnehmen zu lassen und dann die aufgenommenen Dimensionen mit den tatsächlichen zu vergleichen.

In der Realität ist es allerdings nicht leicht zu bewerkstelligen, die Kamera des Scanners orthogonal zur Laserebene zu positionieren. Will man nämlich den Bildbereich der Kamera optimal nutzen, müsste die Kamera sich auf halber Höhe im Scannbereich befinden. Dort würde die Kamera jedoch mit den zu scannenden Objekten kollidieren und die Sicht auf die Laserlinie wäre häufig verdeckt. Um diese Probleme zu lösen, wird die Kamera um den

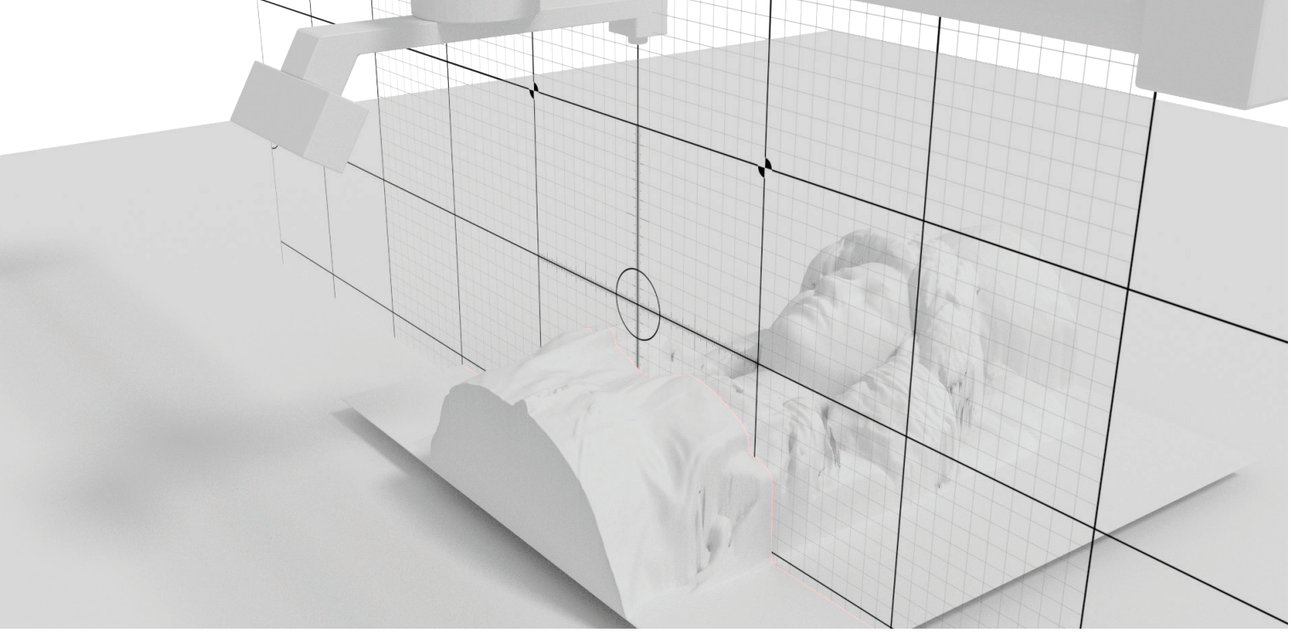


Abb. 39 Veranschaulichung Raster in Laser-Ebene
Abb. 40 Raster aus Kameraperspektive

Schnittpunkt zwischen Blickachse und Laserebene rotiert und zwar mindestens soweit, bis die Unterkante der Kamera oberhalb des Scannbereichs liegt. Das von der Kamera aufgenommene Bild muss jetzt allerdings nicht mehr nur skaliert, sondern zusätzlich auch perspektivisch entzerrt werden, um die tatsächliche Form der Konturlinie zu ermitteln.

Die optimale Position und Ausrichtung hängt von der verwendeten Kamera und deren Blickfeld ab. Die Anzahl der nicht erfassbaren Bereiche, kann reduziert werden, indem eine zweite Kamera hinzugefügt wird. Diese filmt die Laserlinie aus der entgegengesetzten Richtung, wodurch sich „blinde Flecken“, also Bereiche in denen die Laserlinie verdeckt wird, minimieren. In einer Versuchsreihe wurden 15, 30, 45 und 60 Grad Ausrichtungen zur Laserebene getestet. Bei 45 Grad konnten die besten Ergebnisse erzielt werden: Hier war ein besonders gutes Verhältnis zwischen wenig Verschattung der Laserlinie und effektiv genutztem Bildbereich gegeben. Das verwendete Kameramodul erreicht eine Auflösung von 1292 x 972 Bildpunkten bei konstanten 42 Bildern pro Minute. Der genutzte Bildbereich ist trapezförmig, wobei die Grundseiten aus 740/1220 Bildpunkten bestehen, während die Höhe 630 Bildpunkte misst. Da das entzerrte und skalierte Bild einem Quadrat mit 200 x 200 mm entspricht, lässt sich über die genutzten Bildpunkte eine Auflösung von 1/3 mm erreichen. Digital kann die Auflösung über dies noch verfeinert werden.

Solange sich die Lage von Scanneinheit zum Objekt nicht verändert, kann nur eine Konturlinie erfasst werden. Deshalb ist es notwendig, die Scanneinheit durch eine Bewegungseinheit zu ergänzen. Dies ermöglicht ein gezieltes Positionieren und gleich-

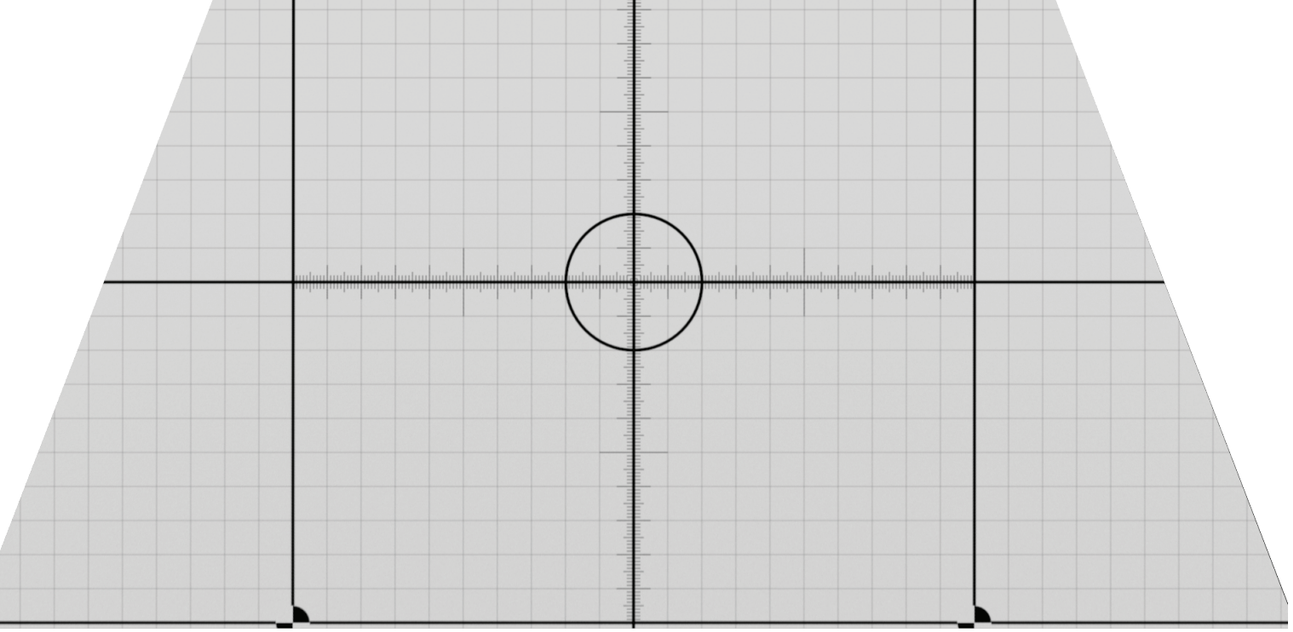


Abb. 41 Perspektivische Entzerrung Raster
Abb. 42 Perspektivische Entzerrung Kamerabild

mäßiges Bewegen der Scanneinheit. Um den kompletten Vorlagenbereich zu erfassen, wird dieser in mehreren Bahnen abgefahren. Die Geschwindigkeit mit der gefahren wird, hängt dabei von der gewünschten Auflösung ab. Bei einer Bildrate von 42 Bildern pro Sekunde werden 42 Konturlinien in einer Sekunde erfasst. Soll eine Auflösung von 0,5 mm erreicht werden, muss folglich die Scanneinheit mit 21 mm pro Sekunde bewegt werden. Für die Steuerung der Bewegungen wird ein Arduino eingesetzt, auf dem die GRBL-Software[i] läuft. Positions- und Geschwindigkeitsvorgaben werden von der Scannsoftware berechnet, als G-Code per WiFi zu einem der Raspberry Pi's gesendet und von dort per seriellem Terminal an den Arduino übermittelt. Die Software auf dem Arduino wandelt die Befehle in Impulse um, mit denen Stepper Endstufen und die daran angeschlossenen Motoren gesteuert werden.

Die beiden Kameras werden jeweils über einen Raspberry Pi angesteuert. Das aufgenommene Video wird per WiFi direkt auf den Rechner, auf dem die Scannsoftware läuft, übertragen und dort, noch während der Scann ausgeführt wird, ausgewertet.

Jede der beiden Kameras liefert pro gescannter Bahn ein Videofile, wobei dessen jeweilige Position in X- und Y-Richtung bekannt ist. Der Name des Videos entspricht der Bahn und der Kamera, von der es aufgenommen wurde.

Sobald das Videofile über Wifi von der Kamera an den Computer gesendet wurde, wird es vom Programm geöffnet und bearbeitet. Zunächst wird jedes Einzelbild entzerrt und in Graustufen umgewandelt. Danach wird das Bild Spaltenweise analysiert.Jede Spalte entspricht einer Einheit in X-Richtung. Die Position des hellsten

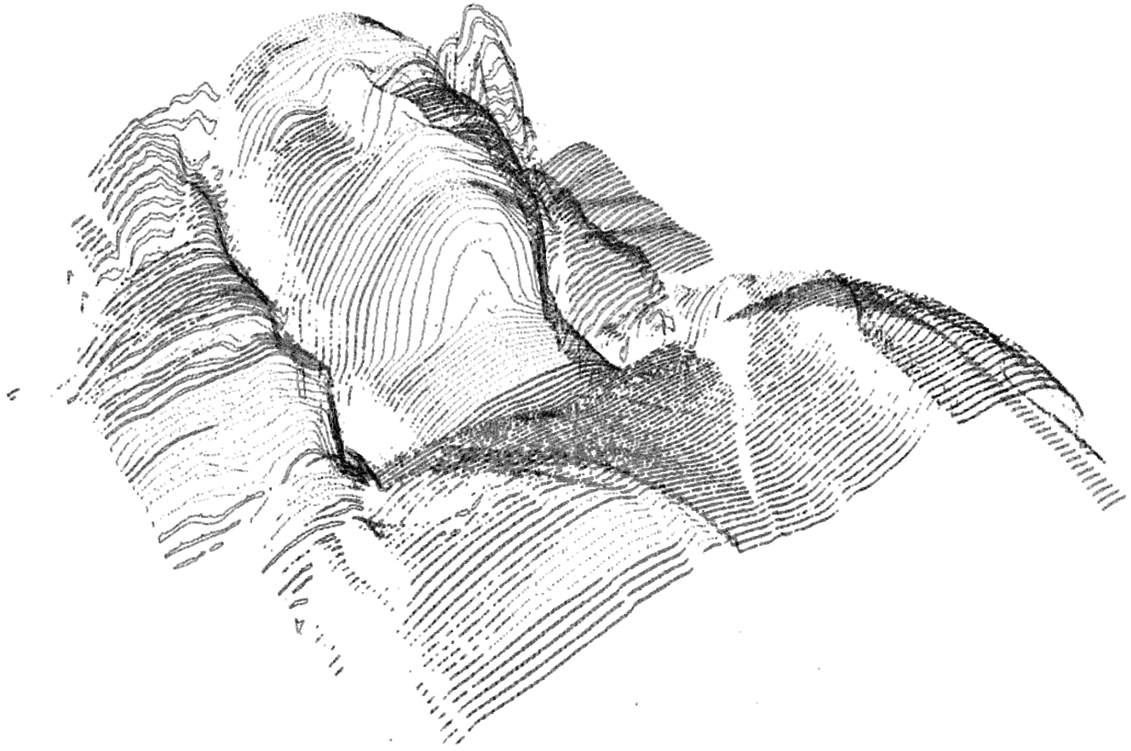


Abb. 43 Punktwolke

Wertes pro Spalte entspricht der Höhe der Konturlinie bzw. dem gesuchten Wert in Z-Richtung.

Erreicht der hellste Wert einer Spalte allerdings nicht einen Mindestwert wird er, um Fehler zu minimieren, ignoriert. Die ermittelten Werte werden in einer Matrize gespeichert. Die Anzahl der Zeilen und Spalten der Matrize entsprechen den Abmessungen des Vorlagenbereichs mal der Auflösung in Punkten pro Millimeter. Aus den absoluten Werten für die Position des Films werden die Positionen der Einzelbilder abgeleitet und davon wiederum die Position der Spalten des entzerrten Bildes berechnet. Somit können die ermittelten Höhenwerte an die entsprechende Stelle in der Matrize eingetragen werden. Der Höhenwert wird als 16 bit Graustufenwert in der Matrize notiert. Schwarz entspricht dabei der Untergrenze des Scannbereichs, Weiss stellt die Obergrenze dar. Da es durch Verzögerungen beim Aufnahmestart der Kamera zu Positionsabweichungen in Fahrtrichtung des Scanners kommen kann, werden die Resultate jedes Videos in eine eigene Matrize geschrieben. Nachdem alle Videos des Scannvorgangs fertig ausgewertet sind, werden die resultierenden Matrizen so zueinander ausgerichtet, dass sie sich bestmöglich überlagern. Die resultierende kombinierte Matrize wird als Graustufenbild abgespeichert.

In früheren Versuchen wurde aus den erfassten Werten ein geschlossenes Mesh generiert. Dies erwies sich durchaus als brauchbar zur direkten Herstellung von Werkzeugbahnen für einen 3D-Druck, für Modifikationen an den erfassten Daten war allerdings ein sogenannter Meshmodeler notwendig.

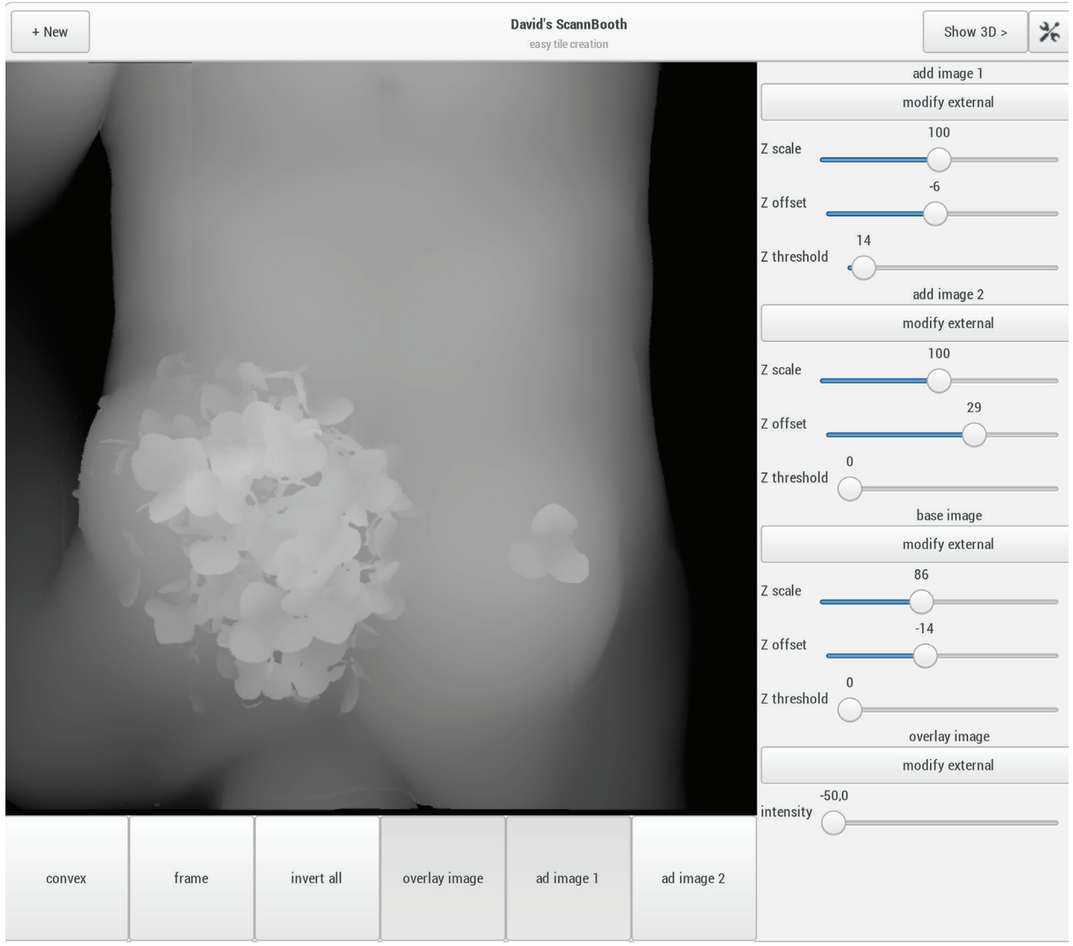


Abb. 44 Interface Scann-Programm

1.03 DIGITALES ARBEITEN

Das vom Scanner erfasste Relief wird in der Scannsoftware als Graustufenbild dargestellt. Diese Art der Darstellung ermöglicht die Weiterverarbeitung des Reliefs mit Zeichen- und Fotoverarbeitungsprogrammen. Die X und Y Position der Bildpunkte entsprechen den horizontalen Achsen des Scannbereichs. Die Helligkeitswerte der Bildpunkte entsprechen der vertikalen Erhebung der gescannten Oberflächen. Schwarz bedeutet eine Höhe von 0 mm, Weiß eine Höhe von 200 mm

Die Abmessung des Bildes hängt von den Abmessungen des Scannbereichs in X und Y Richtung und der Auflösung des Scans ab. Bei einer Auflösung von 0,5mm werden pro 1mm zwei Punkte erfasst. Bei einem Scannbereich mit 40cm mal 40cm und einer Auflösung von 0,5mm hat das resultierende Graustufenbild 800 mal 800 Bildpunkte.

Durch das Verändern der Grauwerte kann Einfluss auf die vertikale Erhebung der Form genommen werden. Wird die Helligkeit des Bildes verstellt so werden die gescannten Elemente angehoben oder abgesenkt. Änderungen am Kontrast wirken sich auf gescannte Elemente als Skalierung in Z-Richtung aus.

In der Scannsoftware können auch mehrere Scanns kombiniert werden. Dabei werden die Grauwerte der beiden Graustufenbilder Punkt für Punkt miteinander verglichen und der jeweils hellere Wert für das kombinierte Bild genutzt.

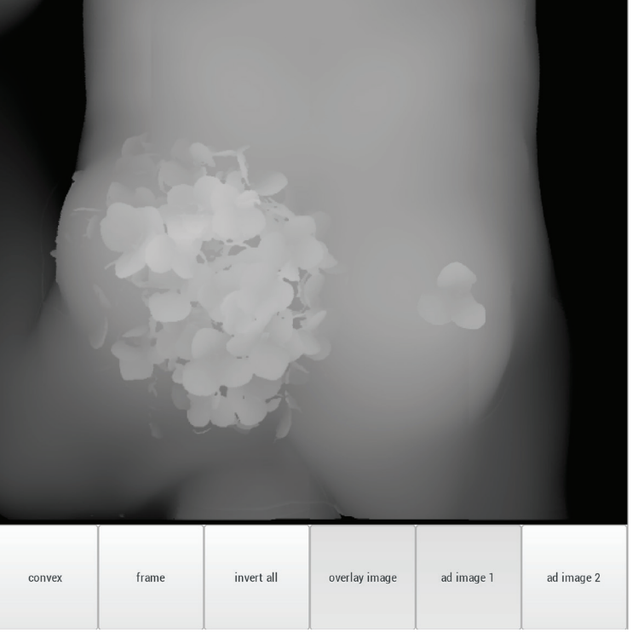
Eine weitere Bearbeitungsmöglichkeit ist das hinzufügen eines Overlaybildes. Die Grauwerte der Bildpunkte des Overlay-Bildes werden zu den Grauwerten des Scanns addiert bzw. subtrahiert.

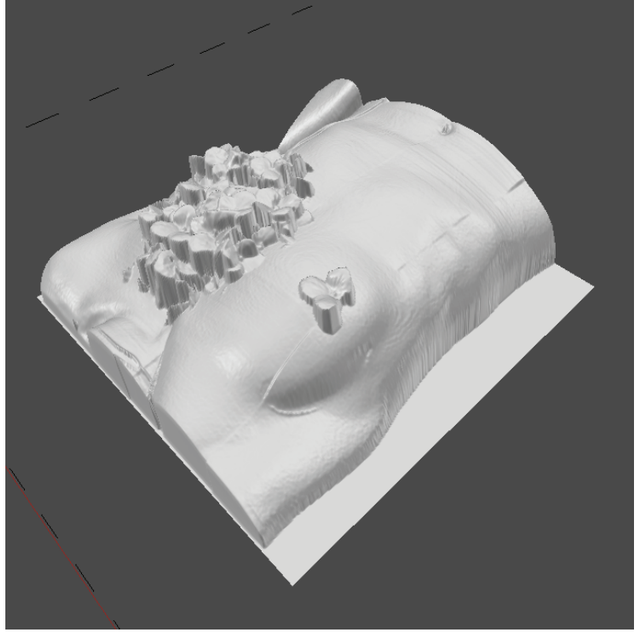
Sowohl der dem Entwurf als Basis dienende Scann als auch importierte zusätzliche Scanns und Overlaybilder können in externen Fotoverarbeitungsprogrammen modifiziert werden. Nach dem Speichern von Änderungen und dem Schließen des externen Programms wird im Scannprogramm eine aktualisierte Ansicht angezeigt.

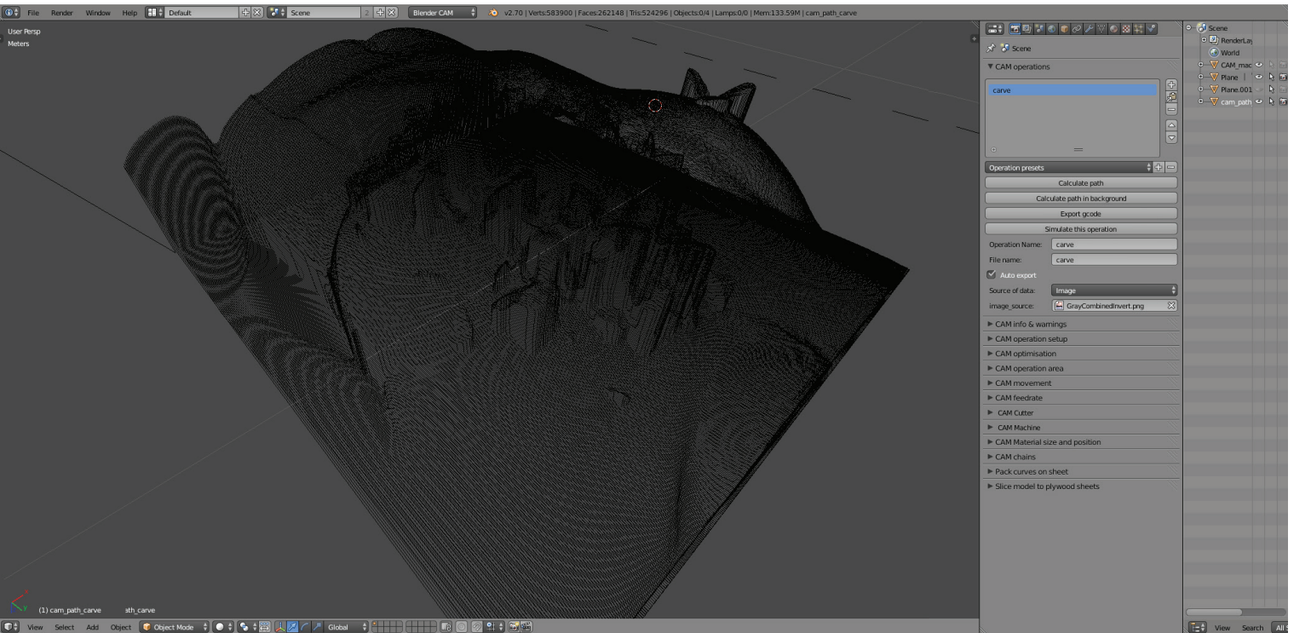
Im Scannprogramm können auch bereits vordefinierte Effekte angewandt werden. Diese beinhalten das Hinzufügen eines Rahmens, das Simulieren eines convexen Untergrundes und das Invertieren sämtlicher Helligkeitswerte.

Während dem digitalen Bearbeiten des Entwurfes kann auch eine dreidimensionale Vorschau im Programm „Blender“ genutzt werden.

1 Grbl ist eine open source Software, die die Bewegung von Maschinen steuert. Siehe dazu: https://github.com/grbl/grbl, Zugriff am 28.05.2015.







2 PRODUKTION

Die Umwandlung des digitalen Entwurfs zurück in ein dreidimensionales Modell stellt einen wesentlichen Schritt im Topokopie-Prozess dar. Dafür wird ein eigens entwickeltes Vakuum-Sand-Schnitz-Verfahren zur Herstellung einer Sandgussform eingesetzt. Nachfolgend werden die einzelnen notwendigen Schritte erklärt.

2.01 VORBEREITEN DER DATEN

Bevor der Schnitzvorgang gestartet werden kann, müssen aus dem Graustufenbild entsprechende Werkzeugbahnen berechnet werden. Da ein Gusselement hergestellt werden soll, wird eine Gussform mit der Negativform der gewünschten Oberfläche benötigt. Hierfür invertiert das Scannprogramm zunächst die Grauwerte des Graustufenbildes und spiegelt den Entwurf.

Durch das Drücken des „Show 3D“-Knopfs im Scannprogramm wird das Graustufenbild als 3D Modell im Programm „Blender CAM“ angezeigt. Beim diesem Wechsel in den Vorschaumodus werden die benötigten Voreinstellungen zum Generieren des Maschinencodes automatisch vom Programm angelegt. Die Parameter für die Maschine, die das Schnitzen übernimmt, also die Geschwindigkeit, Abmessungen und Werkzeuge sind bereits vordefiniert. Es muss nur noch durch das Drücken des „Calculate path“-Buttons der Berechnungsvorgang der Werkzeugbahnen gestartet werden. Handelt es sich um einen besonders kleinteiligen Entwurf, so sind mehrere Pfade notwendig. In einem ersten Schritt wird mit einem groben Werkzeug rasch Material abgetragen. In einem zweiten Schritt werden mit einem feineren Werkzeug Details herausgearbeitet.

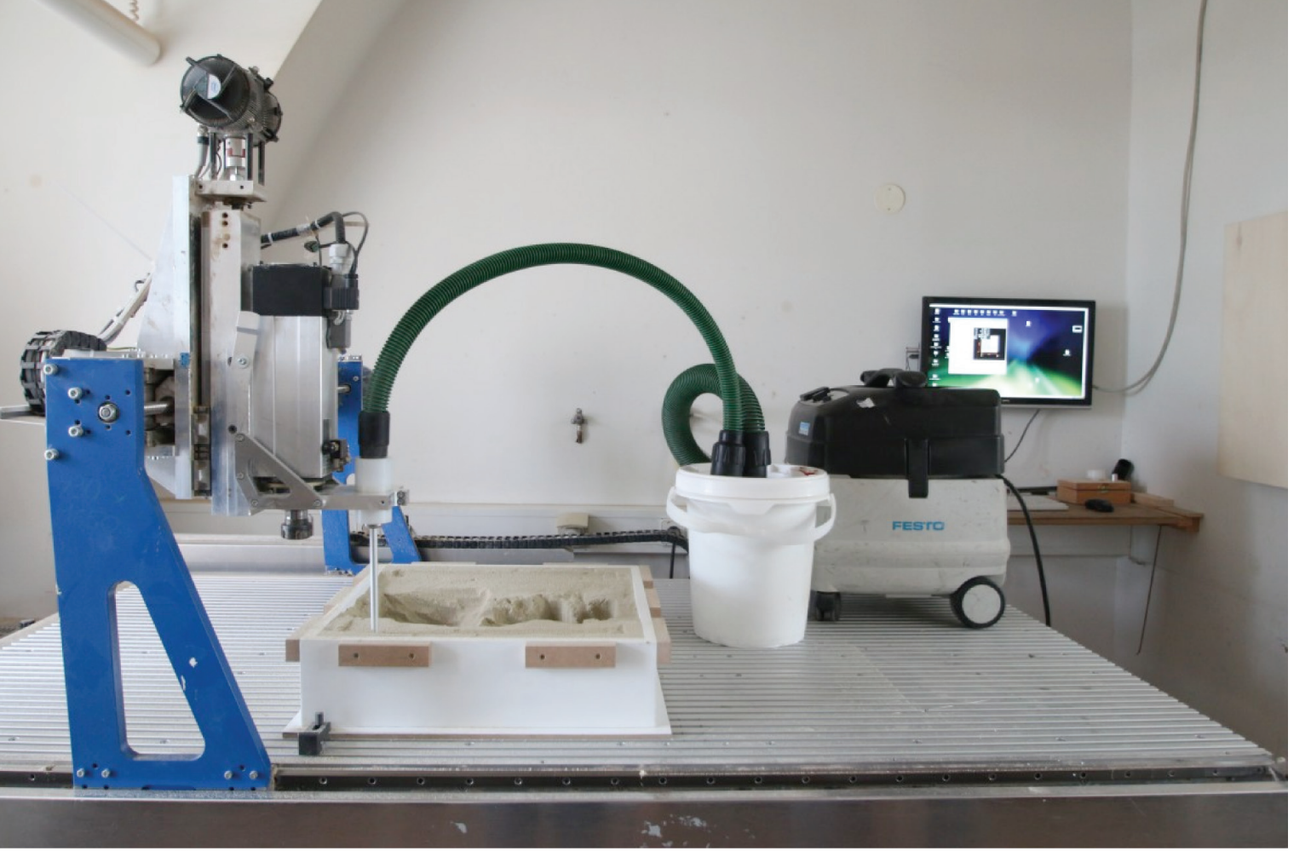
Der so generierte Maschinencode wird von der Maschinen-Steuerungssoftware ausgeführt.

Abb. 45 Entwurf im Scannprogramm
Abb. 46 3D Modell Voransicht
Abb. 47 Werkzeugbahnen

2.02 VORBEREITEN DES FORMKASTENS

Für ein schönes Ergebnis sowohl beim Schnitzen der Gussform, wie auch beim Guss selbst, ist es notwendig, den Formkasten gut vorzubereiten. Um eine möglichst stabile Form mit schönen Außenkanten schnitzen zu können, sollte der Formkasten in alle Richtungen mindestens fünf Zentimeter größer sein als das Objekt. Zudem sollte er aus einem stabilen Material bestehen, das sich nicht bei Feuchtigkeit verzieht oder aufquillt. Für eine einfachere Handhabung empfiehlt es sich, während des Einbringens und Verdichten des Sandes einen Rahmenaufsatz auf dem Formkasten anzubringen. Ist der Formsand ausreichend verdichtet oder gepresst, kann der Aufsatz abgenommen und die Sandoberfläche mit dem Formkasten bündig abgezogen werden. Als Formsand eignet sich beispielsweise eine Mischung aus neun Teilen feinkörnigen Quarzsands und einem Teil gemahlener Bentoniterde. Als „Bindemittel“wird Wasser eingesetzt. Die optimale Konsistenz ist dann erreicht, wenn der Sand rückstandsfrei geformt werden kann und bei entsprechendem Druck an nur einer Stelle bricht. Zerfällt er in mehrere Stücke, muss noch mehr Wasser eingebracht werden. „Zerfließt“ er wie ein weicher Teig oder wird klebrig, so enthält er bereits zu viel Feuchtigkeit. In dem Fall kann entweder trockener Sand hinzugefügt werden oder man wartet unter kontinuierlichem Rühren darauf, dass genug Feuchtigkeit verdunstet. Die richtige Konsistenz ist für eine erfolgreiche Herstellung der Guss unumgänglich, dazu feuchter Sand am Schnitzwerkzeug festklebt und die Absaugung verklebt oder stellenweise einfach „verrinnt“.

Abb. 48 - Abb. 51 Konsistenzüberprüfung Gusssand
Abb. 52 - Abb. 54 Befüllen Formkasten



2.03 SCHNITZEN

Ist der Formkasten vorbereitet, kann er an der dafür vorgesehenen Stelle unter dem Schnitzwerkzeug positioniert werden. Hierbei ist zu beachten, den Nullpunkt einzustellen, damit die jeweilige Maschine an der richtigen Stelle arbeitet. Das Schnitzwerkzeug kann einerseits auf Dreiachsfräsen, andererseits auf einem Roboterarm montiert werden. Zuerst wird ein grobes Schnitzwerkzeug in die CNC Maschine eingespannt. Mit diesem Werkzeug wird ein Großteil des Materials Abgetragen. Das Werkzeug ist innen hohl, damit das abgetragene Material gleich abgesaugt werden kann. Der entfernte Sand wird in einem Sandabscheider aufgefangen. Gemäß dem zuvor generierten Maschinencode fährt das Werkzeug die Bahnen ab und trägt so schichtweise den Formsand ab. Nach dem groben Entfernen des überflüssigen Materials wird auf ein feineres Werkzeug gewechselt. Mit diesem werden Details herausgearbeitet und die Spuren des groben Werkzeugs geglättet. Wenn auch der zweite Schnitzdurchgang beendet ist, kann das Gussmaterial vorbereitet werden.

Abb. 55 Setup für den Schnitzvorgang
Abb. 56 + Abb. 57 Verschiedene Werkzeuge während des Schnitzvorgangs



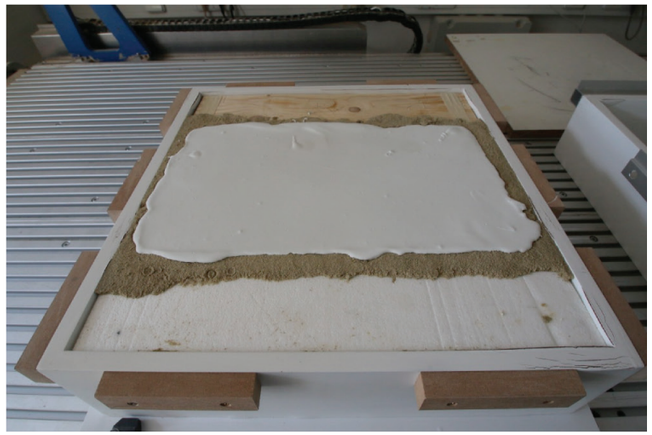



Abb. 58 + Abb. 59 Fertige Negativform
Abb. 60 + Abb. 61 Ausgießen
Abb. 62 + Abb. 63 Entformen

2.04 GIESSEN

Aus dem Gewicht des abgetragenen Sandes kann darauf geschlossen werden, wie groß das Volumen des Gussobjekts ist und dem entsprechend, wie viel Gusswerkstoff benötigt wird. Als Gusswerkstoff eignen sich einerseits Kunststeine auf Zementbasis, andererseits diverse Metalle wie beispielsweise Aluminium oder Bronze.

Bei der Verwendung von Zement sollte Flussmittel verwendet werden, um Gussfehler zu minimieren, ohne den Formkasten „rütteln“ zu müssen. Zudem kann so der Wassereintrag verringert werden. Es wurden sowohl Tests mit Schnellzement und mit Weißzement durchgeführt. Beides eignet sich gut zum Ausgießen der Sandform und verbindet sich auch ohne Trennmittel nicht mit dem Formsand. Das Gussmaterial sollte vorsichtig in die Form gegossen werden. Die Zeit bis das Gussstück ausgehärtet ist, hängt vom verwendeten Material ab.

2.05 ENTFORMEN

Nach dem Härten beziehungsweise Abkühlen des Gusses werden zunächst die Kanten vom Sand entfernt. Danach kann das Objekt aus dem Kasten gehoben werden. Um möglichst wenig Sand zu verlieren, wird das Gussstück abgesaugt, wobei der gepresste Sand mit Hilfe eines Kunststoff oder Holzstäbchens abgesprengt werden kann. Auch ein Bürstenaufsatz oder Pinsel eignen sich gut zum Entfernen der Sandrückstände.

IV ZUSAMMENFASSUNG

Das Scannen

Der Scannprozess konnte soweit beschleunigt werden, dass relativ schnell und mit vielen Objekten und Materialien experimentiert werden kann, sofern sie die maximale Größe des Vorlagenbereichs nicht überschreiten. So kann bereits nach einer kurzer Zeit an der Maschine ein Gefühl dafür entwickelt werden, was zu guten Scannergebnissen führt beziehungsweise welche Materialien und Formen ungeeignet sind. Objekten, die stark reflektieren oder zu viel Licht schlucken, können mit Puder, Malerkrepp, Cyclododecan Spray oder feinen Tüchern präpariert werden. Gibt es beim Scann Probleme wegen der Verschattung des Lasers, kann es helfen die Orientierung des Objekts zu ändern oder die Überlappung der Scannbahnen zu erhöhen.

Obgleich Veränderungen digital vorgenommen werden können oder in der Software Effekte und Filter bereitstehen, ist es besonders reizvoll, schon während dem Scannprozess einzugreifen. Ähnlich wie bei der Copy Art können einige Effekte nur erzielt werden, wenn der Scann Prozess verstanden und entsprechend manipuliert wird. Das auf diese Art erzeugte Ergebnis kann häufig nur mit erheblich größerem Aufwand digital imitiert werden.

Das Verfahren eignet sich besonders gut für Bodyart, da die Laserlinie auf der Hautoberfläche gut von den Kameras erkannt wird. Allerdings stellt die Dauer eines Scanns ein Problem für bewegliche beziehungsweise sich bewegende Objekte dar. Verändert sich die Position eines Objekts oder Körperteils während dem Scannen, werden charakteristische Kanten im Graustufenbild sichtbar. Wird beispielsweise zu viel oder tief geatmet, wird eine Wellenbewegung im Scann sichtbar.

Die Software

Die Funktionen die momentan in die Software integriert sind, setzen sich aus jenen Funktionen zusammen, die absolut notwendig sind, um das Verfahren benützen zu können sowie einigen, die als Beispiel für zusätzliche Bearbeitungsmöglichkeiten dienen. Es handelt sich dabei jedoch nur um einen Bruchteil der Möglichkeiten, die in den Prozess eingebunden werden könnten. Für einen konkreten Anwendungsfall könnte der Workflow natürlich durch Anpassungen am Benutzerinterface und vorgefertigte Filter und Effekte optimiert werden.

Obgleich die Versuchskandidaten durchaus in der Lage waren, das Programm im momentanen Zustand zu bewältigen, waren auch nach der Einführung Hilfestellungen notwendig, um einzelne Schritte zu beschleunigen. Nach einer allgemeinen Anfangsskepsis, waren die meisten Testpersonen nach kurzer Zeit eifrig und ausgiebig am Experimentieren.

Externe Software

Externe Bildbearbeitungsprogramme sind durchaus geeignet, das Graustufenbild weiterzubearbeiten, allerdings ist es hilfreich, das jeweilige Programm vorher schon zu beherrschen. Eine der Hauptschwierigkeiten bei der Bearbeitung stellt das Graustufenbild selbst dar. Da bei Schwarz-Weiß-Bildern hell und dunkel als Licht und Schatten wahrgenommen werden, muss man sich zunächst an den veränderten „Informationswert“ der einzelnen Schattierungen gewöhnen. Auch die Auflösung und Farbwiedergabe des verwendeten Bildschirms kann die Arbeit wesentlich beeinflussen, da die Unterscheidung einzelner Grautöne mit freiem Auge für das Arbeiten essenziell notwendig ist.

Ausgabeverfahren

Das gewählte Sandschnitz-Verfahren ermöglicht es, besonders große Objekte günstig herzustellen. Mit den momentan zur Verfügung stehenden Werkzeugen sind der Komplexität jedoch Grenzen gesetzt. Verbesserungspotenzial gibt es sowohl bei den verwendeten Schnitzaufsätzen, als auch bei der Berechnung der Werkzeugbahnen.

Auch wenn die Dauer der Herstellung der Sandgussform mit diesem Verfahren im Vergleich zu anderen Gussverfahren sehr schnell ist, kann die Ausgabe nicht so schnell erfolgen, wie es bei einem Kopierer der Fall ist. Dies gilt momentan jedoch für alle 3D-Druckverfahren.

Potenzial und Perspektiven

Erwartungsgemäß bestätigte sich die Annahme, dass ein solcher Prozess funktionieren kann. Das Verfahren erwies sich als so außergewöhnlich, dass Künstler wie Gottfried Bechtold, Edgar „Esche“ Leissing, Alexander Viscio und Alexandra Wacker sich umgehend interessiert zeigten und zu Versuchen bereit erklärten. Aus terminlichen Gründen konnten die Testreihen mit den Künstlern jedoch nicht vor Abgabe dieses Textes durchgeführt werden.

Bei Versuchsreihen mit Personen ohne speziellen künstlerischen Hintergrund konnte festgestellt werden, dass schnell ein ansprechendes Ergebnis erzielt werden kann. Von einem raschen Erfolgserlebnis angespornt, wurden die Testpersonen zu weiteren Experimenten animiert, weshalb sich das Arbeiten in Kleingruppen als sinnvoll entpuppte.

Der Scannprozess selbst reizt durch das Abfahren der Konturlinien mit einer eigenen Ästhetik. Dieses „temporäre Kunstwerk“ eignet sich bestens für eine Anwendung im Rahmen einer Performance.

Das Paket aus Scanner, Steuerung und Software ergibt eine funktionstüchtige und in sich abgeschlossene Einheit. Bei der Arbeit mit dem Gerät konnten noch Verbesserungsmöglichkeiten die Ergonomie betreffend festgestellt werden. Beispielsweise wäre das Anordnung der Objekte praktischer, wenn die Vorlagenplatte wie eine Schublade herausgezogen werden könnte oder sich der Scanner nach oben klappen ließe. Ebenso wäre es wünschenswert, den Scanner in einer vertikalen Anordnung benützen zu können.

Der nächste „evolutionäre“ Schritt des Verfahrens, ist vermutlich die Weiterentwicklung zu einer kompletten dreidimensionalen Aufnahme und Herstellung von Objekten mit Hinterschneidungen. Um dies zu bewerkstelligen sind signifikante Modifikationen an Hard- und Software notwendig. Dies führt zwar zur Entwicklung neuer Möglichkeiten, bedeutet aber auch den Verlust einiger charakteristischer Eigenheiten. Mit Anpassungen der Hard- und Software wäre es zusätzlich möglich, eine Echtzeit-Darstellung der erfassten Daten zu erreichen.

Die speziell von den Künstlern gewünschte Erweiterung des Scannbereichs ließe sich hingegen leicht und kostengünstig verwirklichen.

V ANHANG

BIBLIOGRAPHIE

Baumann, Roland: Algorithmen, die Fantasien beflügen. In Prévost, Romain (Hg.): ETH Zürich Globe,1/2015

Bloch, Christoph: 3D Surface-Scanning and Reconstruction, Methods and Experiments. Wien 2004

Chadwick, Helen: Of Mutability. London 1986

Fastermann, Petra: 3D-Druck/ Rapid Prototyping. Berlin Heidelberg 2012

Fastermann, Petra: 3D-Drucken, Wie die generativen Fertigungstechnik funktioniert, Berlin Heidelberg 2014

Huemer, Peter: Image in Motion, Arbeiten mit dem Medium Fotokopie 1987-2002. Klagenfurt 2002

Kowalski, Klaus: Plastische Bilder. Bielefeld 1996

Mühleck, Georg: „Medium: Photokopie". Montréal 1987

Museum für Gestaltung Gewerbemuseum Basel: FotoKopie, Fotografie und Imitation. Basel 1989

O Ö. Landesgalerie Linz: Künstler-Symposium, Zeichnung und Medium Fotokopie Gmunden 1997. Weitra 1997

O Ö. Landesgalerie: Zwischenbilder Zwischenräume, Kopigrafische und elektroprafische Arbeiten österreichischer Künstler. Linz 1994

Urbons, Klaus: Copy Art, Kunst und Design mit dem Fotokopierer. Köln 1991

VERWENDETE WEBSITES

www.123dapp.com

www.3dsystems.com

blendercam.blogspot.co.at

www.concept-laser.de

www.david-3d.com

www.eduardspoerri.ch

www.faro.com/products/metrology/faro-scanarm/overview

formlabs.com

github.com/grbl/grbl/wiki

www.gramaziokohler.arch.ethz.ch/web/d/lehre/81.html

Abb. 64 Angewandte Technik: Realkopie Blow-Up

www.lytro.com/illum

medusa.disneyresearch.com

www.nikonmetrology.com/en_EU/Products/Laser-Scanning/Hand-held-scanning/ModelMaker-MMDx

opencv-python-tutroals.readthedocs.org/en/latest/py_tutorials/py_tutorials.html

picamera.readthedocs.org/en/release-1.8/index.html

python-gtk-3-tutorial.readthedocs.org/en/latest

www.raspberrypi.org

www.renishaw.com/en/omp400-compact-high-accuracy-touch-probe--6089

www.sparkfun.com/products/12779

www.stratasys.com

www.voxeljet.de

www.zf-laser.com

ABBILDUNGSVERZEICHNIS

Abb. 01 http://i.ytimg.com/vi/VTJq9Z5g4Jk/maxresdefault.jpg

Abb. 02 http://www.disneyresearch.com/wp-content/uploads/Project_Medusa_setup1.png © Copyright 2014 Disney

Abb. 03 http://gramaziokohler.arch.ethz.ch/data/ProjectImages/02_Web/M/040/060333_040_BilderDerDreiWaende_ML_049_WM.jpg

Abb. 04 David Schwärzler

Abb. 05 http://3druck.com/wp-content/uploads/2012/03/2_Architekturmodell-Voxeljet.jpg

Abb. 06 http://www.concept-laser.de/fileadmin/branchenbilder/luftfahrt/luftfahrt_brennkammer.jpg

Abb. 07 Anton Schwärzler

Abb. 08 http://www.stratasys.com/3d-printers/production-series/~/media/Image%20Gallery/900mc_parts_on_tray.jpg?bc=White&as=0&h=380&w=650

Abb. 09 http://usglobalimages.stratasys.com/Image%20Gallery/Objet1000_BMW_F_15_Air_Gate_Back_Dark_Background.jpg?v=635635627865393335

Abb. 10 David Schwaerzler

Abb. 11 http://formlabs.com/media/upload/Chase_Me_girl.jpg

Abb. 12 http://www.3dsystems.com/sites/www.3dsystems.com/files/styles/ddd_printer-gallery_xl/public/prox-950-v10-quickcast.jpg

Abb. 13, 16 David Schwärzler

Abb. 14, 15, 17 Leonie Schwärzler

Abb. 18 http://media.vam.ac.uk/media/thira/collection_images/2006AE/2006AE8921_jpg_l.jpg, © Victoria and Albert Museum, London

Abb. 19 http://3.bp.blogspot.com/-faDitbzXHGE/Ut0mAPaz9lI/AAAAAAAAAN8/TZLMwLGZv-I/s1600/barbara+Astman+Myra+1977.jpg

Abb. 20 http://ccca.concordia.ca/c/images/big/a/astman/ast016.jpg

Abb. 21 http://www.obeygiant.com/images/free/stickers/icon.pdf

Abb. 22 http://upload.wikimedia.org/wikipedia/en/7/75/AndreTheGiantSticker.gif

Abb. 23 Huemer, Peter: Image in Motion, Arbeiten mit dem Medium Fotokopie 1987-2002. Klagenfurt 2002

Abb. 24 O Ö. Landesgalerie: Zwischenbilder Zwischenräume, Kopigrafische und elektroprafische Arbeiten österreichischer Künstler. Linz 1994, S. 91

Abb. 25, 26 Urbons, Klaus: Copy Art, Kunst und Design mit dem Fotokopierer. Köln 1991, S.123, 101

Abb. 27 http://images.metmuseum.org/CRDImages/es/web-large/144867.jpg

Abb. 28 http://images.metmuseum.org/CRDImages/eg/web-large/DP226021.jpg

Abb. 29 http://images.metmuseum.org/CRDImages/es/web-large/ES4895.jpg

Abb. 30 – 64 David Schwärzler

SCANN-SOFTWARE

geschrieben in Python

```
#!/usr/bin/python
from gi.repository import Gtk, Gio, GObject, Gdk, GdkPixbuf
import time, sys, math, getpass, telnetlib, subprocess, datetime, glob, threading, serial
import cv2
import numpy as np
GObject.threads_init()


settings = Gtk.Settings.get_default()
#settings.set_property("gtk-application-prefer-dark-theme", True)

class WorkerThread(threading.Thread):
    def __init__(self, callback):
        threading.Thread.W__init__(self)
        self.callback = callback
    def run(self):
        if index3 == 0: #1.5mm
            movementSpeed = 3780
            resX = 134
            resZ = 134
            resY = 0.66666666
        elif index3 == 1: #1mm
            movementSpeed = 2520
            resX = 200
            resZ = 200
            resY = 1
        elif index3 == 2: #1/2mm
            movementSpeed = 1260
            resX = 400
            resZ = 400
            resY = 2
        elif index3 == 3: #1/3mm
            movementSpeed = 840
            resX = 600
            resZ = 600
            resY = 3
        elif index3 == 4: #1/4mm
            movementSpeed = 630
            resX = 800
            resZ = 800
            resY = 4
        index1float = index1
        scannAreaX = int((float(index1float)+1)*200)
        index2float = index2
        scannAreaY = int((float(index2float)+1)*100)
        index4float = index4
        offsetX = int(200/(float(index4float)*2))
        numberOfSteps = (scannAreaX / offsetX) +1
        scannAreaYfloat = scannAreaY
        movementSpeedfloat = movementSpeed
        scannStripTime = int((float(scannAreaYfloat) * 60000) / float(movementSpeedfloat))+500
        scannMovementTime = scannStripTime * (numberOfSteps * 2)
        scannMovementTime = int(((float(scannMovementTime) / 60000) *10) +0.5) /10.0
        print("Scann Started:\n Scann Time:\t\t"+str(scannMovementTime)+"min \n
AreaX:\t\t\t"+str(scannAreaX)+"mm \n AreaY:\t\t\t"+str(scannAreaY)+"mm \n Movement
Speed:\t"+str(movementSpeed)+"mm/min \n Offset:\t\t"+str(offsetX)+"mm \n Number of
Steps:\t"+str(numberOfSteps)+" \n Scann Strip Time:\t"+str(scannStripTime)+"ms \n Scann
Name:\t\t"+str(entry_text))
        subprocess.call(['mkdir', './home/david/Dropbox/diplom/PiScan/"+str(entry_text)])
        processFile = open ('./home/david/Dropbox/diplom/PiScan/"+str(entry_text)+"/process.sh', 'w')
        processFile.write ("#!/bin/bash\npython /home/david/Dropbox/diplom/PiControl/
makeMeshIntensityGrayMultiThreadNumpy.py "+str(resX)+" "+str(resY)+" "+str(resZ)+"
"+str(offsetX)+" "+str(numberOfSteps)+" "+str(entry_text)+" "+str(scannAreaX)+"
"+str(scannAreaY))
        processFile.close()
        subprocess.call(["chmod +x /home/david/Dropbox/diplom/PiScan/"+str(entry_text)+"/process.
sh"], shell=True)
        start = time.time()

        ## establish telnet connection with linuxcnc

#       host = "128.130.120.120"
#       port = "5007"
#
#       tn = telnetlib.Telnet(host, port)
#       tn.set_debuglevel(9999)
#       time.sleep(1.0)
#       tn.write("hello EMC user-typing-at-telnet 1.0\r\n")
#       time.sleep(1.0)
#       tn.write("set enable EMCTOO\r\n")
#       time.sleep(1.0)
#       tn.write("set mode mdi\r\n")
#       tn.read_until("set mode mdi")
#       tn.write("set set_wait done\r\n")
#       tn.read_until("set set_wait done")

        ## start scann process
#       tn.write("set mdi g0z0\r\n")
##      tn.read_until("set mdi g0z0")
        subprocess.Popen(["ssh pi@10.42.0.100 ,python PiScan/exportSerial.py /dev/ttyUSB0'"],
shell=True)
        # Open grbl serial port
        time.sleep(2)
        ser = serial.serial_for_url("socket://10.42.0.100:7777", timeout=0.5)

        # Wake up grbl
        ser.write("$\n")
        time.sleep(2)   # Wait for grbl to initialize
        grbl_out = ser.readline()

        while grbl_out != "":
            grbl_out = ser.readline() # Wait for grbl response with carriage return
            print , : , + grbl_out.strip()

        ser.write('g1f'+str(movementSpeed)+'x0y0\n') # Send g-code block to grbl
        grbl_out = ser.readline()
        print , : , + grbl_out.strip()
        while grbl_out != "":
            grbl_out = ser.readline()
        while 'Idle' not in grbl_out:
            while grbl_out != "":
                grbl_out = ser.readline()
            ser.write('?\n')
            time.sleep(2)
            grbl_out = ser.readline()
            print , : , + grbl_out.strip()

        for s0 in range(0, numberOfSteps):
            ser.write("g0x"+str(offsetX * s0)+"y0\n")
            grbl_out = ser.readline()
            print , : , + grbl_out.strip()
            while grbl_out != "":
                grbl_out = ser.readline()
            while 'Idle' not in grbl_out:
                while grbl_out != "":
                    grbl_out = ser.readline()
                ser.write('?\n')
                time.sleep(2)
                grbl_out = ser.readline()
                print , : , + grbl_out.strip()
            subprocess.Popen(["ssh pi@10.42.0.100 ,gpio mode 7 out; gpio write 7 1'"], shell=True)
            subprocess.call(["ssh pi@10.42.0.101 ,gpio mode 7 out; gpio write 7 1'"], shell=True)
            ser.write("g1y"+str(scannAreaY+20)+"\n")
            p1 = subprocess.Popen(["ssh pi@10.42.0.100 ,raspivid -w 1292 -h 972 -fps 42 -ISO 100 -ss 23000 -awb
off -awbg 1.5,1.2 -t "+str(scannStripTime)+" -o /home/pi/PiScan/"+str(entry_text)+"/100vid0"+str(s0)+".h264;
gpio write 7 0'"], shell=True)
            subprocess.call(["ssh pi@10.42.0.101 ,raspivid -w 1292 -h 972 -fps 42 -ISO 100 -ss 23000 -awb off
-awbg 1.5,1.2 -t "+str(scannStripTime)+" -o /home/pi/PiScan/"+str(entry_text)+"/101vid0"+str(s0)+".h264;
gpio write 7 0'"], shell=True)
            p1.wait()
            p2 = subprocess.Popen(["python /home/david/Dropbox/diplom/PiControl/makeImageGrayStripes.py
"+str(resX)+" "+str(resY)+" "+str(resZ)+" "+str(offsetX)+" "+str(numberOfSteps)+" "+str(entry_text)+"
"+str(scannAreaX)+" "+str(scannAreaY)+" 0 "+str(s0)], shell=True)
            p3 =subprocess.Popen(["python /home/david/Dropbox/diplom/PiControl/makeImageGrayStripes.py
"+str(resX)+" "+str(resY)+" "+str(resZ)+" "+str(offsetX)+" "+str(numberOfSteps)+" "+str(entry_text)+"
"+str(scannAreaX)+" "+str(scannAreaY)+" 1 "+str(s0)], shell=True)
            ser.write("g0x0y0\n")
            p2.wait()
            p3.wait()
            ser.close()
            subprocess.call(["python /home/david/Dropbox/diplom/PiControl/makeImageGrayCombine.py
"+str(resX)+" "+str(resY)+" "+str(resZ)+" "+str(offsetX)+" "+str(numberOfSteps)+" "+str(entry_text)+"
"+str(scannAreaX)+" "+str(scannAreaY)], shell=True)

            curr = time.time()
            delta = curr-start
            print(delta)
            #time.sleep(1)
            GObject.idle_add(self.callback)


class listboxWindow(Gtk.Window):

    def __init__(self):
```

```
        Gtk.Window.__init__(self, title="David's ScannBooth")
        self.set_border_width(0)
        self.set_default_size(600,700)

        hb = Gtk.HeaderBar()
        hb.set_show_close_button(True)
        hb.props.title = „David's ScannBooth"
        hb.set_subtitle(„easy tile creation")
        self.set_titlebar(hb)

        box1 = Gtk.Box(orientation=Gtk.Orientation.HORIZONTAL)
        self.buttonNew = Gtk.Button(„+ New")
        self.buttonNew.connect(„clicked", self.on_new_clicked)
        box1.add(self.buttonNew)
        hb.pack_start(box1)

        box2 = Gtk.Box(orientation=Gtk.Orientation.HORIZONTAL)
        self.scannboothSettings = Gtk.Button.new_from_icon_name('preferences-system-symbolic', 2)
        self.scannboothSettings.connect(„clicked", self.on_scannboothSettings_clicked)
        self.scannboothSettingsState = 0
        self.buttonShow3D = Gtk.Button(„Show 3D >")
        self.buttonShow3D.connect(„clicked", self.on_show3d_clicked)
        box2.add(self.buttonShow3D)
        box2.add(self.scannboothSettings)
        hb.pack_end(box2)

        self.h2box = Gtk.Box(orientation=Gtk.Orientation.HORIZONTAL, spacing=3)
        hbox = Gtk.Box(orientation=Gtk.Orientation.VERTICAL)
        hbox.set_border_width(0)
        self.add(self.h2box)
        self.h2box.add(hbox)
        self.imageBox = Gtk.Box()
#       self.imageBox.set_relief(Gtk.ReliefStyle.NONE)
        self.imageBox.set_size_request(600,600)
        pixbuf = GdkPixbuf.Pixbuf.new_from_file_at_size('/home/david/Dropbox/diplom/PiControl/Effects/
GrayCombined.png', 600, 600)
        self.image = Gtk.Image.new_from_pixbuf(pixbuf)
#       image = Gtk.Image.new_from_stock(Gtk.STOCK_OPEN,6)
        self.imageBox.add(self.image)
        hbox.pack_start(self.imageBox, True, True, 0)

        effectButtons = Gtk.Box(orientation=Gtk.Orientation.HORIZONTAL, spacing=0)
        hbox.pack_end(effectButtons,False,False, 0)
        convex = Gtk.ToggleButton("convex")
        convex.set_size_request(100,100)
        convex.connect("toggled", self.on_convex_clicked)
        self.convexState = 0
        concave = Gtk.ToggleButton("concave")
        concave.set_size_request(100,100)
        concave.connect("toggled", self.on_concave_clicked)
        self.concaveState = 0
        frame = Gtk.ToggleButton("frame")
        frame.set_size_request(100,100)
        frame.connect("toggled", self.on_frame_clicked)
        self.frameState = 0
        invertAll = Gtk.ToggleButton("invert all")
        invertAll.set_size_request(100,100)
        invertAll.connect("toggled", self.on_invertAll_clicked)
        self.invertAllState = 0
        addImage1 = Gtk.ToggleButton("ad image 1")
        addImage1.set_size_request(100,100)
        addImage1.connect("toggled", self.on_addImage1_clicked)
        self.addImage1State = 0
        addImage2 = Gtk.ToggleButton("ad image 2")
        addImage2.set_size_request(100,100)
        addImage2.connect("toggled", self.on_addImage2_clicked)
        self.addImage2State = 0
        overlayImage = Gtk.ToggleButton("overlay image")
        overlayImage.set_size_request(100,100)
        overlayImage.connect("toggled", self.on_overlayImage_clicked)
        self.overlayImageState = 0
        effectButtons.pack_start(Gtk.Label(,'), True, False,0)
        effectButtons.pack_start(convex, False, False,0)
#       effectButtons.pack_start(concave, False, False,0)
        effectButtons.pack_start(frame, False, False,0)
        effectButtons.pack_start(invertAll, False, False,0)
        effectButtons.pack_start(overlayImage, False, False,0)
        effectButtons.pack_start(addImage1, False, False,0)
        effectButtons.pack_start(addImage2, False, False,0)
        effectButtons.pack_start(Gtk.Label(,'), True, False,0)

        self.fileName = '/home/david/Dropbox/diplom/PiControl/Effects/GrayCombined.png'
        self.folderName = '/home/david/Dropbox/diplom/PiControl/Effects'
        self.mainImageBig = cv2.imread(self.fileName, -1)
        self.mainImage = cv2.resize(self.mainImageBig, (1200,1200))
        self.mainImageZscale = cv2.resize(self.mainImageBig, (1200,1200))
        self.baseRows, self.baseCols = self.mainImage.shape
        self.ad1state = 100
        self.ad2state = 0
        self.ad1astate = 0
        self.ad3state = 100
        self.ad4state = 0
        self.ad3astate = 0
        self.ad5state = 100
        self.ad6state = 0
        self.ad5astate = 0
        self.ad7state = 1
        self.scale7val = 1
        self.on_new_clicked(1)

    def on_show3d_clicked(self,widget):

        subprocess.call([,/home/david/Dropbox/diplom/PiControl/blender-CAM_0.8.0/blender', ,'+str(self.
folderName)+'/Tile/kachel.blend'])

    def on_convex_clicked(self,convex):
#       self.convex = cv2.imread('/home/david/Dropbox/diplom/PiControl/Effects/convex1.png',-1)
        if self.convexState == 0:
            self.convexState = 1
            start = time.time()
            self.convex = np.zeros((self.baseRows, self.baseCols), np.uint16)
            for r0 in range(0,self.baseRows):
                for t0 in range(0,self.baseCols):
#                   self.convex[r0,t0] = (((((((self.baseRows*1.5)**2)-((self.
baseRows/2.0)**2))**(0.5))*(-1))+((((self.baseRows*1.5)**2)-((r0-(self.
baseRows/2.0))**2))**(0.5))) + (((((self.baseCols*1.5)**2)-((self.baseCols/2.0)**2))**(0.5))*(-
1))+((((self.baseCols*1.5)**2)-((t0-(self.baseCols/2.0))**2))**(0.5)))/2) * 150
                    self.convex.itemset((r0,t0), (((((((self.baseRows*1.5)**2)-((self.
baseRows/2.0)**2))**(0.5))*(-1))+((((self.baseRows*1.5)**2)-((r0-(self.
baseRows/2.0))**2))**(0.5))) + (((((self.baseCols*1.5)**2)-((self.baseCols/2.0)**2))**(0.5))*(-
1))+((((self.baseCols*1.5)**2)-((t0-(self.baseCols/2.0))**2))**(0.5)))/2) * 150)
            curr = time.time()
            delta = curr-start
            print(delta)
        elif self.convexState == 1:
            self.convexState = 2
        else:
            self.convexState = 1
        self.combineImages()


    def on_concave_clicked(self,concave):
#       self.concave = cv2.imread('/home/david/Dropbox/diplom/PiControl/Effects/convex1.png',-1)
        if self.concaveState == 0:
            self.concaveState = 1
            self.concave = np.zeros((self.baseRows, self.baseCols), np.uint16)
            for r0 in range(0,self.baseRows):
                for t0 in range(0,self.baseCols):

                    self.concave[r0,t0] = (((self.baseRows*1.5)-((((self.baseRows*1.5)**2)-((r0-
(self.baseRows/2.0))**2))**(0.5))) + (((self.baseCols*1.5)-((((self.baseCols*1.5)**2)-((t0-(self.
baseCols/2.0))**2))**(0.5))))/2) * 150
        elif self.concaveState == 1:
            self.concaveState = 2
        else:
            self.concaveState = 1
        self.combineImages()

    def on_frame_clicked(self,frame):
#       self.frame = cv2.imread('/home/david/Dropbox/diplom/PiControl/Effects/convex1.png',-1)
        if self.frameState == 0:
            self.frameState = 1
            self.frame = np.zeros((self.baseRows, self.baseCols), np.uint16)
            self.frame[:] = 32768
            for r0 in range(0,int(self.baseRows/20.0)):
                solidColor = np.zeros(((self.baseRows-(2*(r0+int(self.baseRows/40.0)))), (self.baseCols-
(2*(r0+int(self.baseRows/40.0))))), np.uint16)
                solidColor[:] = (32768 - (r0 * int(32768/(self.baseRows/20.0))))
                self.frame[(r0+int(self.baseRows/40.0)):((self.baseRows-r0)-int(self.baseRows/40.0)),(r0+int(self.
baseRows/40.0)):((self.baseCols-r0)-int(self.baseRows/40.0))] = solidColor
        elif self.frameState == 1:
            self.frameState = 2
        else:
            self.frameState = 1
```

```
        self.combineImages()

    def on_invertAll_clicked(self,invertAll):
        if self.invertAllState == 0:
            self.invertAllState = 1
        elif self.invertAllState == 1:
            self.invertAllState = 2
        else:
            self.invertAllState = 1
        self.combineImages()

    def on_addImage1_clicked(self,addImage1):
        if self.addImage1State == 0:
            self.addImage1State = 1
            dialog = Gtk.FileChooserDialog(„Please open a Grayscale Image“, self, Gtk.FileChooserAction.
OPEN, (Gtk.STOCK_CANCEL, Gtk.ResponseType.CANCEL, Gtk.STOCK_OPEN, Gtk.ResponseType.OK))
            dialog.set_current_folder('/home/david/Dropbox/diplom/PiScan')
            self.preview = Gtk.Image()
            dialog.set_preview_widget(self.preview)
            dialog.connect(„update-preview“, self.update_preview, self.preview)
            self.add_filters(dialog)

            response = dialog.run()
            if response == Gtk.ResponseType.OK:
                self.fileName2 = dialog.get_filename()
                self.addImage1Image = cv2.imread(self.fileName2, -1)
                self.addImage1ImageR = cv2.resize(self.addImage1Image, (self.baseCols,self.baseRows))
                self.addImage1ImageZscale = cv2.resize(self.addImage1Image, (self.baseCols,self.baseRows))
            elif response == Gtk.ResponseType.CANCEL:
                self.addImage1State = 0
            else:
                self.addimage1State = 3
            dialog.destroy()
        else:
            self.addImage1State = 0
        self.combineImages()

    def on_addImage2_clicked(self,addImage2):
        if self.addImage2State == 0:
            self.addImage2State = 1
            dialog = Gtk.FileChooserDialog(„Please open a Grayscale Image“, self, Gtk.FileChooserAction.
OPEN, (Gtk.STOCK_CANCEL, Gtk.ResponseType.CANCEL, Gtk.STOCK_OPEN, Gtk.ResponseType.OK))
            dialog.set_current_folder('/home/david/Dropbox/diplom/PiScan')
            self.preview = Gtk.Image()
            dialog.set_preview_widget(self.preview)
            dialog.connect(„update-preview“, self.update_preview, self.preview)
            self.add_filters(dialog)

            response = dialog.run()
            if response == Gtk.ResponseType.OK:
                self.fileName3 = dialog.get_filename()
                self.addImage2Image = cv2.imread(self.fileName3, -1)
                self.addImage2ImageR = cv2.resize(self.addImage2Image, (self.baseCols,self.baseRows))
                self.addImage2ImageZscale = cv2.resize(self.addImage2Image, (self.baseCols,self.baseRows))
            elif response == Gtk.ResponseType.CANCEL:
                self.addImage2State = 0
            else:
                self.addimage2State = 3
            dialog.destroy()
        else:
            self.addImage2State = 0
        self.combineImages()

    def on_overlayImage_clicked(self,overlayImage):
        if self.overlayImageState == 0:
            self.overlayImageState = 1
            dialog = Gtk.FileChooserDialog(„Please open a Grayscale Image“, self, Gtk.FileChooserAction.
OPEN, (Gtk.STOCK_CANCEL, Gtk.ResponseType.CANCEL, Gtk.STOCK_OPEN, Gtk.ResponseType.OK))
            dialog.set_current_folder('/home/david/Dropbox/diplom/PiScan')
            self.preview = Gtk.Image()
            dialog.set_preview_widget(self.preview)
            dialog.connect(„update-preview“, self.update_preview, self.preview)
            self.add_filters(dialog)

            response = dialog.run()
            if response == Gtk.ResponseType.OK:
                self.fileName4 = dialog.get_filename()
                self.overlayImage = cv2.imread(self.fileName4, -1)
                self.overlayImageR = cv2.resize(self.overlayImage, (self.baseCols,self.baseRows))
            elif response == Gtk.ResponseType.CANCEL:
                self.overlayImageState = 0
            else:
                self.overlayImageState = 3
            dialog.destroy()
        else:
            self.overlayImageState = 0
        self.combineImages()

    def combineImages(self):
        self.combinedImage = (0+self.mainImageZscale)
        if self.addImage1State == 1:
            self.combinedImage = np.fmax(self.combinedImage, self.addImage1ImageZscale)
#           for r0 in range(0,self.baseRows):
#               for t0 in range(0,self.baseCols):
#                   val1 = self.addImage1ImageZscale[r0,t0]
#                   val0 = self.combinedImage[r0,t0]
#                   if val1 > val0:
#                       self.combinedImage[r0,t0] = val1
#                   else:
#                       self.combinedImage[r0,t0] = val0
        if self.addImage2State == 1:
            self.combinedImage = np.fmax(self.combinedImage, self.addImage2ImageZscale)
#           for r0 in range(0,self.baseRows):
#               for t0 in range(0,self.baseCols):
#                   val1 = self.addImage2ImageZscale[r0,t0]
#                   val0 = self.combinedImage[r0,t0]
#                   if val1 > val0:
#                       self.combinedImage[r0,t0] = val1
#                   else:
#                       self.combinedImage[r0,t0] = val0
        if self.convexState == 1:
            self.convex=np.float64(self.convex)
            self.combinedImage = np.add(self.combinedImage, self.convex)
            self.combinedImage[self.combinedImage > 65535] = 65535
            self.combinedImage[self.combinedImage < 0] = 0
#           for r0 in range(0,self.baseRows):
#               for t0 in range(0,self.baseCols):
#                   val1 = self.convex[r0,t0]
#                   val0 = self.combinedImage[r0,t0]
#                   self.combinedImage[r0,t0] = val1+val0
        if self.concaveState == 1:
            self.combinedImage = np.add(self.combinedImage, self.concave)
            self.combinedImage[self.combinedImage > 65535] = 65535
            self.combinedImage[self.combinedImage < 0] = 0
#           for r0 in range(0,self.baseRows):
#               for t0 in range(0,self.baseCols):
#                   val1 = self.concave[r0,t0]
#                   val0 = self.combinedImage[r0,t0]
#                   if val1 > val0:
#                       self.combinedImage[r0,t0] = val1
#                   else:
#                       self.combinedImage[r0,t0] = val0
        if self.overlayImageState ==1:
            self.overlayImageZscaled = np.rint(self.overlayImageR*(self.scale7val/200.0))
            self.combinedImage = np.add(self.combinedImage, self.overlayImageZscaled)
            self.combinedImage[self.combinedImage > 65535] = 65535
            self.combinedImage[self.combinedImage < 0] = 0
#           for r0 in range(0,self.baseRows):
#               for t0 in range(0,self.baseCols):
#                   val1 = self.overlayImageR[r0,t0]
#                   val2 = self.combinedImage[r0,t0]
#                   val1 = int(((val1/200.0)*self.scale7val)+val2)
#                   if val1 > 65535:
#                       self.combinedImage[r0,t0] = 65535
#                   elif val1 < 0:
#                       self.combinedImage[r0,t0] = 0
#                   else:
#                       self.combinedImage[r0,t0] = val1
        if self.frameState == 1:
            self.combinedImage = np.fmax(self.combinedImage, self.frame)
#           for r0 in range(0,self.baseRows):
#               for t0 in range(0,self.baseCols):
#                   val1 = self.frame[r0,t0]
#                   val0 = self.combinedImage[r0,t0]
#                   if val1 > val0:
#                       self.combinedImage[r0,t0] = val1
#                   else:
#                       self.combinedImage[r0,t0] = val0

        if self.invertAllState == 1:
            self.combinedImage = (65535-self.combinedImage)

        self.combinedImage = np.uint16(self.combinedImage)
        self.combinedImageInvert =  np.uint16(65535-self.combinedImage)
```

```
        self.combinedImageInvert = np.flipud(self.combinedImageInvert)
        cv2.imwrite(self.folderName+'/Tile/GrayCombined.png', self.combinedImage)
        cv2.imwrite(self.folderName+'/Tile/GrayCombinedInvert.png', self.combinedImageInvert)
        self.imageBox.remove(self.image)
        pixbuf = GdkPixbuf.Pixbuf.new_from_file_at_size(self.folderName+'/Tile/GrayCombined.png',
600, 600)
        self.image = Gtk.Image.new_from_pixbuf(pixbuf)
        self.imageBox.add(self.image)
        self.imageBox.show_all()



    def on_new_clicked(self, widget):
        dialog = Gtk.MessageDialog(self, 0, Gtk.MessageType.INFO, (,new scann', 1, ,open previous', 2),
„start new scann or open previous" )
        dialog.format_secondary_text(„You can either choose to \nopen a previous Scann \nor start a new
one")
        dialog.set_size_request(300,200)
        response = dialog.run()
        if response == 1:
            dialog.destroy()
            self.on_newScann_clicked(1)
        elif response == 2:
            dialog.destroy()
            self.on_openPrevious_clicked(1)
        dialog.destroy()

    def on_scannboothSettings_clicked(self, widget):
        if self.scannboothSettingsState ==0:
            self.scannboothSettingsState =1
            self.mainbox = Gtk.Box(orientation=Gtk.Orientation.VERTICAL)
            self.mainbox.set_size_request(300,300)
            box = Gtk.Box(orientation=Gtk.Orientation.VERTICAL)

            label = Gtk.Label(,add image 1')
            box.add(label)
            modExternImage1 = Gtk.Button(,modify external')
            modExternImage1.connect(„clicked",self.on_modExternImage1_clicked)
            box.add(modExternImage1)
            hbox = Gtk.Box(orientation=Gtk.Orientation.HORIZONTAL, spacing=5)
            ad1 = Gtk.Adjustment(self.ad1state,0,200,10)
            self.scale1 = Gtk.Scale(orientation=Gtk.Orientation.HORIZONTAL, adjustment=ad1)
            self.scale1.set_digits(0)
            self.scale1Button = Gtk.Button(,set')
            self.scale1Button.connect(„clicked", self.on_scaleButton_clicked)
            label = Gtk.Label(,Z scale')
            hbox.pack_start(label, False, False, 0)
            hbox.pack_start(self.scale1, True, True, 0)
            hbox.pack_end(self.scale1Button, False, False,0)
            box.add(hbox)

            hbox = Gtk.Box(orientation=Gtk.Orientation.HORIZONTAL, spacing=5)
            ad2 = Gtk.Adjustment(self.ad2state,-100,100,10)
            self.scale2 = Gtk.Scale(orientation=Gtk.Orientation.HORIZONTAL, adjustment=ad2)
            self.scale2.set_digits(0)
            self.scale2Button = Gtk.Button(,set')
            self.scale2Button.connect(„clicked", self.on_scaleButton_clicked)
            label = Gtk.Label(,Z offset')
            hbox.pack_start(label, False, False, 0)
            hbox.pack_start(self.scale2, True, True, 0)
            hbox.pack_end(self.scale2Button, False, False,0)
            box.add(hbox)

            hbox = Gtk.Box(orientation=Gtk.Orientation.HORIZONTAL, spacing=5)
            ad1a = Gtk.Adjustment(self.ad1astate,0,200,1)
            self.scale1a = Gtk.Scale(orientation=Gtk.Orientation.HORIZONTAL, adjustment=ad1a)
            self.scale1a.set_digits(0)
            self.scale1aButton = Gtk.Button(,set')
            self.scale1aButton.connect(„clicked", self.on_scaleButton_clicked)
            label = Gtk.Label(,Z threshold')
            hbox.pack_start(label, False, False, 0)
            hbox.pack_start(self.scale1a, True, True, 0)
            hbox.pack_end(self.scale1aButton, False, False,0)
            box.add(hbox)

            label = Gtk.Label(,add image 2')
            box.add(label)
            modExternImage2 = Gtk.Button(,modify external')
            modExternImage2.connect(„clicked",self.on_modExternImage2_clicked)
            box.add(modExternImage2)
            hbox = Gtk.Box(orientation=Gtk.Orientation.HORIZONTAL, spacing=5)
            ad3 = Gtk.Adjustment(self.ad3state,0,200,10)
            self.scale3 = Gtk.Scale(orientation=Gtk.Orientation.HORIZONTAL, adjustment=ad3)
            self.scale3.set_digits(0)
            self.scale3Button = Gtk.Button(,set')
            self.scale3Button.connect(„clicked", self.on_scaleButton2_clicked)
            label = Gtk.Label(,Z scale')
            hbox.pack_start(label, False, False, 0)
            hbox.pack_start(self.scale3, True, True, 0)
            hbox.pack_end(self.scale3Button, False, False,0)
            box.add(hbox)

            hbox = Gtk.Box(orientation=Gtk.Orientation.HORIZONTAL, spacing=5)
            ad4 = Gtk.Adjustment(self.ad4state,-100,100,10)
            self.scale4 = Gtk.Scale(orientation=Gtk.Orientation.HORIZONTAL, adjustment=ad4)
            self.scale4.set_digits(0)
            self.scale4Button = Gtk.Button(,set')
            self.scale4Button.connect(„clicked", self.on_scaleButton2_clicked)
            label = Gtk.Label(,Z offset')
            hbox.pack_start(label, False, False, 0)
            hbox.pack_start(self.scale4, True, True, 0)
            hbox.pack_end(self.scale4Button, False, False,0)
            box.add(hbox)

            hbox = Gtk.Box(orientation=Gtk.Orientation.HORIZONTAL, spacing=5)
            ad3a = Gtk.Adjustment(self.ad3astate,0,200,1)
            self.scale3a = Gtk.Scale(orientation=Gtk.Orientation.HORIZONTAL, adjustment=ad3a)
            self.scale3a.set_digits(0)
            self.scale3aButton = Gtk.Button(,set')
            self.scale3aButton.connect(„clicked", self.on_scaleButton2_clicked)
            label = Gtk.Label(,Z threshold')
            hbox.pack_start(label, False, False, 0)
            hbox.pack_start(self.scale3a, True, True, 0)
            hbox.pack_end(self.scale3aButton, False, False,0)
            box.add(hbox)

            label = Gtk.Label(,base image')
            box.add(label)
            modExternImage3 = Gtk.Button(,modify external')
            modExternImage3.connect(„clicked",self.on_modExternImage3_clicked)
            box.add(modExternImage3)
            hbox = Gtk.Box(orientation=Gtk.Orientation.HORIZONTAL, spacing=5)
            ad5 = Gtk.Adjustment(self.ad5state,0,200,10)
            self.scale5 = Gtk.Scale(orientation=Gtk.Orientation.HORIZONTAL, adjustment=ad5)
            self.scale5.set_digits(0)
            self.scale5Button = Gtk.Button(,set')
            self.scale5Button.connect(„clicked", self.on_scaleButton3_clicked)
            label = Gtk.Label(,Z scale')
            hbox.pack_start(label, False, False, 0)
            hbox.pack_start(self.scale5, True, True, 0)
            hbox.pack_end(self.scale5Button, False, False,0)
            box.add(hbox)

            hbox = Gtk.Box(orientation=Gtk.Orientation.HORIZONTAL, spacing=5)
            ad6 = Gtk.Adjustment(self.ad4state,-100,100,10)
            self.scale6 = Gtk.Scale(orientation=Gtk.Orientation.HORIZONTAL, adjustment=ad6)
            self.scale6.set_digits(0)
            self.scale6Button = Gtk.Button(,set')
            self.scale6Button.connect(„clicked", self.on_scaleButton3_clicked)
            label = Gtk.Label(,Z offset')
            hbox.pack_start(label, False, False, 0)
            hbox.pack_start(self.scale6, True, True, 0)
            hbox.pack_end(self.scale6Button, False, False,0)
            box.add(hbox)

            hbox = Gtk.Box(orientation=Gtk.Orientation.HORIZONTAL, spacing=5)
            ad5a = Gtk.Adjustment(self.ad5astate,0,200,1)
            self.scale5a = Gtk.Scale(orientation=Gtk.Orientation.HORIZONTAL, adjustment=ad5a)
            self.scale5a.set_digits(0)
            self.scale5aButton = Gtk.Button(,set')
            self.scale5aButton.connect(„clicked", self.on_scaleButton3_clicked)
            label = Gtk.Label(,Z threshold')
            hbox.pack_start(label, False, False, 0)
            hbox.pack_start(self.scale5a, True, True, 0)
            hbox.pack_end(self.scale5aButton, False, False,0)
            box.add(hbox)

            label = Gtk.Label(,overlay image')
            box.add(label)
            modExternImage4 = Gtk.Button(,modify external')
            modExternImage4.connect(„clicked",self.on_modExternImage4_clicked)
            box.add(modExternImage4)
            hbox = Gtk.Box(orientation=Gtk.Orientation.HORIZONTAL, spacing=5)
            ad7 = Gtk.Adjustment(self.ad7state,-50,50,1)
            self.scale7 = Gtk.Scale(orientation=Gtk.Orientation.HORIZONTAL, adjustment=ad7)
```

```
        self.scale7.set_digits(1)
        self.scale7Button = Gtk.Button(,set')
        self.scale7Button.connect(„clicked", self.on_scaleButton4_clicked)
        label = Gtk.Label(,intensity')
        hbox.pack_start(label, False, False, 0)
        hbox.pack_start(self.scale7, True, True, 0)
        hbox.pack_end(self.scale7Button, False, False,0)
        box.add(hbox)

        self.mainbox.add(box)
        self.mainbox.show_all()
        self.h2box.add(self.mainbox)
    else:
        self.h2box.remove(self.mainbox)
        self.scannboothSettingsState = 0
        self.resize(600,700)

def on_modExternImage1_clicked(self,widget):
    cv2.imwrite(self.folderName+'/Tile/addImage1.png', self.addImage1ImageR)
    subprocess.call([,krita', ,'+str(self.folderName)+'/Tile/addImage1.png'])
    self.addImage1ImageR = cv2.imread(self.folderName+'/Tile/addImage1.png', -1)
    self.on_scaleButton_clicked(1)

def on_modExternImage2_clicked(self,widget):
    cv2.imwrite(self.folderName+'/Tile/addImage2.png', self.addImage2ImageR)
    subprocess.call([,krita', ,'+str(self.folderName)+'/Tile/addImage2.png'])
    self.addImage2ImageR = cv2.imread(self.folderName+'/Tile/addImage2.png', -1)
    self.on_scaleButton2_clicked(1)

def on_modExternImage3_clicked(self,widget):
    cv2.imwrite(self.folderName+'/Tile/mainImage.png', self.mainImage)
    subprocess.call([,krita', ,'+str(self.folderName)+'/Tile/mainImage.png'])
    self.mainImage = cv2.imread(self.folderName+'/Tile/mainImage.png', -1)
    self.on_scaleButton3_clicked(1)

def on_modExternImage4_clicked(self,widget):
    cv2.imwrite(self.folderName+'/Tile/overlayImage.png', self.overlayImageR)
    subprocess.call([,krita', ,'+str(self.folderName)+'/Tile/overlayImage.png'])
    self.overlayImageR = cv2.imread(self.folderName+'/Tile/overlayImage.png', -1)
    self.on_scaleButton4_clicked(1)
```



```
def on_scaleButton_clicked(self, widget):
    self.scale1val = self.scale1.get_value()
    self.ad1state = self.scale1val
    self.scale2val = self.scale2.get_value()
    self.ad2state = self.scale2val
    self.scale2val = int((65535/100)*self.scale2val)
    self.scale1aval = self.scale1a.get_value()
    self.ad1astate = self.scale1aval
    self.scale1aval = int((65535/200)*self.scale1aval)

    self.addImage1ImageZscale = np.rint(((self.addImage1ImageR + self.scale2val)/100.0)*self.
scale1val)
    self.addImage1ImageZscale[self.addImage1ImageZscale < 0] = 0
    self.addImage1ImageZscale[self.addImage1ImageZscale < np.rint(((self.scale1aval + self.
scale2val)/100.0)*self.scale1val)] = 0
    self.addImage1ImageZscale[self.addImage1ImageZscale > 65535] = 65535
    self.addImage1ImageZscale = np.uint16(self.addImage1ImageZscale)

    self.combineImages()

def on_scaleButton2_clicked(self, widget):
    self.scale3val = self.scale3.get_value()
    self.ad3state = self.scale3val
    self.scale4val = self.scale4.get_value()
    self.ad4state = self.scale4val
    self.scale4val = np.rint((65535/100)*self.scale4val)
    self.scale3aval = self.scale3a.get_value()
    self.ad3astate = self.scale3aval
    self.scale3aval = np.rint((65535/200)*self.scale3aval)

    self.addImage2ImageZscale = np.rint(((self.addImage2ImageR + self.scale4val)/100.0)*self.
scale3val)
    self.addImage2ImageZscale[self.addImage2ImageZscale < 0] = 0
    self.addImage2ImageZscale[self.addImage2ImageZscale < np.rint(((self.scale3aval + self.
scale4val)/100.0)*self.scale3val)] = 0
    self.addImage2ImageZscale[self.addImage2ImageZscale > 65535] = 65535
    self.addImage2ImageZscale = np.uint16(self.addImage2ImageZscale)

    self.combineImages()

def on_scaleButton3_clicked(self, widget):
    self.scale5val = self.scale5.get_value()
    self.ad5state = self.scale5val
    self.scale6val = self.scale6.get_value()
    self.ad6state = self.scale6val
    self.scale6val = np.rint((65535/100.0)*self.scale6val)
    self.scale5aval = self.scale5a.get_value()
    self.ad5astate = self.scale5aval
    self.scale5aval = np.rint((65535/200.0)*self.scale5aval)

    self.mainImageZscale = np.rint(((self.mainImage + self.scale6val)/100.0)*self.scale5val)
    self.mainImageZscale[self.mainImageZscale < 0] = 0
    self.mainImageZscale[self.mainImageZscale < np.rint(((self.scale5aval + self.
scale6val)/100.0)*self.scale5val)] = 0
    self.mainImageZscale[self.mainImageZscale > 65535] = 65535
    self.mainImageZscale = np.uint16(self.mainImageZscale)

    self.combineImages()


def on_scaleButton4_clicked(self, widget):
    self.scale7val = self.scale7.get_value()
    self.ad7state = self.scale7val
    self.combineImages()

def on_openPrevious_clicked(self,widget):
    dialog = Gtk.FileChooserDialog(„Please open a Grayscale Image", self, Gtk.FileChooserAction.
OPEN, (Gtk.STOCK_CANCEL, Gtk.ResponseType.CANCEL, Gtk.STOCK_OPEN, Gtk.ResponseType.OK))
    dialog.set_current_folder(,/home/david/Dropbox/diplom/PiScan')
    self.preview = Gtk.Image()
    dialog.set_preview_widget(self.preview)
    dialog.connect(„update-preview", self.update_preview, self.preview)
    self.add_filters(dialog)

    response = dialog.run()
    if response == Gtk.ResponseType.OK:
        self.fileName = dialog.get_filename()
        self.folderName = dialog.get_current_folder()
        print(self.folderName)
        subprocess.call([,mkdir', ,'+str(self.folderName)+'/Tile'])
        subprocess.call([,cp', ,/home/david/Dropbox/diplom/PiControl/Effects/Tile/kachel.blend',
,'+str(self.folderName)+'/Tile/kachel.blend'])
    self.imageBox.remove(self.image)
    pixbuf = GdkPixbuf.Pixbuf.new_from_file_at_size(self.fileName, 600, 600)
    self.mainImageBig = cv2.imread(self.fileName, -1)
    self.mainImage = cv2.resize(self.mainImageBig, (1200,1200))
    self.mainImageZscale = cv2.resize(self.mainImageBig, (1200,1200))
    self.baseRows, self.baseCols = self.mainImage.shape
    self.image = Gtk.Image.new_from_pixbuf(pixbuf)
    self.imageBox.add(self.image)
    self.imageBox.show_all()
    dialog.destroy()
    self.combineImages()
elif response == Gtk.ResponseType.CANCEL:
    dialog.destroy()
    self.on_new_clicked(1)
else:
    dialog.destroy()
    self.on_new_clicked(1)

def update_preview(self,dialog,preview):
    filename = dialog.get_preview_filename()
    try:
        pixbuf = GdkPixbuf.Pixbuf.new_from_file_at_size(filename, 300, 300)
        self.preview.set_from_pixbuf(pixbuf)
        have_preview = True
    except:
        have_preview = False
    dialog.set_preview_widget_active(have_preview)
    return


def add_filters(self, dialog):
    filter_png = Gtk.FileFilter()
    filter_png.set_name(„PNG Images")
    filter_png.add_mime_type(„image/png")
    dialog.add_filter(filter_png)

def on_newScann_clicked(self, widget):
    toplevel = self.get_toplevel()
    self.dialog2 = Gtk.Dialog(„New Scann", toplevel, 0)
    self.dialog2.set_default_size(100,100)
    hb = Gtk.HeaderBar()
```

```
        hb.set_show_close_button(True)
        hb.props.title = „David's ScannBooth“
        hb.set_subtitle(„New Scann“)
        self.dialog2.set_titlebar(hb)
        self.switch = Gtk.Switch()
        self.switch.connect(„notify::active“, self.on_switch_activated)
        self.switch.set_active(False)
        self.combo = Gtk.ComboBoxText()
        self.combo.insert(0, „0“, „simple“)
        self.combo.insert(1, „1“, „advanced“)
        self.combo.connect('changed', self.changed_cb)
        hb.pack_start(self.switch)
        hb.pack_end(self.combo)
        hbox = Gtk.Box()
        hbox.set_border_width(0)
        self.dialog2.vbox.pack_start(hbox,True,True,0)
        hbox.show()
        self.listbox = Gtk.ListBox()
        self.listbox.set_selection_mode(Gtk.SelectionMode.NONE)
        hbox.pack_start(self.listbox, True, True, 0)

        emptyRow = Gtk.ListBoxRow()
        emptyHbox = Gtk.Box(orientation=Gtk.Orientation.HORIZONTAL, spacing=0)
        emptyHbox.set_size_request(20,20)

        emptyRow.add(emptyHbox)
        self.listbox.add(emptyRow)

        self.row1 = Gtk.ListBoxRow()
        hbox1 = Gtk.Box(orientation=Gtk.Orientation.HORIZONTAL, spacing=50)
        self.row1.add(hbox1)
        label = Gtk.Label(„name of scann“, xalign = 0)
        self.entry = Gtk.Entry()
        self.entry.connect(„changed“, self.enter_callback)
        now = datetime.datetime.now()
        timeString = now.strftime(„%Y-%m-%d_%H:%M“)
        self.entry.set_text(timeString)
        self.enter_callback(self.entry)
        hbox1.pack_start(label, True, True, 30)
        hbox1.pack_start(self.entry, True, True, 30)
        self.listbox.add(self.row1)
        self.row2 = Gtk.ListBoxRow()
        hbox2 = Gtk.Box(orientation=Gtk.Orientation.HORIZONTAL, spacing=50)
        self.row2.add(hbox2)
        label = Gtk.Label(„dimension of scann area X axis“, xalign=0)
        self.combo1 = Gtk.ComboBoxText()
        self.combo1.insert(0, „0“, „20cm“)
        self.combo1.insert(1, „1“, „40cm“)
#       self.combo1.insert(2, „2“, „60cm“)
        self.combo1.connect('changed', self.changed_cb1)
        self.combo1.set_active(1)
        hbox2.pack_start(label, True, True, 30)
        hbox2.pack_start(self.combo1, False, True, 30)

        self.listbox.add(self.row2)

        self.row3 = Gtk.ListBoxRow()
        hbox3 = Gtk.Box(orientation=Gtk.Orientation.HORIZONTAL, spacing=50)
        self.row3.add(hbox3)
        label = Gtk.Label(„dimension of scann area Y axis“, xalign=0)
        self.combo2 = Gtk.ComboBoxText()
        self.combo2.insert(0, „0“, „10cm“)
        self.combo2.insert(1, „1“, „20cm“)
        self.combo2.insert(2, „2“, „30cm“)
        self.combo2.insert(3, „3“, „40cm“)
        self.combo2.insert(4, „4“, „50cm“)
        self.combo2.insert(5, „5“, „60cm“)
        self.combo2.insert(6, „6“, „70cm“)
#       self.combo2.insert(7, „7“, „80cm“)
#       self.combo2.insert(8, „8“, „90cm“)
#       self.combo2.insert(9, „9“, „100cm“)
#       self.combo2.insert(10, „10“, „110cm“)
#       self.combo2.insert(11, „11“, „120cm“)
#       self.combo2.insert(12, „12“, „130cm“)
#       self.combo2.insert(13, „13“, „140cm“)
        self.combo2.connect('changed', self.changed_cb2)
        self.combo2.set_active(3)
        hbox3.pack_start(label, True, True, 30)
        hbox3.pack_start(self.combo2, False, True, 30)

        self.listbox.add(self.row3)
        self.row4 = Gtk.ListBoxRow()
        hbox4 = Gtk.Box(orientation=Gtk.Orientation.HORIZONTAL, spacing=50)
        self.row4.add(hbox4)
        label = Gtk.Label(„resolution“, xalign=0)
        self.combo3 = Gtk.ComboBoxText()
        self.combo3.insert(0, „0“, „1.5mm“)
        self.combo3.insert(1, „1“, „1mm“)
        self.combo3.insert(2, „2“, „1/2mm“)
        self.combo3.insert(3, „3“, „1/3mm“)
        self.combo3.insert(4, „4“, „1/4mm“)
        self.combo3.connect('changed', self.changed_cb3)
        self.combo3.set_active(2)
        hbox4.pack_start(label, True, True, 30)
        hbox4.pack_start(self.combo3, False, True, 30)

        self.listbox.add(self.row4)

        self.row5 = Gtk.ListBoxRow()
        hbox5 = Gtk.Box(orientation=Gtk.Orientation.HORIZONTAL, spacing=50)
        self.row5.add(hbox5)
        label = Gtk.Label(„scann overlap“, xalign=0)
        self.combo4 = Gtk.ComboBoxText()
        self.combo4.insert(0, „0“, „0%“)
        self.combo4.insert(1, „1“, „50%“)
        self.combo4.insert(2, „2“, „75%“)
        self.combo4.connect('changed', self.changed_cb4)
        self.combo4.set_active(1)
        hbox5.pack_start(label, True, True, 30)
        hbox5.pack_start(self.combo4, False, True, 30)

        self.listbox.add(self.row5)

        emptyRow = Gtk.ListBoxRow()
        emptyHbox = Gtk.Box(orientation=Gtk.Orientation.HORIZONTAL, spacing=0)
        emptyHbox.set_size_request(20,20)
        emptyRow.add(emptyHbox)
        self.listbox.add(emptyRow)
        row7 = Gtk.ListBoxRow()
        hbox7 = Gtk.Box(orientation=Gtk.Orientation.HORIZONTAL, spacing=0)
        row7.add(hbox7)
        self.button = Gtk.Button()
        self.button.set_size_request(50,50)
        self.buttonLabel = Gtk.Label()
        self.buttonLabel.set_markup(„<big>          Start Scann          </big>“)
        self.button.add(self.buttonLabel)
        self.button.connect(„clicked“, self.on_start_scann_clicked)
        hbox7.pack_start(self.button, True, True, 30)

        self.listbox.add(row7)

        emptyRow = Gtk.ListBoxRow()
        emptyHbox = Gtk.Box(orientation=Gtk.Orientation.HORIZONTAL, spacing=0)
        emptyHbox.set_size_request(20,20)
        emptyRow.add(emptyHbox)
        self.listbox.add(emptyRow)
        self.dialog2.show_all()
        self.combo.set_active(0)

        self.hbox6 = Gtk.Box(orientation=Gtk.Orientation.HORIZONTAL, spacing=0)
        label = Gtk.Label(xalign=.5)
        label.set_markup(„<big>scann in progress</big>“)
        spinner = Gtk.Spinner()
        spinner.start()
        self.hbox6.pack_start(label, True, True, 30)
        self.hbox6.pack_start(spinner, True, True, 30)
#       self.startup_info(1)

        self.dialog2.run()
        self.dialog2.destroy()
        subprocess.Popen([„ssh pi@10.42.0.100 ‚gpio mode 7 out; gpio write 7 0“'], shell=True)

    def on_switch_activated(self,switch,gparam):
        if self.switch.get_active():
            subprocess.Popen([„ssh pi@10.42.0.100 ‚gpio mode 7 out; gpio write 7 1“'], shell=True)
            print(„laser  on“)
        else:
            subprocess.Popen([„ssh pi@10.42.0.100 ‚gpio mode 7 out; gpio write 7 0“'], shell=True)
            print(„laser off“)
```

```
    def on_start_scann_clicked(self, button):

        self.row1.set_sensitive(False)
        self.row2.set_sensitive(False)
        self.row3.set_sensitive(False)
        self.row4.set_sensitive(False)
        self.row5.set_sensitive(False)
        self.combo.set_sensitive(False)
        self.button.remove(self.buttonLabel)
        self.button.set_size_request(50,50)
        self.button.add(self.hbox6)
        self.button.show_all()
        self.button.set_sensitive(False)
        thread = WorkerThread(self.reset_window)
        thread.start()

    def reset_window(self):
        self.dialog2.destroy()
        self.fileName = „/home/david/Dropbox/diplom/PiScan/"+str(entry_text)+'/GrayCombined.png'
        self.folderName = „/home/david/Dropbox/diplom/PiScan/"+str(entry_text)
        subprocess.call([,mkdir', ,'+str(self.folderName)+'/Tile'])
        self.imageBox.remove(self.image)
        pixbuf = GdkPixbuf.Pixbuf.new_from_file_at_size(self.fileName, 600, 600)
        self.mainImageBig = cv2.imread(self.fileName, -1)
        self.mainImage = cv2.resize(self.mainImageBig, (1200,1200))
        self.mainImageZscale = cv2.resize(self.mainImageBig, (1200,1200))
        subprocess.call([,cp', „/home/david/Dropbox/diplom/PiControl/Effects/Tile/kachel.blend', ,'+str(self.
folderName)+'/Tile/kachel.blend'])
        self.baseRows, self.baseCols = self.mainImage.shape
        self.image = Gtk.Image.new_from_pixbuf(pixbuf)
        self.imageBox.add(self.image)
        self.imageBox.show_all()
        self.combineImages()
#       now = datetime.datetime.now()
#       timeString = now.strftime(„%Y-%m-%d_%H:%M")
#       self.entry.set_text(timeString)
#       self.enter_callback(self.entry)
#       self.combo.set_sensitive(True)
#       self.changed_cb(1)
#       self.row1.set_sensitive(True)
#       self.button.remove(self.hbox6)
#       self.button.set_size_request(50,50)
#       self.button.add(self.buttonLabel)
#       self.button.show_all()
#       self.button.set_sensitive(True)


    def changed_cb(self, combo):
        index = self.combo.get_active()
        if index == 0:
            self.row2.set_sensitive(False)
            self.row3.set_sensitive(False)
            self.row4.set_sensitive(False)
            self.row5.set_sensitive(False)
            self.row2.hide()
            self.row3.hide()
            self.row4.hide()
            self.row5.hide()
            self.dialog2.resize(500,200)
        if index == 1:
            self.row2.set_sensitive(True)
            self.row3.set_sensitive(True)
            self.row4.set_sensitive(True)
            self.row5.set_sensitive(True)
            self.row2.show()
            self.row3.show()
            self.row4.show()
            self.row5.show()
    def changed_cb1(self, combo1):
        global index1
        index1 = self.combo1.get_active()
#       print(index1)
    def changed_cb2(self, combo2):
        global index2
        index2 = self.combo2.get_active()
#       print(index2)
    def changed_cb3(self, combo3):
        global index3
        index3 = self.combo3.get_active()
#       print(index3)
    def changed_cb4(self, combo4):
        global index4
        index4 = self.combo4.get_active()
#       print(index4)
    def enter_callback(self, entry):
        global entry_text
        entry_text = self.entry.get_text()
#       print(entry_text)
    def startup_info(self, widget):
        dialog = Gtk.MessageDialog(self, 0, Gtk.MessageType.INFO, Gtk.ButtonsType.OK, „Defaut Start
Procedure for David's PiScan")

        dialog.format_secondary_text(„1.\tTurn on Wi-Fi Hotspot\t\t\t\t\t\t\t \n2.\tConnect BZT Ethernet-
Cable \n3.\tSet Wired Network to BZT \n\n4.\tStart AXIS \n5.\tHome all axes \n6.\tSet X axis touch off 692
\n\tSet Y axis touch off 1500 \n\tSet Z axis touch off 28 \n\n7.\tOpen Terminal and Start Emcrsh")
        dialog.set_size_request(300,300)
        dialog.run()


        dialog.destroy()

win = listboxWindow()
win.connect(„delete-event", Gtk.main_quit)
win.show_all()
Gtk.main()
```

AUSRICHTEN DER EINZELNEN SCAN-BAHNEN
geschrieben in Python

```
#!/usr/bin/env python
# camera video alignement
import time, sys, math, getpass, telnetlib, subprocess, datetime, glob, shutil, thread,
multiprocessing
import cv2
import numpy as np
start = time.time()

fileName, resX, resY, resZ, offsetX, numberOfPasses, timeString, scanAreaX, scanAreaY
= sys.argv
resX = int(resX)
resY = int(resY)
resZ = int(resZ)
offsetX = int(offsetX)
numberOfPasses = int(numberOfPasses)
scanAreaX = int(scanAreaX)
scanAreaY = int(scanAreaY)
#resX = 400
#resY = 2          #resY in frames per mm
#resZ = 400
#offsetX = 100
numberOfCams = 2
#numberOfPasses = 7

resXfloat = resX
resYfloat = resY
resZfloat = resZ
offsetXfloat = offsetX
numberOfPassesfloat = numberOfPasses


if __name__ == ‚__main__‘:

    for c0 in range(0, (numberOfCams)):
        ## Align individual stripes
        #load base image
        baseImg = cv2.imread(‚/home/david/Dropbox/diplom/PiScan/‘+timeString+‘/
Gray/10‘+str(c0)+‘Gray00.png‘,-1)
        baseRows,baseCols = baseImg.shape
        baseImgBig = np.zeros(((baseRows+20), baseCols), np.uint16)
        baseImgBig[10:(baseRows+10), 0:baseCols ] = baseImg
        baseImg = baseImgBig
        # create empty array
        d = np.empty(41)
        # align one strip at a time
        for s0 in range(1,(numberOfPasses)):
            # compare overlaping part of strips
            baseImgCropped = baseImg [40:(baseRows-20), ((resX-
5)+((s0-1)*int(offsetX*resX/200.0))-int(offsetX*resX/400.0)):((resX+5)+((s0-
1)*int(offsetX*resX/200.0))-int(offsetX*resX/400.0))]
            addImg = cv2.imread(‚/home/david/Dropbox/diplom/PiScan/‘+timeString+‘/
Gray/10‘+str(c0)+‘Gray0‘+str(s0)+‘.png‘,-1)
            for d0 in range (0, 41):
                # move selection one pixel per iteration
                addImgCropped = addImg [(10+d0):(baseRows-(50-d0)), ((resX-
5)+((s0-1)*int(offsetX*resX/200.0))-int(offsetX*resX/400.0)):((resX+5)+((s0-
1)*int(offsetX*resX/200.0))-int(offsetX*resX/400.0))]
                # calculate value for how different the two strips are
                dist = np.linalg.norm(baseImgCropped-addImgCropped)
                d[d0] = dist
            # find index of array entry with lowest value
            shiftValue = np.argmin(d)
            rows,cols = addImg.shape
            addImgBig = np.zeros(((rows+41), cols), np.uint16)
            addImgBig[20:(rows+20), 0:cols ] = addImg
            #shift image
            addImg = addImgBig[(shiftValue):(rows+shiftValue), 0:cols]*1
            # combine stripe by highest Value
            rows,cols = addImg.shape
            baseImg[10:(int((scanAreaY*resY*1.0)+20)+10),0:cols] =
np.fmax(baseImg[10:(int((scanAreaY*resY*1.0)+20)+10),0:cols], addImg)
#           for r0 in range(0,rows):
#               for t0 in range(0,cols):
#                   val1 = baseImg[10+r0,t0]
#                   val0 = addImg[r0,t0]
#                   if val1 > val0:
#                       baseImg[(r0+10),t0] = val1
#                   else:
#                       baseImg[(r0+10),t0] = val0

        baseRows,baseCols = baseImg.shape
        baseImg = baseImg[baseRows-int(scanAreaY*resY*1.0):baseRows,
int(resX/2.0):int((scanAreaX*resX/200.0)+int(resX/2.0))]
        cv2.imwrite(‚/home/david/Dropbox/diplom/
PiScan/‘+timeString+‘/10‘+str(c0)+‘Gray.png‘, baseImg)


        gray1 = cv2.imread(‚/home/david/Dropbox/diplom/PiScan/‘+timeString+‘/101Gray.
png‘,-1)
        gray0 = cv2.imread(‚/home/david/Dropbox/diplom/PiScan/‘+timeString+‘/100Gray.
png‘,-1)
        rows,cols = gray0.shape
        g1Cropped = gray1 [40:(int(scanAreaY*resY*1.0)-40), :]
        for d0 in range (0, 41):
            # move selection one pixel per iteration
            g0Cropped = gray0 [(10+d0):(int(scanAreaY*resY*1.0)-(70-d0)), :]
            # calculate value for how different the two strips are
            dist = np.linalg.norm(g1Cropped-g0Cropped)
            d[d0] = dist
        # find index of array entry with lowest value
        shiftValue = np.argmin(d)
        g0Big = np.zeros((int((scanAreaY*resY*1.0)+61), cols), np.uint16)
        g0Big[30:int((scanAreaY*resY*1.0)+30), 0:cols ] = gray0
        #shift image
        gray0 = g0Big[(shiftValue):(int(scanAreaY*resY*1.0)+shiftValue), 0:cols]*1
        # combine stripe by highest Value
#       gray1 = np.fmax(gray1, gray0)


        grayCombined = np.fmax(gray1, gray0)
#       grayRows, grayCols = gray1.shape
#       for r0 in range(0,grayRows):
#           for t0 in range(0,grayCols):
#               val1 = gray1[r0,t0]
#               val0 = gray0[r0,t0]
#               if val1 > val0:
#                   grayCombined[r0,t0] = val1
#               else:
#                   grayCombined[r0,t0] = val0
        cv2.imwrite(‚/home/david/Dropbox/diplom/PiScan/‘+timeString+‘/GrayCombined.
png‘, grayCombined)
        curr = time.time()
        delta = curr-start
                                                                                print(delta)
```

AUSWERTUNG DER KAMERABILDER
geschrieben in Python

```
#!/usr/bin/env python
# camera video processing

import time, sys, math, getpass, telnetlib, subprocess, datetime, glob, shutil, thread,
multiprocessing
import cv2
import numpy as np
start = time.time()

fileName, resX, resY, resZ, offsetX, numberOfPasses, timeString, scanAreaX, scanAreaY,
c0, s0 = sys.argv
resX = int(resX)
resY = int(resY)
resZ = int(resZ)
offsetX = int(offsetX)
numberOfPasses = int(numberOfPasses)
scanAreaX = int(scanAreaX)
scanAreaY = int(scanAreaY)
c0 = int(c0)
s0 = int(s0)
#resX = 400
#resY = 2          #resY in frames per mm
#resZ = 400
#offsetX = 100
numberOfCams = 2
#numberOfPasses = 7

resXfloat = resX
resYfloat = resY
resZfloat = resZ
offsetXfloat = offsetX
numberOfPassesfloat = numberOfPasses


## split video into individual images
if __name__ == '__main__':

        print „Process: Camera: „,c0,“ Video: „,s0
        subprocess.call([,mkdir', ,/home/david/Dropbox/diplom/
PiScan/'+timeString+'/Gray'])
    ## process frames and generate displacementmap
    # create matirx for grayscale image
        dimensionY = int((scanAreaY*resY*1.0)+20)
        imageSizeX = ((float(resXfloat)*float(numberOfPassesfloat))-((float(resXfloat)/20
0)*float(offsetXfloat)*(float(numberOfPassesfloat)-1)))
        imgGray = np.zeros((dimensionY, int(imageSizeX)), np.uint16)
        # points for transformation matrix
        if c0 == 0:
#           pts1 = np.float32([[13,649],[1283,633],[320,398],[972,388]]) #scaner 1
          pts1 = np.float32([[80,608],[1206,625],[306,68],[1003,74]]) #scaner 2
        elif c0 == 1:
#           pts1 = np.float32([[53,237],[1244,231],[287,846],[1009,840]]) #scaner 1
          pts1 = np.float32([[1182,564],[66,578],[969,21],[267,31]]) #scaner 2


        pts2 = np.float32([[0,0],[(resX),0],[0,(resZ)],[(resX),(resZ)]])

    # create transformation matrix
        M = cv2.getPerspectiveTransform(pts1,pts2)
        cap = cv2.VideoCapture(,/home/david/Dropbox/diplom/
PiScan/'+timeString+'/10'+str(c0)+'vid0'+str(s0)+'.h264')
        f0=1
        StripList = []
        for num in range(0,dimensionY):
          ret, frame = cap.read()
          if(ret):
            f0 = f0+1
            img = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
            rows,cols = img.shape

            # apply perspective transformation
            img = cv2.warpPerspective(img,M,(1292,972))

            # crop y1:y2, x1:x2
            img = img[0:(resZ), 0:(resZ)]
            img = cv2.GaussianBlur(img,(5,5),0)
            imgMax = np.argmax(img, axis=0)
            imgMaxVal = np.amax(img, axis=0)
            imgMax[imgMaxVal < 10] = resZ
            imgMax = np.rint((65535.0/ resZ)*(resZ-imgMax))
            StripList.append(imgMax)
          else:
            cap.release()
            break
        cap.release()
        Strip = np.array(StripList)
        Strip = np.uint16(Strip)
        Strip = np.flipud(Strip)
        Strip = np.fliplr(Strip)
        del StripList
        imgGray[0:dimensionY,
(int(s0*offsetX*(resX/200.0))):(int(s0*offsetX*(resX/200.0)+resX)) ] = Strip
        cv2.imwrite(,/home/david/Dropbox/diplom/PiScan/'+timeString+'/Gray/10'+s
tr(c0)+'Gray0'+str(s0)+'.png', imgGray)
```

MEIN DANK GILT...

… Univ. Prof. Arch. Dipl. Ing. Christian Kern für die Betreuung von und das Interesse an meiner Diplomarbeit

… meiner Familie für anhaltende Unterstützung und endlose Geduld insbesondere …
- … Karl Schwärzler, dem ich mitunter ein Grundverständnis für Maschinenbau, Elektrotechnik und Informatik verdanke
- … Kathy Schwärzler, der ich nicht nur Kreativität und Englischkenntnisse verdanke
- … Leonie Schwärzler für seelischen und grammatikalischen Beistand und intensives Testen meines Prozesses
- … Anton Schwärzler für jederzeitige universelle Einsatzbereitschaft
- … Gerhard Schwärzler für die Begeisterung an meinem Prozess und seine Fähigkeiten im Bereich Networking

… meinen Freunden, Bekannten und Kollegen …
- … Josef Schwendinger für die Begleitung im Studium, Unterstützung beim Layout und als Versuchskaninchen
- … Johannes Lerbscher für seine kompetente Beratung in Sachen Zement
- … Michael Frühstück, Lukas Vögel und Max Weh als weitere Versuchskaninchen
- … Thomas Amann für das Beisteuern der Raspberry Pi‘s
- … Iris, Martina, Katharina, Stephi, Clarissa, David, Elena und allen anderen für das jahrelange Mitfiebern und Helfen

… dem Institut für Kunst und dreidimensionales Gestalten …
- … der gesamten Belegschaft der Modellbauwerkstatt: insbesondere Walter Fritz, Kornelia Fischer und Ronald Buchinger
- … den Fotografen Christian Chladek und Augustin Fischer
- … Florian Rist für das Verstehen meiner Scherze und der Finessen meines Prozesses
- … Raimund Krenmüller als Leidensgenosse und für inspirierende Gespräche

… den Künstlern Gottfried Bechtold, Edgar Leissing, Alexander Viscio und Alexandra Wacker für ihre Experimentierfreudigkeit