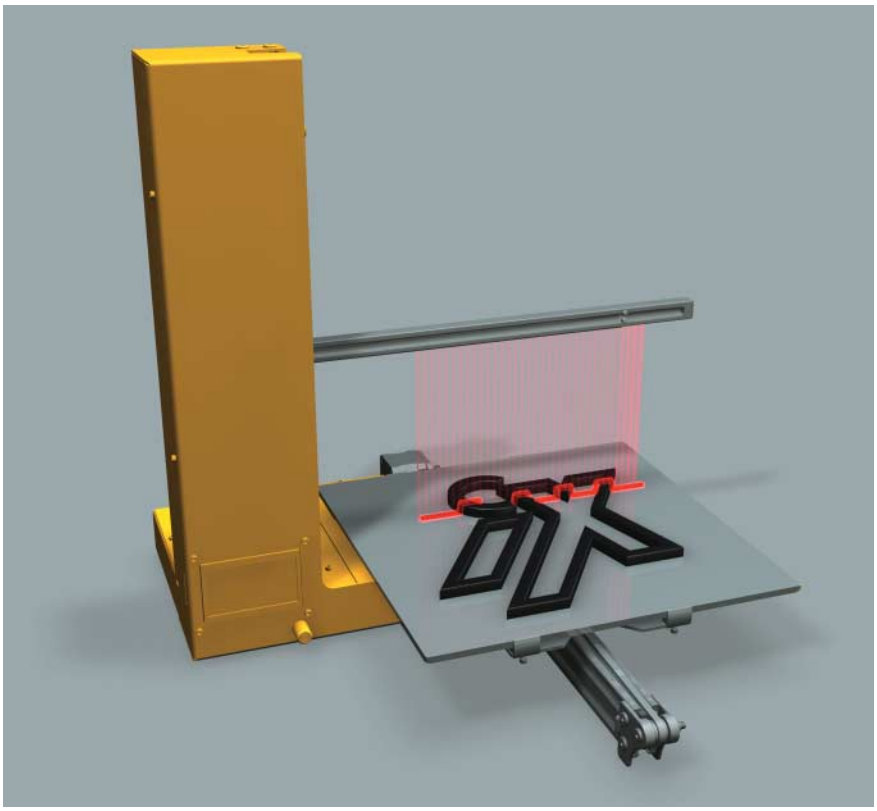


3D-Druck-Tutorial, Teil 1: 3D-Scan mit Open-Source-Software

Abgetastet

Klaus Knopper

Der Weg vom realen Objekt zum 3D-Modell und einer 3D-gedruckten Kopie kann aufwendig und teuer sein. Freie Hard- und Softwarekomponenten können den kompletten Workflow abdecken.

Nicht nur auf engagierte Hobbyisten üben 3D-Drucker eine große Faszination aus. Für den Erfinder und Unternehmer ist 3D-Prototyping eine willkommene Option, Proofs of Concept und Vorführgeräte in Eigenregie oder für Partner herzustellen und weiterzuentwickeln, lange bevor die Serienreife des Produkts erreicht ist.

Allerdings dürfte das Bauen von Gegenständen mit dem 3D-Drucker in den wenigsten Fällen schneller vonstattengehen als mit traditionellem Werkzeug und Holz oder Metall – gerade komplexe Formen erfordern mitunter eine Druckzeit von mehreren Tagen und bergen das Risiko, im letzten Moment zu scheitern. Aber immerhin muss der Konstrukteur in der Phase des eigentlichen Druckens nicht mehr selbst aktiv werden.

Tutorialinhalt

Teil 1:	**3D-Scan und Umwandlung in Volumenmodelle**
Teil 2:	3D-Konstruktion versus „Zeichnen“
Teil 3:	3D-Druck mit OctoPrint

Dieses dreiteilige Tutorial beleuchtet die Arbeitsschritte der 3D-Modellierung mithilfe für viele Betriebssysteme frei verfügbarer Open-Source-Software und gibt eine Einführung in die Komponenten des in Abbildung 1 gezeigten Workflows.

Vom realen Objekt zum Modell – 3D-Scan

Auf das kreative eigene Erstellen von 3D-Modellen wird der zweite Teil dieses Tutorials ausführlich eingehen. Zunächst soll das Erfassen und Scannen realer Gegenstände sowie das Umwandeln in Volumenmodelle im Vordergrund stehen.

Fotos, selbst die bekannten Photosphere-Varianten mit „3D-View“, sind grundsätzlich zweidimensional (x-y-Koordinaten), das heißt, die Tiefeninformation (z) fehlt, was bei einer einzigen Kamera beziehungsweise geringem Abstand zwischen zwei Kameras durchaus verständlich ist. Allenfalls lässt sich die Tiefeninformation noch bei Graustufenbildern aus der Helligkeit jedes Pixels extrahieren, was zur Oberflächenbildung für sogenannte Lithophane (auf die der Konstruktions-Teil noch eingehen wird) dienen kann.

Als Basis für ein echtes 3D-Volumenmodell benötigt man, als Ergebnis eines 3D-Scans, zunächst eine Punktwolke. Das sind x-y-z-Koordinaten, aus denen sich mittels einiger mathematischer Operationen die Außenfläche eines geschlossenen Körpers berechnen lässt.

Erste Experimente: MakerScanner, Ciclop & Horus

Mit einem Materialeinsatz von unter 50 Euro ist der MakerScanner vermutlich das preisgünstigste und lehrreichste (nebenbei eines der ersten) DIY-Scanner-Projekt, das dabei hilft, das Verfahren zu verstehen – gleichzeitig aber auch das unkomfortabelste. Das System generiert mit einer USB-Webcam und einem Linearlaser durch manuelles Verändern der Oberflächenausleuchtung eine Punktwolke. Eine professionelle Variante beschreibt der Kasten „Zum Vergleich: David Scanner“.

Zumindest einen Vorteil gegenüber leichter automatisierbaren Verfahren hat der MakerScanner: Die Objekte können auch mal etwas größer sein. Baut man sie weiter entfernt von der Kamera auf, könnte sich – in Abhängigkeit vom Aufnahmewinkel und der Reichweite des Li-

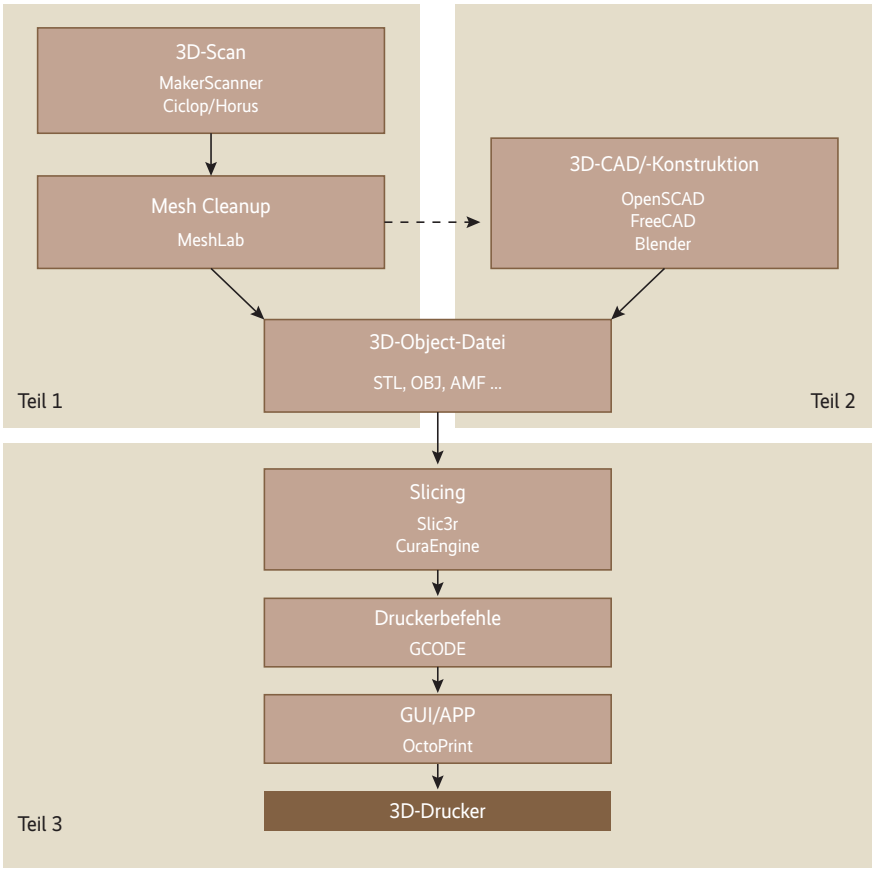
nearlasers – bei möglichst dunkler Umgebung sogar eine Person im Profil scannen lassen. Da MakerScanner aber nur von einer Seite aufnimmt, ist der Aufwand für die Nachbearbeitung – mit dem Entfernen von Pixeln aus der Umgebung oder gar dem Zusammensetzen mehrerer Scans – ziemlich hoch. Der Abschnitt „Nachbearbeiten der Punktwolke mit MeshLab“ geht später genauer auf die Nachbearbeitung ein.

Um die Oberfläche eines Objektes in regelmäßigen Abständen zu scannen, kommt bei kleineren Objekten gerne ein Drehteller zum Einsatz – per Software gesteuert und mit dem Auslöser zur Bildaufnahme gekoppelt. Dies erleichtert das beim MakerScanner etwas wackelige manuelle Abtasten und erlaubt auch einen Rundum-Scan. Auch bei dieser Methode enthält die Punktwolke Löcher. Die können später beim Erzeugen geschlossener Volumina Probleme bereiten, da die auf dem Drehteller aufliegende Unterseite der Objekte natürlich nicht erfasst wird und folglich dort Punkte fehlen.

Ebenfalls ein Projekt zum Selbstbau, hier sogar mit vielen 3D-druckbaren Teilen, ist der Ciclop, ein Open-Source-Projekt von BQ. Das Gerät bietet neben der Webcam zwei Linearlaser, die das Scannen beschleunigen beziehungsweise die Oberflächenabdeckung vervollständigen sollen. Diverse Anbieter haben die Hardware beinahe zum Selbstkostenpreis inklusive der gedruckten Teile, Motor und Arduino-Board für rund 100 Euro im Portfolio.

Leider stellt aber die komplementäre Software Horus, obwohl in portablem Python geschrieben und grundsätzlich auf Linux, macOS oder Windows lauffähig, inzwischen eine Herausforderung dar, sowohl bei Selbstkompilierern als auch für die Installation als Paket. Man muss hierfür entweder eine laut Hersteller zu Optimierungszwecken gepatchte Version der Grafikbibliothek OpenCV installieren oder den Quellcode entsprechend patchen. Andernfalls lässt sich die Belichtungszeit nicht verändern. Die OpenCV-Spezialversion wiederum harmoniert nicht mit den meisten modernen Linux-Installationen, und das Projekt wurde lange nicht aktualisiert.

Eine Anpassung auf aktuelle Linux-Versionen ist also etwas mühsam, als Belohnung dafür winkt immerhin eine nette grafische Oberfläche mit Preview und integrierten Optimierungsfunktionen wie dem Festlegen des verwendeten Scanbereichs (siehe Abbildung 3). Das kann die Nachbearbeitungszeit etwas verkürzen. Das erste Kalibrieren des Systems ist noch ein richtiges Abenteuer, aber einmal mit den richtigen Werten und der passenden Beleuchtung eingerichtet, kann es Gegenstände bis zu 18 cm hoch und breit leicht halbautomatisch einscannen und als Punktwolken speichern.



Jeder Teil dieses Tutorials deckt einen spezifischen Bereich des typischen 3D-Workflows ab (Abb. 1).

Mehr Automatisierung: Freihandfotos und COLMAP

In den diversen App-Stores für Android oder iOS finden sich etliche Apps, die auf wunderbare Weise Freihand-3D-Scans versprechen – zumindest für Personen mit viel Geduld, einer ruhigen Hand und ausreichend Speicher für mindestens 20 Rundumfotos, für die man mitunter eine Leiter braucht. Die Idee ist natürlich gut: Warum eine Spezialhardware zum Scannen anschaffen, wenn Software per Stitching aus vielen Einzelbildern eine 3D-Szene generieren kann? Man kennt das ja vom Cardboard als Photosphere – auch wenn dies keine echten 3D-Bilder sind, sondern nur aus einer

iX-TRACT

- Per 3D-Scan lassen sich von realen Gegenständen in mehreren Schritten bearbeitbare 3D-Modelle erstellen.
- Zwar erfordert das Nachbearbeiten der Punktwolken oft mehr Aufwand als das eigentliche Erfassen, es ist aber zum Entfernen von Scanfehlern und Artefakten unumgänglich.
- MeshLab ist – im Gegensatz zum eher künstlerisch orientierten Blender – ein technisch-mathematisches Werkzeug zum Bearbeiten von 3D-Punktwolken und -Modellen.
- Ein mit den richtigen Tools erstelltes „wasserdichtes“ 3D-Volumenmodell kann direkt als 3D-Druckvorlage dienen.

Zum Vergleich: David Scanner

Nicht unerwähnt bleiben sollen hier proprietäre Pakete für den 3D-Scan. Mit Preisen ab rund 2500 Euro für das aktuelle Modell SLS-3 bietet der Structured Light Scanner der von HP 2016 übernommenen David Vision Systems eine gute Variante. Die Geräte liefern mit einer Kombination aus einem per Mini-Beamer erzeugten Muster und einer hochauflösenden Kamera sowie spezieller Software sehr präzise Scans.

Nach eingehender Kalibrierung (ähnlich wie beim Ciclop) und Ausmessung des Hintergrunds verhindert die Software über eingebaute Filter Artefakte schon recht gut automatisch. Per Software zusammengesetzte „Freihand"-Scans ohne Drehteller sind ebenfalls möglich, ähnlich wie bei der im Artikel gezeigten COLMAP-Variante. Allerdings bietet die Firma die zugehörigen Programme ausschließlich für Windows an.

bestimmten Position einen räumlichen Eindruck der Umgebung vermitteln.

Ein 3D-Scan per Bildfolge funktioniert quasi umgekehrt: Statt eines stationären Beobachters mit einer Szene als 360°-Aufnahme nimmt man hier ein unbewegtes Objekt aus vielen Winkeln auf – bekannt als „Structure from Motion" (SfM) beziehungsweise Fotogrammetrie.

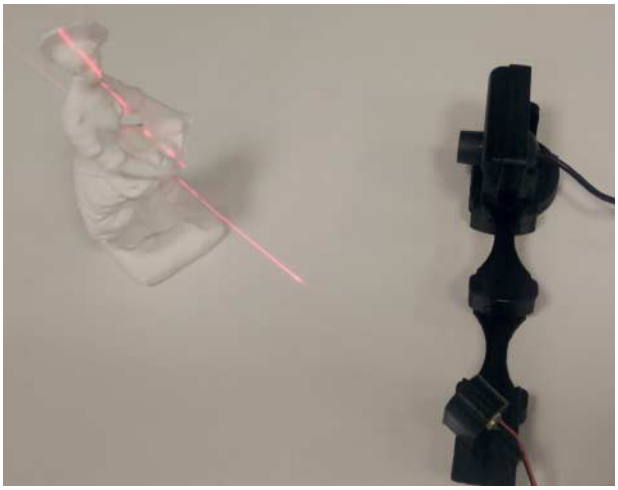
Eine relativ neue SfM- und Multi-View-Stereo-Software (MVS) ist das unter BSD-Lizenz stehende COLMAP. Debian-Nutzer können es per

```
sudo apt install -t unstable colmap
```

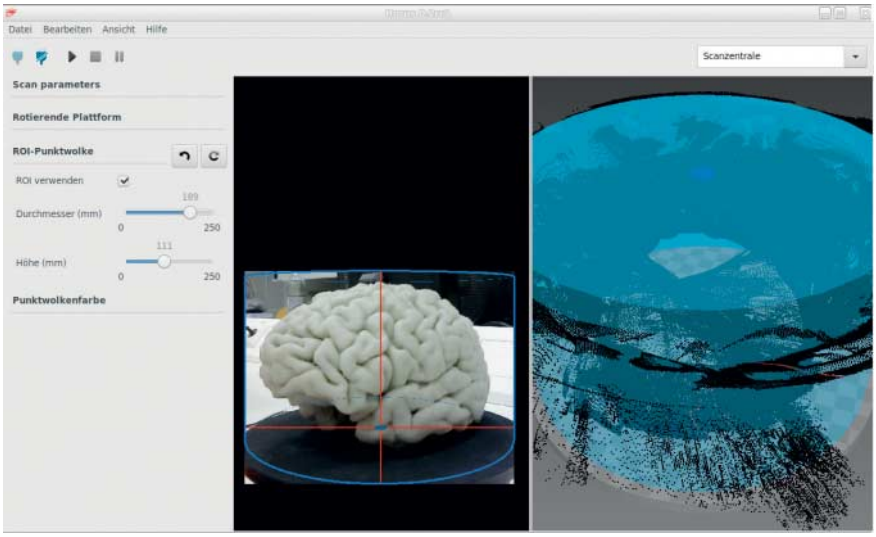
aus dem unstable-Tree installieren. Eigentlich benötigt das Programm einen Grafikkartentreiber mit OpenGL-3.3-Unterstützung und die MESA-Bibliotheken – nicht nur, um mit schnellem GPU-Rendering arbeiten zu können, sondern um überhaupt zu starten. Setzt man zwei Umgebungsvariablen, funktioniert es aber auch mit Software-Rendering:

```
MESA_GL_VERSION_OVERRIDE=3.3 LIBGL_ALWAYS_
                                SOFTWARE=1 colmap gui
```

Wobei man in dieser Variante bei „Automatic Reconstruct" das Feld „GPU" tunlichst deselektiert und den gpu_index auf einen Wert größer 0 setzen sollte (siehe Abbildung 4). Das dürfte vermutlich ein Fehler der frühen Version sein. Das Rekonstruieren lässt sich dank umfangreicher Kommandozeilenoptionen auch skripten.



Zwar ist ein MakerScanner sehr preisgünstig, aber er liefert auch nur eine magere Qualität (Abb. 2).



Hat man die Mühen der Kalibrierung gemeistert, bietet Horus eine nette grafische Oberfläche mit Preview und integrierten Optimierungsfunktionen (Abb. 3).

Nachbearbeiten der Punktwolke mit MeshLab

Ein beliebtes Programm zum Verbessern und Reparieren von 3D-Daten aller Art ist MeshLab. Es bringt zwar eine ausführliche Dokumentation der Filter und Optionen mit, bleibt aber mathematisch anspruchsvoll und verlangt vom Anwender fortgeschrittene Kenntnisse. Leider haben die Entwickler wenige Plausibilitäts-Checks eingebaut, sodass das Programm eingelesene oder erzeugte Daten vor der Weiterverarbeitung nur sehr lax prüft. Darüber hinaus verabschiedet sich MeshLab bei Speichermangel oder auch schon „schlechten" Daten gern mal ohne Speicherung der Zwischenergebnisse.

Aus diesem Grund und da MeshLab in der aktuellen Version kein Rückgängigmachen kennt, sollte man sicherheitshalber nach jedem Schritt speichern. Ähnlich wie GIMP unterscheidet das Programm dabei zwischen den als „Layer" erzeugten Ergebnissen und dem Gesamtprojekt. Im Regelfall genügt es, das jeweils erfolgreiche Zwischenergebnis zu speichern und die eventuell noch vorhandenen weiteren Layer zu löschen, sofern man sie nicht in einem späteren Schritt benötigt. Unten folgt ein komplexes Beispiel, mit dem sich materialsparende und künstlerisch wertvoll wirkende Voronoi-Muster auf Volumenoberflächen erzeugen lassen.

Zunächst soll aber aus der via 3D-Scan generierten Punktwolke ein druckbares Modell im STL-Format entstehen. Dazu lädt man die gescannte Punktwolke im Standard-Polygon-Format (PLY) *Blumenmaedchen.ply* in MeshLab. Anschließend markiert man im Edit-Werkzeug per „Select Vertices" oder „Select

Faces in a rectangular region“ überflüssige Punkte und entfernt sie (siehe Abbildung 5).

Im nächsten Schritt normalisiert der Aufruf „Filters/Normals, Curvatures and Orientation/Compute normals for point sets“ alle verbleibenden Punkte. Das Programm legt dabei einen Vektor von innen nach außen durch alle Punkte, wodurch sich die Flächen bei der Berechnung des Volumenkörpers später einheitlich und senkrecht zur Normalen ausrichten lassen. Einige Filter funktionieren nicht oder produzieren Löcher, wenn die Punkte „falsch herum“ orientiert sind.

Unter „Reconstruction and Remeshing“ den „Screened Poisson Filter“ auswählen. Je höher man die „Reconstruction Depth“ setzt, desto detailreicher gerät das Ergebnis. Leider werden auch eventuell anfangs übersehene Scanfehler mitgenommen. Ab einer Tiefe von 12 kann der Speicherverbrauch bei der Berechnung exorbitant steigen. Der Filter erzeugt ein Modell, indem er alle Punkte durch Flächen mit der Orientierung der zuvor korrigierten Normalen verbindet. Das bedeutet zwar nicht automatisch, dass das Modell wasserdicht ist, aber zumindest ist es größtenteils geschlossen, bis auf gegebenenfalls nicht scanbare Bereiche, die als „Löcher“ verbleiben. Diese lassen sich in einem zusätzlichen Schritt manuell durch Einfügen weiterer Punkte „kitten“.

Letzter Feinschliff vor dem Drucken

Mit diesem Filter schließt man gelegentlich größere „leere“ Bereiche, da in diesen die Normalen nach außen zeigen, mit einer Art „Ballon“, was merkwürdig aussieht (siehe Abbildung 6). Man kann dem abhelfen durch das manuelle Einfügen zusätzlicher Punkte und Flächen in den „offenen“ Bereichen **vor** Ausführen des Filters. Alternativ kann man die Ballons später entfernen und beispielsweise durch einen Sockel, Zylinder oder Würfel als Standfläche ergänzen. Nicht mehr benötigte Layer sollte man anschließend löschen – und das Speichern nach jedem Schritt nicht vergessen.

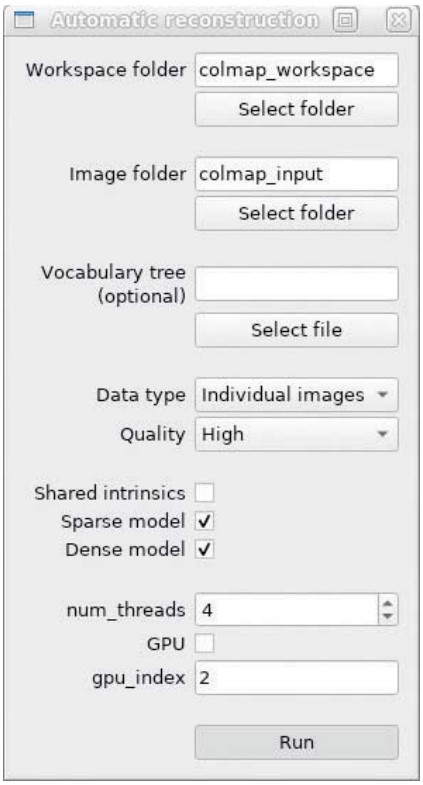
Eventuell entstandene Artefakte, etwa die eben genannten Ballons, lassen sich ähnlich wie die überflüssigen Bildpunkte oben mit dem Selektionstool entfernen, allerdings bekommt das Modell dabei wieder Löcher. Wenn die falsch verbundenen Punkte sehr weit auseinanderliegen, kann der Auswahlfilter „Select by length“ als Alternative dienen. Optional kann der „Laplacian Smooth“-Filter das Modell erneut glätten.

Zur weiteren Bearbeitung oder für die Druckvorstufe sichert man das Ergebnis üblicherweise als STL-Datei. Da das STL-Format weder Farb- noch Oberflächenstrukturinformationen berücksichtigt, empfiehlt es sich, gegebenenfalls zusätzlich eine Kopie im **.ply*-Format aufzubewahren.

Voronoi-Muster auf Oberflächen erzeugen

Nicht nur materialsparend, sondern auch beeindruckend filigran können mit Voronoi-Mustern modifizierte Flächen sein. Hierbei wird eine Art unregelmäßiges Wurzelmuster erzeugt, das das Modell wie gewachsen aussehen lässt. Weniger für solide Bauteilkonstruktion, aber als ansehnliche Designvariante interessant, sind Voronoi-Objekte auch auf Plattformen wie Thingiverse sehr beliebt.

Als Beispiel soll hier eine in z-Richtung skalierte Kugel dienen, die als Ei ein

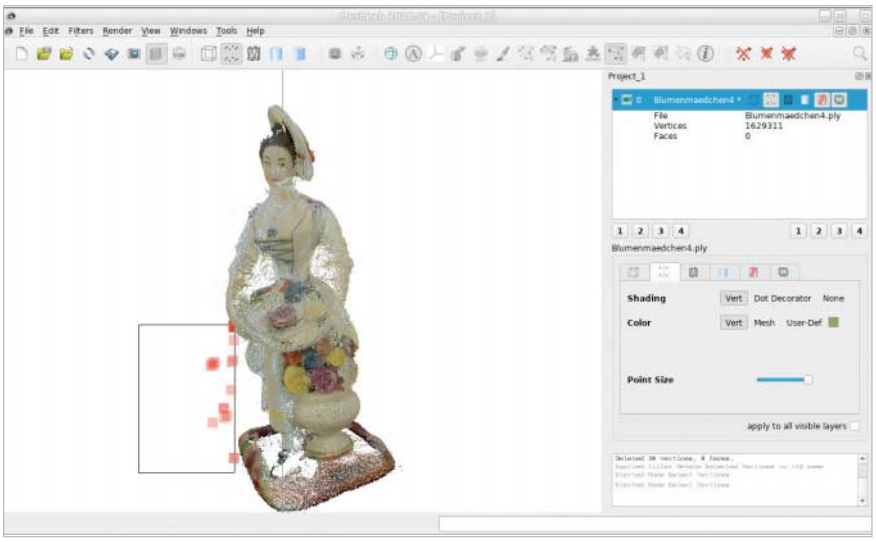


Beim Einsatz von Software-Rendering gilt es, einige Besonderheiten zu beachten (Abb. 4).

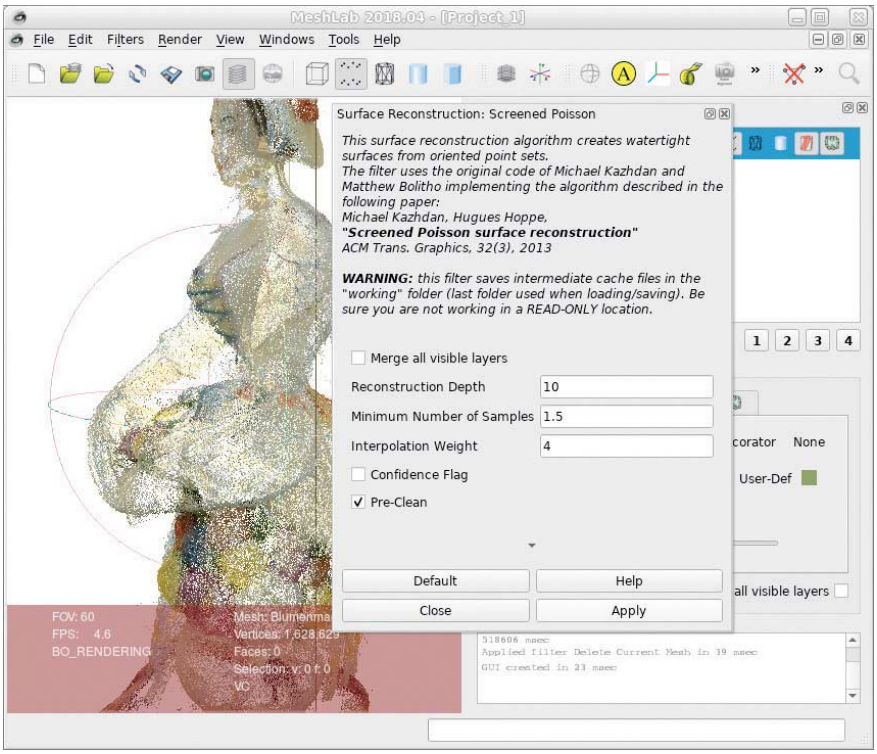
Tipps für den erfolgreichen 3D-Scan

Beim Aufnehmen einer Punktwolke ist es besonders wichtig, dass die gescannten Punkte tatsächlich auf der Oberfläche des Objektes liegen und nicht durch Reflexionseffekte oder die Umgebung zu viele „Artefakte“ entstehen. Glänzende oder komplett schwarze Oberflächen sollte man vermeiden. Sofern keine lebenden Objekte eingescannt werden, hat sich Kreidestaub, auf das Objekt gesprüht oder mit einem Pinsel aufgetragen, als Verbesserer für Oberflächen bewährt. Auch Malerkrepp wird oft empfohlen, wenn es sich um glatte, reflektierende Oberflächen handelt.

Für eine reflexionsarme Umgebung bietet sich hinter dem Scanbereich großflächig angebrachtes schwarzes Kartonpapier an. Und um Probleme mit Schattenwurf auszuschließen, sollte man den Raum mit einem stark streuenden Licht ausleuchten, etwa mit einer großen, matten LED-Lampe nicht aber mit einer punktförmigen Lichtquelle.



Aus den Scandaten sind zunächst einige überflüssige Bildpunkte zu entfernen (Abb. 5).



Bei der Oberflächenrekonstruktion füllt MeshLab Lücken in der Punktwolke mit ballonartigen Gebilden auf (Abb. 6).

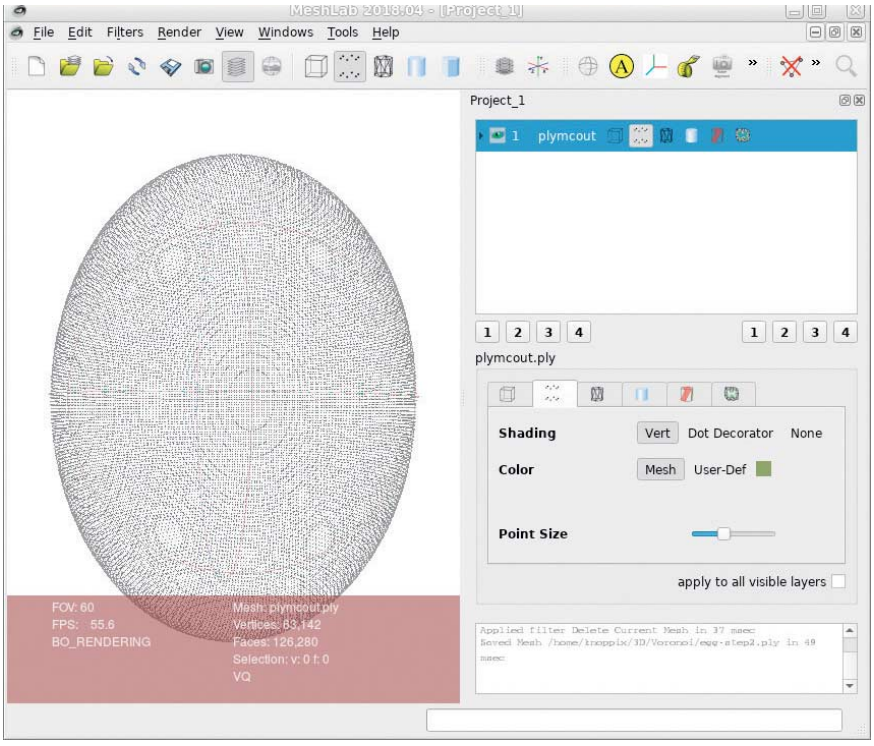
geädertes Muster erhalten soll. Sie lässt sich mit dem im zweiten Teil des Tutorials detaillierter vorgestellten OpenSCAD mit folgendem Einzeiler erstellen:

```
module egg(size=100){ scale([0.75,0.75,1])
sphere(d=size,$fn=100); } egg();
```

Zunächst muss man das Mesh (STL-Datei) importieren. Hier ist es hilfreich, sich neben der „Fill"-Version auch das Linienmuster und die Punktwolke der Flächeneckpunkte anzeigen zu lassen: „Points" und „Wireframe" in der Layer-Ansicht beziehungsweise unter „Render" bei einer älteren MeshLab-Version.

Anschließend setzt man unter „Filters/remeshing, simplification and reconstruction/surface reconstruction VCG" die Voxel-Größe im Feld „perc" auf den Wert 0.5. Mit dem Klick auf Apply erzeugt dieser Filter eine neue, in diesem Fall feiner aufgelöste Punktwolke auf der Oberfläche.

Blendet man jetzt in der Layer-Ansicht mit einem Klick auf das Augensymbol das alte Mesh aus, kann man das Aussehen des neuen kontrollieren. Auch hier ist wieder die Punktwolken- oder die Wireframe-Ansicht hilfreich. Im mit der rechten Maustaste aufgerufenen Kontextmenü entfernt man das ursprünglich geladene Mesh nun per „Delete current mesh" aus der Bearbeitung. Es empfiehlt sich, das in Abbildung 7 gezeigte neue Mesh – wie in MeshLab üblich – als PLY-Datei zu exportieren.



Eine Verfeinerung des Gitters mit zusätzlichen Knoten bewirkt eine glattere Oberfläche des Modells (Abb. 7).

Das Grobmodell weiter verfeinern

Im folgenden Schritt setzt man analog unter „Filters/remeshing, simplification and reconstruction/Subdivision Surfaces: Loop" die Voxel-Größe ebenfalls auf 0.5 (perc). Nach dem Klick auf Apply enthält das Mesh noch einmal deutlich mehr Zwischenpunkte, was sich dem Anwender gegebenenfalls erst beim Hereinzoomen zeigt. Die neuen Punkte sind nicht mehr regelmäßig angeordnet.

Für die nächste Stufe setzt man in „Filters/Sampling/Poission-disk Sampling" „Number of samples" auf (ungefähr) 200 und selektiert die Checkbox „Exact Number of Samples". Sollte das Kästchen nicht existieren, so wie bei alten MeshLab-Versionen, ist der Wert „Explicit Radius/perc" auf 0.5 zu setzen und anschließend im Ausgabefenster zu kontrollieren, ob die Anzahl erzeugter Punkte bei etwa 200 liegt. Diese sollen die „Löcher" im erzeugten Muster darstellen. Falls die Zahl nicht wie gewünscht ist, muss man sich mit einem neuen Versuch – dabei das zuvor erzeugte Mesh wieder löschen – und geändertem perc-Wert an die gewünschte Zahl heranarbeiten. In diesem Schritt sind unbedingt das Original-Mesh **und** das neu erzeugte „Poisson Disk"-Mesh in der Layer-Ansicht zu behalten.

Im Dialog „Filters/Sampling/Voronoi Vertex Coloring" stellt man die Eigenschaften für die zu generierenden Muster ein. Hierzu ist als „To be colored mesh" das erste und als „Vertex Mesh" das im

vorigen Schritt erzeugte Mesh „Poisson Disk“ anzugeben sowie „Enable Backdistance“ auszuwählen. Details zur Bedeutung der einzelnen Optionen liefert die per Button einblendbare Live-Hilfe.

Das generierte Muster sieht man am besten, wenn man anschließend im Layer des ersten Mesh das „Fill“-Rendering aktiviert. Das „Poisson Disk“-Mesh kann nun gelöscht, das verbleibende Mesh sollte wieder zwischengespeichert werden.

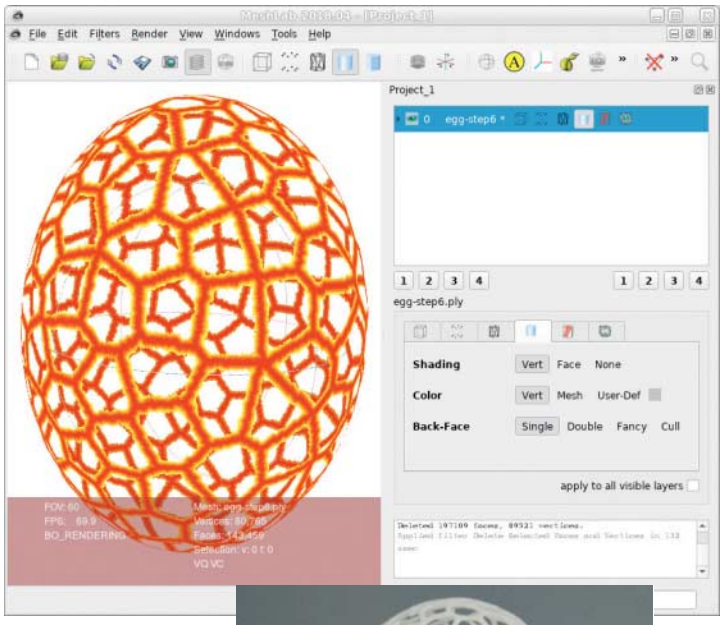
Für das Einstellen der Liniendicke ist in „Filters/Selection/Select faces by vertex quality“ ebenfalls „Enable preview“ einzuschalten. In der Punktwolke ist die Auswahl etwas besser zu sehen als in der „Fill“-Variante. Mit dem Wert 0 für „Min Quality“ lässt sich per „Max Quality“ die Dicke der Linien einstellen. Sobald das Ergebnis gut aussieht, speichert der Klick auf „Apply“ die Änderungen.

Der nächste Schritt übernimmt das Beseitigen der Zwischenräume. Dazu muss man unter „Filters/Selection/Invert Selection“ sowohl die Flächen (Faces) als auch die Knoten (Vertices) invertieren. Anschließend entfernt „Filters/Selection/Delete selected faces and vertices“ die Zwischenräume, und die Linien bleiben stehen (siehe Abbildung 8). Noch sind diese allerdings flach, das heißt, das Objekt wäre nicht druckbar.

Dafür glättet man mit „Filters/Smoothing, fairing and deformation/Laplacian smooth“ mit einem Laplace-Filter zunächst die Linien. Diesen Filter muss man gegebenenfalls bei unveränderten Werten mehrfach anwenden, bis das Ergebnis gefällt.

Nun unter „Remeshing, simplification and reconstruction/Uniform mesh resampling“ die Optionen „Enable multisample“ und „Absolute Distance“ aktivieren, „precision perc“ auf 0.5 (je kleiner dieser Wert, desto detaillierter wird die Oberfläche – und desto länger und speicherintensiver ist das Rendering) sowie „offset perc on“ auf 52 setzen. Da dieser Filter bekanntermaßen sehr „empfindlich“ reagiert, sollte man das Mesh vorher nochmal speichern. Mit dem „Apply“ erhalten die Linien nun endlich auch ein Volumen. Hat das „Offset Mesh“ die gewünschte Form, kann man das ursprüngliche Mesh wieder entfernen.

Anschließend folgt der letzte optische Schliff: „Filters/Smoothing, fairing and deformation/Taubin smooth“ glättet die Strukturen, wobei die Werte Lambda und mu die Dicke der Verbindungen beeinflussen, was sich im Preview kontrollieren lässt. Nach dem Speichern als STL lässt sich die Datei drucken, Abbildung 9 zeigt das lasergedruckte fertige Modell.



Das vormals geschlossene Ei besitzt jetzt eine Voronoi-Struktur als Oberfläche (Abb. 8).

Für ältere MeshLab-Versionen existiert auf Thingiverse eine ausführliche Anleitung. Aus Platzgründen fehlen einige Abbildungen mit Zwischenschritten. Den vollständigen Satz gibt es über den iX-Listingservice (siehe ix.de/ix1902118).

Fazit

Mithilfe eines 3D-Scanners und oft aufwendiger Nachbearbeitung lassen sich reale Objekte in 3D-Modelle überführen und in den gängigen Dateiformaten für die Weiterbearbeitung oder Druckvorstufe (STL, OBJ) speichern. Dass die per Scan erzeugten Modelle oft viel Speicherplatz in Anspruch nehmen und trotz hoher Auflösung nicht sehr maßgetreu sind, liegt in der Natur des Scanvorgangs und der bei der Nachbearbeitung eingegangenen Kompromisse bezüglich „glatter“ Oberflächen versus Präzision. Auch

So sieht das lasergedruckte Objekt aus (Abb. 9).

lassen sich dabei nicht alle Details erfassen, vor allem wenn sie hintereinanderliegen, sodass die Kamera sie nicht aufnehmen kann. Der nächste Teil wird sich mit dem Zusammensetzen und Rekonstruieren vorhandener Modelle und der maßgenauen Konstruktion neuer Modelle mithilfe von 3D-CAD-Programmen beschäftigen. (avr@ix.de)

Glossar

Manifold (wasserdicht, Eigenschaft): ein Objekt, das sich durch eine geschlossene Fläche ohne Löcher, fehlende Punkte oder falsch herum gedrehte Flächen (Innenfläche zeigt nach außen) auszeichnet. Es beschreibt sozusagen den „Idealfall“, der sich recht problemlos mit allen Programmen weiterverarbeiten lässt.

Mesh: Ein Gitternetzmodell beschreibt die Oberfläche eines Objekts.

Punktwolke (Point Cloud, Dateiendung oft *.ply*): eine Liste von x-y,-z-Koordinaten, die die Eckpunkte von Ebenen definieren können oder einfach lose im Raum liegen.

Prof. Dipl.-Ing. Klaus Knopper

lehrt Softwareengineering am Fachbereich Betriebswirtschaft der Hochschule Kaiserslautern und forscht unter anderem im Bereich Open-Source-Systeme sowie räumlich-akustischer Repräsentation von Softwaremodellen.

Alle Links: ix.de/ix1902118