Tim C. Lueth, Franz Irlinger: "Berechnete Erzeugung von dreidimensionalen Oberflächenmodellen im STL-Format aus der Beschreibung planarer Mechanismen für die Generative Fertigung durch Selektives-Lasersintern [Computational 3D Surface Generation of Planar Mechanismus using STL File Format for Generative Manufacturing by Selective Laser Sintering]", angenommener Beitrag im Konferenzband 10. Kolloquium Getriebetechnik, TU Ilmenau, Sep. 11-13, 2013, pp 1-18.

BERECHNETE ERZEUGUNG VON DREIDIMENSIONALEN OBERFLÄCHENMODELLEN IM STL-FORMAT AUS DER BESCHREIBUNG PLANARER MECHANISMEN FÜR DIE GENERATIVE FERTIGUNG DURCH SELEKTIVES LASERSINTERN

Tim C. Lueth und Franz Irlinger**

*Technische Universität München, Lehrstuhl Mikrotechnik und Medizingerätetechnik (MIMED), Boltzmannstr. 15, Geb. 1, tim.lueth@tum.de, irlinger@tum.de

Abstract (deutsch und englisch)

Der Beitrag beschreibt detailliert wie die Bestandteile eines planaren Mechanismus als geometrische Körper modelliert und im STL-Dateiformat für den 3D-Druck gespeichert werden können. Anhand des Entwurfs eines Viergelenks wird die komplette Kette vom mathematischen Modell bis hin zum Schreiben der druckbaren STL-Datei beschrieben. Die erläuterten Beispiele können leicht in MATLAB nachvollzogen werden.

Die Autoren sind überzeugt, dass der Entwurf von Mechanismen vor einem Umbruch steht, da ein automatischer Entwurf eines Mechanismus aus einer Bewegungsanforderung möglich wird und der 3D-Druck eines solchen kostengünstig und kurzfristig möglich ist. In Kombination mit preiswerten Motoren und einfachste Steuerungen stellen generativ gefertigte Mechanismen eine Alternative zur komplexen Programmierung von Robotern dar.

This paper describes in detail how to generate STL-Files using MATLAB for the generation of Mechanisms and Kinematic structures. For a 4Bar-Linkage, all design steps to come from a kinematic description to a 3D printable spatial object are explained.

The authors are convinced that the design of mechanisms will become popular again, since the automatic design to fulfill a movement task is

possible. 3D printing of mechanisms is cheap and possible within hours. Off-the-shelf motors and simple control boards are a real alternative to a complex programming task for 6DoF robots.

1 Motivation

Die Lehre der ungleichförmig übersetzenden Getriebe ist wesentliches Werkzeug zur formalen Beschreibung von Mechanismen. Neben der Analyse derartiger Konstruktionen ist vor allem die automatisierte, d.h. berechnete Synthese einer Konstruktion zur Lösung einer Bewegungsaufgabe von Bedeutung. Hier stehen grafische und analytische Lösungsverfahren in Kombination mit einer numerischen Optimierung zur Verfügung.

Eine einmal gefundene Mechanismus/Getriebe-Lösung steht jedoch immer im Wettbewerb zur Lösung derselben Bewegungsaufgabe durch einen allgemeinen programmierbaren Bewegungsautomaten, d.h. einen Roboter. Letzterer hat den scheinbaren Vorteil, dass er im Gegensatz zu der Getriebelösung immer wiederverwendet werden kann und nur neu programmiert werden muß.

Dieser Vorteil gilt jedoch nur unter der impliziten Annahme, dass die Programmierung eines Roboters für eine Aufgabe tatsächlich einfacher ist als die Konstruktion und die Herstellung eines Bewegungsautomaten für eine Aufgabe. Gelingt es, einen Bewegungsautomaten aus einer Bewegungsaufgabe heraus vollständig automatisiert zu entwerfen und herzustellen, so dass nur noch preiswerte Antriebe angeflanscht werden müssen, verschwindet der Vorteil des Roboters. Es entstehen neue Anwendungsgebiete für die Mechanismen, dort wo die Kosten der Struktur deutlich unter den Kosten der Antriebe und der Steuerung liegen.

Der Beitrag stellt detailliert vor, wie unter ausschließlicher Nutzung von MATLAB direkt die notwendigen STL-Dateien für die generative Fertigung von Mechanismen bzw. einer vollständigen Getriebelösung automatisiert erzeugt werden können. Unabhängig von dem aktuellen Forschungsstand beschränkt sich dieser Beitrag auf planare Mechanismen.

2 Oberflächenmodellierung geometrischer Körper

In diesem Beitrag werden geometrische Körper vollständig durch die Gesamtmenge der ebenen Oberflächen des Körpers beschrieben. Dies ist die optimale Darstellungsform für die spätere Konvertierung in das STL-Dateiformat. Ein dreidimensionaler Körper besteht daher aus Eckpunkten (engl. Vertex) und ebenen Flächen (engl. Facet), die durch die Eckpunkte aufgespannt werden. Die Form der Modellierung eines Körpers durch ebene Flächen ist in der Computergrafik, beim Export aus CAD-Programmen und in der generativen Fertigung ein sehr verbreiteter Standard. Diese Modellierung wird auch direkt von MATLAB unterstützt. Sie wird Boundary Representation (BREP) genannt.

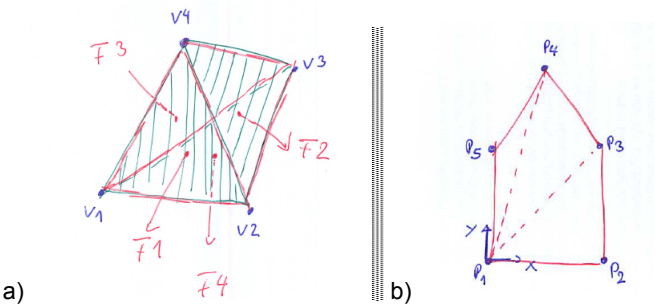


Abb. 1: a) Ein Körper wird durch die ihn umschließenden ebene Flächen (grün) beschrieben, die durch Konturen (rot) definiert sind, die durch die Eckpunkte (blau) des Körpers aufgespannt werden. b) Grundsätzlich können auch Flächen, die nicht dreieckig sind, bzw. deren eingrenzenden Konturen vollständig in dreieckige Teilflächen bzw. Dreieckskonturen unterteilt werden (engl. Tessellation).

Betrachten wir den Tetraeder aus Abb. 1, so wird schnell klar, dass wir eine Liste der Eckpunkte (Vertex-List, VL) mit den räumlichen Koordinaten benötigen. Die MATLAB-Notation lautet für eine Liste der xyz-Koordinaten von vier Eckpunkten eines Tetraeders:

$$VL = [\ x_1\ \ y_1\ \ z_1;\ x_2\ \ y_2\ \ z_2;\ x_3\ \ y_3\ \ z_3;\ x_4\ \ y_4\ \ z_4\ ]$$

bzw. $VL = [x_1\ y_1\ z_1; x_2\ y_2\ z_2; x_3\ y_3\ z_3; x_4\ y_4\ z_4]$

Für den Tetraeder aus Abb. 1a lautet die Liste beispielsweise:

```
VL=[0 0 0; 10 0 0; 10 10 0; 5 5 5];
```

Prinzipiell könnte eine ebene Fläche eine Kontur mit beliebig vielen Eckpunkten zur Eingrenzung benötigen. Zur Beschleunigung vieler Algorithmen werden jedoch grundsätzlich alle Flächen bzw. Konturen in Dreiecke bzw. Dreieckskonturen zerlegt. Diese Zerlegung (Abb. 1b) wird Tessellierung genannt.

Bei der Beschreibung der Flächen werden daher immer 3 Punkte aus der Vertex-List benötigt, um eine Fläche durch die Umlaufkontur zu beschreiben. Jede Fläche besteht aus 3 Punkten aus der Vertex List. Gibt es n Flächen ist die Flächen-Liste (Facet-List, FL) $n \times 3$ Eckpunkte lang.

$$FL = \begin{bmatrix} \mathrm{f}_1 ; \\ \mathrm{f}_2 ; \\ \vdots \\ \mathrm{f}_n \end{bmatrix} = \begin{bmatrix} v_{1,1} & v_{1,2} & v_{1,3}; \\ v_{2,1} & v_{2,2} & v_{2,3}; \\ \vdots & \vdots & \vdots \\ v_{n,1} & v_{n,2} & v_{n,3} \end{bmatrix}$$

Die Fläche 2 aus Abb. 1a beschreiben wir einfach mit [2 3 4] oder [3 4 2] oder [4 2 3]. Diese Darstellungen sind austauschbar. Mit der rechten-Hand-Regel wird gleichzeitig der Normalenvektor der Fläche definiert. Er muss immer von dem Körper weg zeigen und darf nie in den Körper hinein zeigen. Um die Flächennormale umzukehren genügt es, zweiten und dritten Index einer Fläche zu tauschen. Die Punkte [2 4 3], [3 2 4] und [4 3 2], beschreiben daher alle eine Fläche an derselben Position, jedoch zeigt hier der Normalenvektor fälschlicherweise nach innen. In MATLAB schreiben wir demnach für den Tetraeder aus Abb. 1a die Flächen-Liste:

```
FL=[1 2 4; 2 3 4; 3 1 4; 1 3 2];
```

Zur Darstellung am Bildschirm mit MATLAB genügt das Kommando *trisurf.*

```
trisurf (FL,VL(:,1), VL(:,2), VL(:,3)); axis equal; view (0,30); hold on; rotate3d
```

Um ein Objekt einzufärben (flächenbezogen oder punktbezogen), es zu beleuchten oder transparent zu gestalten, gibt es in MATLAB weitere Kommandos. Wichtig für die Darstellung ist oft noch eine Beschriftung der Eckpunkte mit dem Kommando *text*.

```
for i=1:size(VL,1); text (VL(i,1), VL(i,2), VL(i,3), sprintf ('P%i',i)); end
```

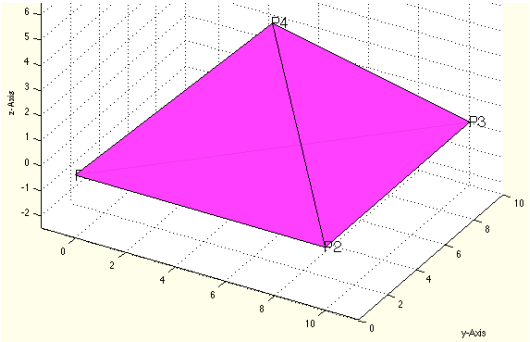


Abb. 2: Das MATLAB Kommando trisurf (FL,VL(:,1), VL(:,2), VL(:,3)) zeichnet einen Körper bestehend aus einer Vertex-List VL und einer Facet-List FL am Bildschirm.

Wir definieren für den späteren Gebrauch eine kleine MATLAB-Funktion:

```
function VLFLplot (VL,FL)
  trisurf (FL,VL(:,1), VL(:,2), VL(:,3)); axis equal; hold on;
  for i=1:size(VL,1); text (VL(i,1), VL(i,2), VL(i,3), sprintf ('P%i',i)); end
end
```

3 Generierung einer STL-Datei

Um geometrische Körper mit einem 3D-Drucker direkt auszudrucken oder die Daten zu einem Druckservice (z.B. www.shapeways.com) hochzuladen, benötigt man eine Datei im STL-Dateiformat. Diese kann man sich im ASCII-Textformat sehr einfach mit einer kleinen MATLAB Funktion selbst aus einer Vertex-List und einer Facet-List erzeugen:

```
function VLFLwriteSTL (VL,FL)
  fid=fopen ('TEST.STL','W');                                  % Open file to write
  fprintf (fid,'solid TEST\n');                                % Write first line: solid
  n=size (FL,1);                                               % n = number of facets
  for i=1:n                                                    % Loop to write all facets
    p1=VL(FL(i,1),:);  p2=VL(FL(i,2),:); p3=VL(FL(i,3),:);     % 3 vertices of the facet
    pn=cross (p2-p1, p3-p1); pn=pn/norm (pn);                  % calc the normal vector
    fprintf (fid,' facet normal %f %f %f\n', pn(1),pn(2),pn(3)); % Write normal vector
    fprintf (fid,'  outer loop\n');                            % Write: outer loop
    fprintf (fid,'   vertex %f %f %f \n',p1(1),p1(2),p1(3));   % Write 1st. vertex x y z
    fprintf (fid,'   vertex %f %f %f \n',p2(1),p2(2),p2(3));   % Write 2nd. vertex x y z
    fprintf (fid,'   vertex %f %f %f \n',p3(1),p3(2),p3(3));   % Write 3rd. vertex x y z
    fprintf (fid,'  endloop\n');                               % Write: endloop
    fprintf (fid,' endfacet\n');                               % Write: endfacet
  end                                                          % End of the loop
  fprintf (fid,'endsolid\n');                                  % Write line: endsolid
  fclose (fid);                                                % Close the file
end
```

Eine STL-Datei beginnt mit einer ersten Zeile mit dem Text *solid*. Danach werden alle Dreiecksflächen in das File geschrieben. Jede Fläche besteht aus 7 Zeilen. Die 1. Zeile jeder Fläche gibt die Koordinaten des Normalenvektors der Fläche nach der rechten Hand-Regel an:

$$\mathbf{v}_n = (p_2 - p_1) \times (p_3 - p_1); \quad \mathbf{e}_n = \mathbf{v}_n / |\mathbf{v}_n|$$

Es folgt eine 2. Zeile mit dem Text: *outer loop.* Dann folgen drei Zeilen mit den Koordinaten der drei Eckpunkte jeweils mit dem Vorspann: *vertex.* Es folgen eine 6. Zeile mit dem Text *endloop* und eine 7. Zeile mit *endfacet.* Dann folgen die anderen Flächen. Die letzte Zeile der Datei heißt: *endsolid*.

Für das Beispiel mit dem Tetraeder erhält man so die STL-Datei „TEST.STL“. Diese Datei kann mit jedem STL-Viewer angesehen werden oder auf einem 3D-Drucker ausgedruckt werden. Die Einheit für den Wert 1 wird vor dem Druck nachgefragt: 1 entspricht 1mm oder 1 entspricht 1m:

```
solid
 facet normal 0.000000 -0.707107 0.707107
  outer loop
   vertex 0.000000 0.000000 0.000000
   vertex 10.000000 0.000000 0.000000
   vertex 5.000000 5.000000 5.000000
  endloop
 endfacet
 facet normal 0.707107 -0.000000 0.707107
  outer loop
   vertex 10.000000 0.000000 0.000000
   vertex 10.000000 10.000000 0.000000
   vertex 5.000000 5.000000 5.000000
  endloop
 endfacet
 facet normal -0.707107 0.707107 0.000000
  outer loop
   vertex 10.000000 10.000000 0.000000
   vertex 0.000000 0.000000 0.000000
   vertex 5.000000 5.000000 5.000000
  endloop
 endfacet
 facet normal 0.000000 0.000000 -1.000000
  outer loop
   vertex 0.000000 0.000000 0.000000
   vertex 10.000000 10.000000 0.000000
   vertex 10.000000 0.000000 0.000000
  endloop
 endfacet
endsolid
```

Ein bekanntes kostenloses Programm zur Darstellung und Analyse von STL-Dateien ist MeshLab. Es ist für Windows und Mac-Rechner verfügbar.

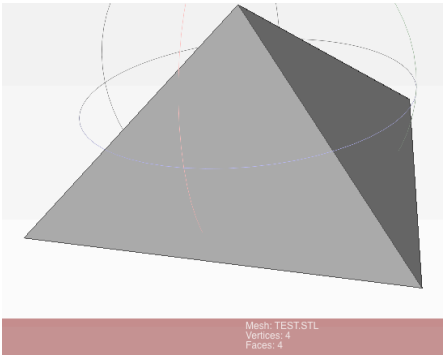


Abb. 3: Die selbsterzeugte STL-Datei kann direkt von dem STL-Anzeige- und Analyseprogramm MeshLab eingelesen werden.

4 Einfache 3D-Modellierung mit konvexen Polygonen

Um einfach planare Mechanismen entwerfen zu können, bietet es sich an, sogenannte zweieinhalb-dimensionale (2½D) Körper zu entwerfen. Diese bestehen aus einer ebenen Grundfläche, die durch eine Kontur umschrieben ist, sowie einer zweiten in z-verschobenen Deckfläche, die mit der Grundfläche identisch ist.

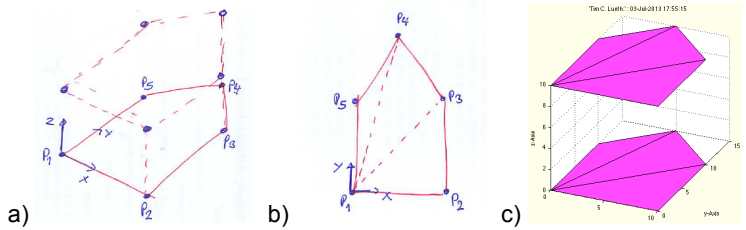


Abb. 4: a) Ein 2½D-Körper besteht aus zwei identischen übereinander angeordneten ebenen Flächen. b) Konvexe Polygone können leicht in Dreiecke zerlegt (tesselliert) werden. c) MATLAB Darstellung

Zur Beschreibung der ebenen 2D-Grundfläche verwenden wir eine Punktliste ähnlich der Vertex-List jedoch nun nur mit xy-Koordinaten: $PL = [x_1\ y_1;\ x_2\ y_2;\ \ldots; x_n\ y_n]$

Sollen daraus dreidimensionale Koordinaten werden geschieht das in MATLAB einfach für die z-Koordinate mit: $VL = [PL, z * ones(size(PL,1),1)]$

Aus dem Polygon der Grundfläche des Fünfecks aus dem Beispiel kann also schnell die Vertex-List der unteren und der oberen Ecken berechnet werden.

```
PL=[0 0; 10 0; 10 10; 5 15; 0 10]          % Punkte Liste
VLU=[PL, 0*ones(size(PL,1),1)]             % Ergänze unten z= 0 Koordinate
VLO=[PL, 10*ones(size(PL,1),1)]            % Ergänze oben z=10 Koordinate
```

Soll ein konvexes Polygon in dreieckige Flächen bzw. Polygonzüge zerlegt werden geschieht dies für n Punkte sehr einfach durch die Flächen [1 2 3], [1 3 4], [1 4 5] usw. Der erste Punkt 1 bleibt für alle Dreiecke identisch, während der zweite und dritte Punkt wandern. In MATLAB kann man die Erzeugung einer Facet-List für n Punkte durch eine einzige Zeile schreiben:

```
n=size(PL,1);        FL=[ones(1,n-2); 2:n-1; 3:n]'
```

Bei dieser Schreibweise zeigen die Normalenvektoren der Flächen nach oben. Dies würde aber nur für die Dachfläche korrekt sein. Die untere Fläche müsste die letzten beiden Spalten vertauscht haben, damit die Normalenvektoren nach unten zeigen. Die geht in MATLAB sehr elegant:

```
FLO=FL;              FLU=[FL(:,1) FL(:,3) FL(:,2)];
```

Da für den Körper alle 10 Punkte benötigt werden, kann man die Vertex-Listen hintereinander hängen. Möchte man auch die Flächen hintereinander hängen, darf man nicht vergessen, bei der oberen Flächenliste zu jedem Index die Anzahl n der Punkte hinzuzuaddieren, da oben die Indizes nicht von 1 bis n (d.h. 1..5) sondern von n+1 bis 2*n (6..10) laufen.

```
VL=[VLU;VLO];        FL=[FLU; FLO+n];
```

Für einen Vollkörper fehlen jetzt nur noch die Seitenwände. Das Prinzip ist klar: Für jedes rechteckige Seitenwandsegment benötigen wir 2 Dreiecke.

```
FLA=[1:n; 2:n 1; 2+n:n+n 1+n]'
FLB=[1:n; 2+n:n+n 1+n; 1+n:n+n]'
FL=[FLU;FLO+n;FLA;FLB]
```

Damit kann der Körper bestehend aus Vertex-List und Facet-List aus einer Polygonbeschreibung generiert und als STL-Datei gespeichert werden.

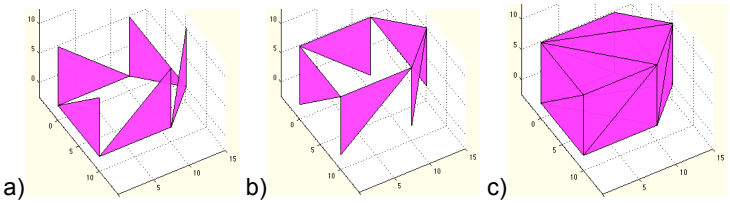


Abb. 5: a) Flächen A der Außenwand. b) Flächen B der Außenwand. c) Finale Beschreibung des Körpers durch Vertex-List und Facet-List.

Selbstverständlich wird eine solche Vorgehensweise in einer kleinen Funktion zusammengefasst.

```
function [VL,FL]=VLFLofPLz (PL,z)
   n=size(PL,1);
   VLU=[PL, 0*ones(n,1)];
   VLO=[PL, z*ones(n,1)];
   FL=[ones(1,n-2); 2:n-1; 3:n]';
   FLO=FL; FLU=[FL(:,1) FL(:,3) FL(:,2)];
   VL=[VLU;VLO];
   FLA=[1:n; 2:n 1; 2+n:n+n 1+n]';
   FLB=[1:n; 2+n:n+n 1+n; 1+n:n+n]';
   FL=[FLU;FLO+n;FLA;FLB];
end
```

5 Tessellierung verschachtelter Konturen

Bei den konvexen Polygonen haben wir uns des Tricks bedient, dass die Reihenfolge der Punkte der Punktliste gleichzeitig auch den Polygonzug definiert hat. Dies ist nicht mehr möglich, wenn wir beispielsweise die Koppel eines 4-Gelenks mit zwei Bohrungen für die Drehgelenke beschreiben wollen. Dann besteht die ebene Grundfläche aus mehreren verschachtelten Konturen. In diesem Fall müssen wir neben der Punktliste auch noch eine Kantenliste (Edge-List, EL) definieren, die angibt, welche Punkte aufeinander folgen.

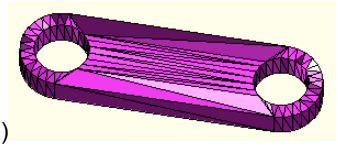
a)  b) 

Abb. 6: a) Körper mit Bohrungen benötigen zur Beschreibung der Grundfläche neben einer Punktliste PL zusätzlich noch eine Kantenliste EL, die erläutert, wie die Punkte verbunden werden müssen, um geschlossene Konturen zu definieren. b) Drei Konturen der Fläche

$$EL = \begin{bmatrix} e_1; \\ e_2; \\ \vdots \\ e_n \end{bmatrix} = \begin{bmatrix} p_{1,1} & p_{1,2}; \\ p_{2,1} & p_{2,2}; \\ \vdots & \vdots \\ p_{n,1} & p_{n,2} \end{bmatrix}$$

Ist diese Liste sinnvoll sortiert, dann beschreiben die ersten Kanten die äußerste Kontur und die nachfolgenden Kanten die Konturen, die innerhalb der ersten liegen. Unter dem Begriff *Delaunay-Triangulation* findet man in der Literatur Algorithmen zur Tessellierung von Flächen in Dreiecke. Diese Verfahren sind seit einigen Jahren Bestandteil von MATLAB. Für zweidimensionale Punktlisten und zweidimensionale Kantenlisten kann man sich die Tessellierung in Dreiecke mit drei Kommandos berechnen lassen.

```
dt=DelaunayTri(PL, EL);        % Definieren der Konturen durch Punkte und Kanten
inindex=inOutStatus(dt);       % Flächenindex außerhalb innerer Konturen
FL=dt(inindex,:);              % Flächenliste nur der äußersten Kontur
```

Mit den drei Funktionen können beliebige Flächen tesselliert werden:

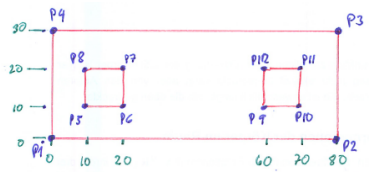
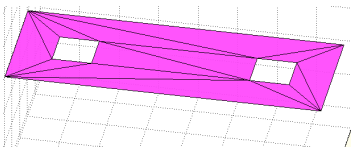
a)  b) 

Abb. 7: a) Fläche mit Bohrungen. b) Ergebnis der Delaunay-Triangulation

Für die Abb. 7 lauten Punkte-Liste (PL) und Kanten-Liste (EL) wie folgt:

PL=[0 0;80 0;80 30;0 30;10 10;20 10; 20 20; 10 20;60 10;70 10;70 20; 60 20]

EL=[1 2;2 3;3 4;4 1;5 6;6 7;7 8; 8 5;9 10;10 11;11 12;12 9]

Die Berechnung der Wandflächen geschieht analog zu den oben beschriebenen konvexen Konturen. Einziger Unterschied ist, dass die geschlossenen Konturen nacheinander abgearbeitet werden müssen und während die Wände der äußeren Kontur nach außen zeigen, müssen die Wände der Bohrungen nach innen zeigen.

Die MATLAB-Funktion zur allgemeinen Erzeugung der 2½D-Körper ist hier aus Platzgründen nicht explizit abgedruckt kann aber von den Autoren angefordert werden. Sie ist nur unwesentlich länger als die oben gedruckte.

6 Transformationen der Körper im Raum

Bekanntermaßen beschreibt eine 3D-Rotationsmatrix ${}^0\mathbf{R}_A$ in ihren drei Spalten die drei Einheitsvektoren der gedrehten Orthonormalbasis A in dem Ursprungskoordinatensystem 0. Um die Lage eines neuen Koordinatensystems A relativ zu dem Ursprungskoordinatensystem anzugeben, benötigt man dann nur noch den Translationsvektor ${}^0\mathbf{t}_A$ zum Ursprung des neuen Koordinatensystems A. Rotation und Translation werden häufig mit der homogenen Transformationsmatrix zusammengeschrieben. Das Ursprungssystem steht als Index links oben. Das Zielsystem als Index rechts unten.

$$ {}^0\mathbf{T}_A = \begin{bmatrix} {}^0\mathbf{R}_A & {}^0\mathbf{t}_A \\ \mathbf{0} & 1 \end{bmatrix} = \begin{bmatrix} \mathbf{e}_x & \mathbf{e}_y & \mathbf{e}_z & \mathbf{t} \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad mit \quad {}^0\mathbf{R}_A = \begin{bmatrix} \mathbf{e}_x & \mathbf{e}_y & \mathbf{e}_z \end{bmatrix}_A, \quad {}^0\mathbf{t}_A $$

Möchte man die Position eines Punktes gegeben durch Koordinaten im System A in das System 0 umrechnen, geschieht dies durch:

$$ {}^0\mathbf{p} = {}^0\mathbf{R}_A \cdot {}^A\mathbf{p} + {}^0\mathbf{t}_A \quad \Leftrightarrow \quad {}^A\mathbf{p} = \left({}^0\mathbf{R}_A\right)^{-1}\left({}^0\mathbf{p} - {}^0\mathbf{t}_A\right) = \left({}^0\mathbf{R}_A\right)^{T}\left({}^0\mathbf{p} - {}^0\mathbf{t}_A\right) $$

Wenn jeder Punkt um eine 4. Koordinate mit dem festen Wert 1 ergänzt wird $\mathbf{p} = \begin{pmatrix} x & y & z & 1 \end{pmatrix}$, dann können diese Gleichungen erheblich verkürzt geschrieben werden:

$$ {}^0\mathbf{p} = {}^0\mathbf{T}_A \cdot {}^A\mathbf{p} \quad \Leftrightarrow \quad {}^A\mathbf{p} = {}^A\mathbf{T}_0 \cdot {}^0\mathbf{p} $$

Wir nutzen im Weiteren nur die kürzere Schreibweise mit den T-Matrizen.

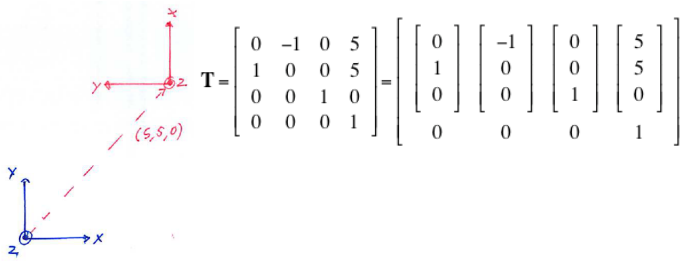


Abb. 8: Die neue Lage der Einheitsvektoren und die Verschiebung des Ursprungs kann direkt in der 4x4 Matrix abgelesen werden.

Bei einer Verschiebung von (20,20,0) statt (5,5,0) schreiben wir die Matrix aus Abb. 8 in MATLAB:

```
T=[0 -1 0 20; 1 0 0 20; 0 0 1 0; 0 0 0 1]   % Definieren eines Koordinatensystems
```

Dann definieren wir einen Körper aus dem Fünfeck aus dem Abschnitt 4 und zeichnen diesen:

```
[VL,FL]=VLFLofPLz (PL,10)                   % Erzeuge einen Körper aus einem Polygon
trisurf (FL,VL(:,1), VL(:,2), VL(:,3)); axis equal; view (0,30); hold on; rotate3d
```

Stellen wir uns vor, die Koordinaten in der VL bezögen sich auf das Koordinatensystem T, dann könnten wir die Weltkoordinaten wie folgt berechnen. Wir ergänzen eine feste 4. Koordinate 1 und transformieren die vollständige Vertex-List mit einem einzigen MATLAB-Kommando.

```
VL=[VL ones(size(VL,1),1)];                 % Ergänze eine 4 Koordinate 1
VLT=(T*VL')';                               % Transformiere alle Punkte
trisurf (FL,VLT(:,1), VLT(:,2), VLT(:,3)); axis equal; view (0,30); hold on;
```

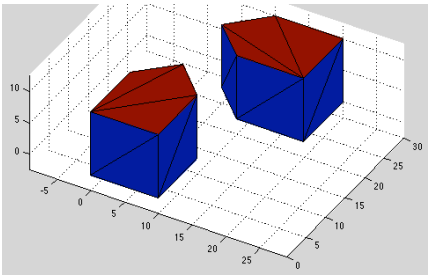


Abb. 9: Bei der Verwendung von 4x4 Matrizen können die Körper über ein kurzes MATLAB-Kommando im Raum transformiert werden.

Wir haben jetzt quasi zwei Körper:

- Definiert durch Facet-List FL und die ursprüngliche Vertex-List VL
- Definiert durch Facet-List FL und die transformierte Vertex-List VLT

7 Fusionieren mehrerer Körper in einer STL-Datei

Um mehrere Körper zu verbinden, die später gemeinsam gedruckt werden sollen, wie beispielsweise ein montiertes Viergelenk, müssen nur die entsprechenden Eckpunktlisten (Vertex-List) und Flächenlisten (Facet-List) aneinandergehängt werden. Das haben wir eigentlich bereits zur Erzeugung der Polygonkörper getan. Die Vertex-Listen werden direkt hintereinander gehängt. Die Facet-Listen werden ebenfalls hintereinander gehängt, wobei die Indizes der zweiten Liste um die Anzahl der Eckpunkte aus der ersten Vertex-List erhöht werden müssen.

```
function [VL,FL]=VLFLcat (VLA,FLA,VLB,FLB)
  VL=[VLA;VLB];
  FL=[FLA;FLB+size(VLA,1)];
end
```

Die beiden Körper aus dem letzten Beispiel würde man daher für das 3D-Drucken gemeinsam in eine STL-Datei schreiben. Ihre räumliche Lage zueinander bleibt beim Druck erhalten.

```
PL=[0 0; 10 0; 10 10; 5 15; 0 10]                  % Definiere eine Flächenkontur
[VL,FL]=VLFLofPLz (PL,10)                          % Erzeuge VLFL-Körper
T=[0 -1 0 20; 1 0 0 20; 0 0 1 0; 0 0 0 1]          % Definiere Koordinatensystem T
VL=[VL ones(size(VL,1),1)]; VLT=(T*VL')';          % Körper im Koordinatensystem T
 [VL,FL]=VLFLcat(VL,FL,VLT,FL);                    % Verbinde die beiden Körper
VLFLwriteSTL(VL,FL);                               % Schreibe STL-Datei
VLFLplot(VL,FL);                                   % Zeichne beide Körper
```

8 Berechnung eines montierten Viergelenks in STL

Mit den bereits beschriebenen Funktionen haben die Autoren drei kleine MATLAB-Funktionen erstellt, um gedruckte Viergelenke bzw. deren STL-Dateien direkt aus mathematischen Beschreibungen zu generieren. Die Verkettung bzw. „virtuelle“ Montage findet über die homogenen Transformationsmatrizen statt. Die drei Element-Funktionen lauten:

function [VL,FL]=**VLFLbolt** (RA,H)
Bolzen mit Außenradius RA und Höhe H

function [VL,FL]=**VLFLspacer** (RA,RI,H)
Hülse mit Außenradius RA, Innenradius RI und Höhe H

function [VL,FL]=**VLFLlinkage** (R,RI,H,D)
Koppel mit Außenradius RA, Innenradius RI, Höhe H und Lochabstand D in x.

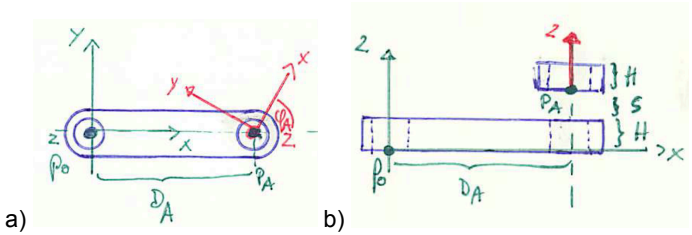


Abb. 10: a) Aufsicht auf die Koppel b) Seitenansicht mit Höhe H der Koppel und Abstand s zwischen den Koppeln. Der Punkt $\mathbf{p}_0$ ist der Ursprung der Koppel, $\mathbf{p}_A$ ist das Ende der Koppel und liegt in z höher als $\mathbf{p}_0$

Zur Berechnung der kinematischen Kette verwenden wir die homogene Transformationsmatrix aus Abschnitt 6. Für eine Koppel wie in Abb. 11 mit einem Fixpunkt $\mathbf{p}_0$ im Ursprung, einer Entfernung D_A in x-Richtung zu einer Drehachse in $\mathbf{p}_A$ lautet die beschreibende Transformationsmatrix

$$ {}^0\mathbf{T}_A = \begin{pmatrix} \cos\varphi_A & -\sin\varphi_A & 0 & D_A \\ \sin\varphi_A & \cos\varphi_A & 0 & 0 \\ 0 & 0 & 1 & H+s \\ 0 & 0 & 0 & 0 \end{pmatrix} \Rightarrow {}^0\mathbf{p} = {}^0\mathbf{T}_A \cdot {}^A\mathbf{p} \Rightarrow {}^0\begin{pmatrix} D_A \\ 0 \\ H+s \\ 1 \end{pmatrix} = {}^0\mathbf{T}_A \cdot {}^A\begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix} $$

Wenn wir uns also für die Position des Ursprungs (0,0,0) des Koordinatensystems A in den Koordinaten des Koordinatensystems 0 interessieren, müssen wir einfach nur diesen Nullvektor (und die ergänzte 4. Koordinate 1) mit der Transformationsmatrix multiplizieren.

Die Höhe H ist notwendig, da unsere Koppel in der Realität dreidimensional ist, also das nächste Glied darüber liegen muss. Der Abstand s ist notwendig, damit beim 3D-Druck die Körper nicht fest verbunden werden. Der Wert für s ist für das Selektive-Laser-Sintern für Kontaktflächen im mm-Bereich zwischen 0,3 mm (mit Kraft lösbar) und 0,5 mm (ohne Kraft lösbar).

Diese T-Matrizen zur Beschreibung der Koordinatensysteme erzeugen wir im weiteren ebenfalls über eine einfache MATLAB Funktion:

function T=**TofDPhiH** (D,Phi,H)

Nach diesen Vorüberlegungen können wir jetzt ein Viergelenk direkt aus der mathematischen Beschreibung heraus räumlich berechnen und in Kunststoff drucken. Die Aufgabe lautet: Drei Glieder A (60 mm), B (20 mm), C (30 mm) jeweils im Winkel von 60 Grad ($\pi/3$) zu verbinden und dann ein viertes Glied D zu entwerfen. Wir nehmen als Höhe 8 mm und als Spaltmaß 0,5 mm.

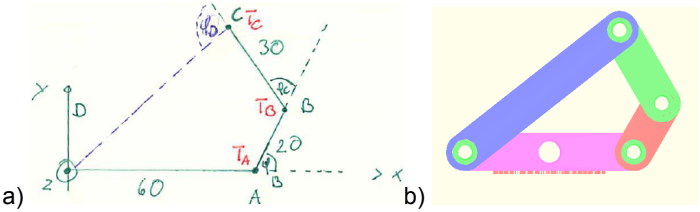


Abb. 11: a) Auslegung eines Viergelenks, b) räumliche Umsetzung

Im ersten Schritt definieren wir die Koordinatensysteme:

TA=**TofDPhiH** (60, pi/3, 8+0.5) % 60 mm lang, 60º Drehung, 8.5 Abstand

TB=TA***TofDPhiH** (20, pi/3, 8+0.5) % 20 mm lang, 60 Grad Drehung

TC=TB***TofDPhiH** (30, 0 , 8+0.5) % 30 mm lang, Drehung ist unwichtig

Die Länge D_D (70 mm) des vierten Glieds D berechnen wir aus dem Translationsvektor von TC. Ebenso wird der Winkel φ_D mit dem arctan aus dem Translationsvektor von TC berechnet. Die Matrix TD im Ursprung kann dann mit TD= TofDPhiH (0, φ_D , 3*(8+0.5)) berechnet werden.

Im zweiten Schritt werden dann die geometrischen Körper berechnet:

[VLA,FLA]= **VLFLlinkage** (8,4,8,60);

[VLB,FLB]= **VLFLlinkage** (8,4,8,20); VLB=(TA*[VLB ones(1,size(VLB,1))']')'

[VLC,FLC]= **VLFLlinkage** (8,4,8,30); VLC=(TB*[VLC ones(1,size(VLC,1))']')'

[VLD,FLD]= **VLFLlinkage** (8,4,8,DD); VLD=(TD*[VLD ones(1,size(VLD,1))']')'

Im dritten Schritt werden dann die geometrischen Körper verbunden:

[VL,FL]=**VLFLcat** (VLA,FLA,VLB,FLB)

[VL,FL]=**VLFLcat** (VL,FL,VLC,FLC)

[VL,FL]=**VLFLcat** (VL,FL,VLD,FLD)

Im vierten Schritt, der hier nicht beschrieben wird, ergänzen wir dann Bolzen und Abstandshülsen analog durch Erzeugung und Translation.

Im fünften Schritt, zeichnen wir den Körper, schreiben ihn in eine STL-Datei und drucken ihn beispielsweise im SLS-Verfahren als montiertes Viergelenk.

VLFLplot (VL,FL)

VLFLwriteSTL (VL,FL)

a) b)

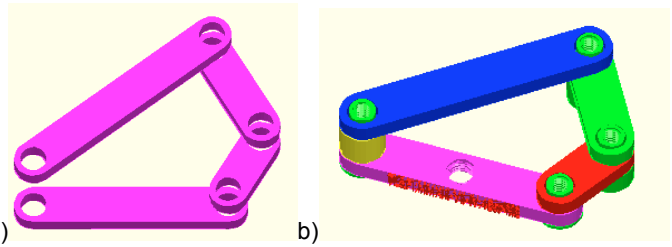


Abb. 12: a) Durch Transformationen angeordnete vier Gelenkglieder. b) Zusammensetzung eines Viergelenks aus den drei Komponenten Bolzen, Hülse und Glied. Die gezeigte Kinematik kann direkt am Bildschirm angezeigt und aus MATLAB in 3D-gedruckt werden.

9 Alternative Berechnung

Zur Vollständigkeit sei erwähnt, dass man dieselbe Berechnung auch mit einem veränderten Ansatz zur Erstellung der Transformationsmatrix hätte durchführen können. Dabei wird die Drehachse in den Ursprung gelegt.

a) b)

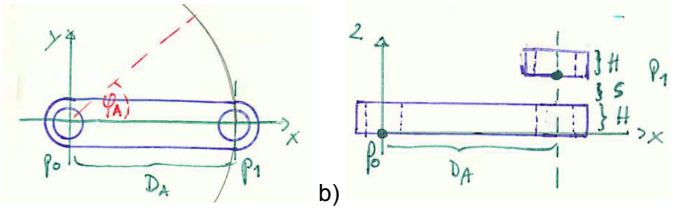


Abb. 13: a) Aufsicht auf die Koppel b) Seitenansicht mit Höhe H der Koppel und Abstand s zwischen den Koppeln. Der Punkt $\mathbf{p}_0$ ist der Ursprung der Koppel, $\mathbf{p}_1$ ist das Ende der Koppel und liegt in z höher als $\mathbf{p}_0$

Liegt der linke Drehpunkt $\mathbf{p}_0$ der Koppel im Ursprung, dann beschreibt in Abhängigkeit von dem Drehwinkel φ_A und dem Abstand D_A zum zweiten Drehpunkt die Matrix ${}^0\mathbf{T}_A$ das Koordinatensystem dieses zweiten bzw. folgenden Drehpunkts $\mathbf{p}_1$

$$ {}^0\mathbf{T}_A = \begin{pmatrix} \cos\varphi_A & -\sin\varphi_A & 0 & D_A\cdot\cos\varphi_A \\ \sin\varphi_A & \cos\varphi_A & 0 & D_A\cdot\sin\varphi_A \\ 0 & 0 & 1 & H+s \\ 0 & 0 & 0 & 0 \end{pmatrix} \Rightarrow {}^0\mathbf{p} = {}^0\mathbf{T}_A \cdot {}^A\mathbf{p} \Rightarrow {}^0\begin{pmatrix} D_A\cdot\cos\varphi_A \\ D_A\cdot\sin\varphi_A \\ H+s \\ 1 \end{pmatrix} = {}^0\mathbf{T}_A \cdot {}^A\begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix} $$

Bei dieser Darstellung, die auch Vorteile besitzt, muss man jedoch daran denken, dass die Eckpunkte der Vertex-Listen des entsprechenden Glieds nur um φ_A gedreht und nicht noch um den Translationsanteil des Glieds verschoben werden. Fazit: Zur Berechnung von Kinematiken ist diese Darstellung möglicherweise eleganter. Die Transformation der Vertex-Listen der Glieder ist nicht ganz so elegant.

Danksagung

Die Autoren wünschen allen viel Freude beim zukünftigen „Drucken“ von Forschungsergebnissen der Getriebelehre.

Wir möchten an dieser Stelle den früheren Getriebelehrern des heutigen Lehrstuhls MIMED der TU München danken bzw. die Erinnerung an deren Leistung für die Getriebelehre wachhalten. Es waren die Professoren in Folge: F.A. Klingenfeld (1868-1880), W. Marx (1880-1886), Ludwig E. Burmester (1887-1912), S. Finsterwalder (1912-1931), G. Marx (-1940), H. Wögerbauer (1940-1945), Rudolf Beyer (1948-1960), R. Unterberger (1955-1977) und Joachim Heinzl (1978-2005).

Weiterer Dank gilt der Deutschen Forschungsgemeinschaft für die Bewilligung einer Anlage (EOS) zum Selektiven Lasersintern.