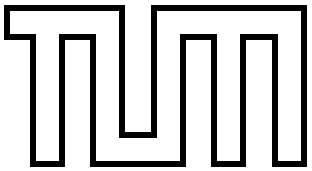


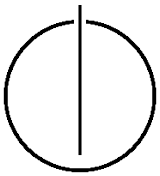
FAKULTÄT FÜR INFORMATIK

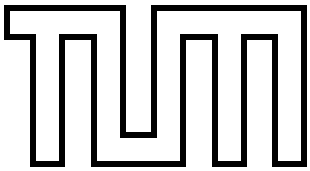
DER TECHNISCHEN UNIVERSITÄT MÜNCHEN

Master's Thesis in Informatik

Entwicklung von Werkzeugen zur automatischen Erkennung geometrischer Strukturen in triangulierten CAD-Daten

Philipp Hübner





FAKULTÄT FÜR INFORMATIK

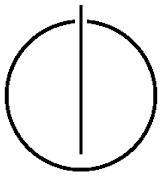
DER TECHNISCHEN UNIVERSITÄT MÜNCHEN

Master's Thesis in Informatik

Entwicklung von Werkzeugen zur automatischen Erkennung geometrischer Strukturen in triangulierten CAD-Daten

Development of tools for automatic recognition of geometric structures in triangulated CAD data

Autor:	Philipp Hübner
Betreuer:	Prof. Dr. Rüdiger Westermann
Abgabedatum:	6. Januar 2014



Ich versichere, dass ich diese Masterarbeit selbstständig verfasst
und nur die angegebenen Quellen und Hilfsmittel verwendet habe.

München, 6. Januar 2014 Philipp Hübner

Abstract

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Listings

1 Einleitung

> Es stimmt nicht, daß die Kosten die Preise bestimmen. Die im Markt erzielbaren Preise definieren die Kosten, die man sich leisten kann.
>
> *(Rainer Megerle (*1949), Chef Mergele AG, Nürnberg)*

1.1 Motivation

Das grundlegende Bestreben jedes Unternehmens ist es, seine Existenz am Markt zu sichern. Eine konjunkturell angespannte Wirtschaftslage, zunehmende Globalisierung der Märkte, höherer Konkurrenzdruck und kontinuierlich steigende Anforderungen nach qualitativ hochwertigen Produkten stellen für die Unternehmen der Automobilindustrie wachsende Herausforderungen dar.

Unsichere Absatzprognosen, ein erschwerter Zugang zu Kapital und ein stark erhöhter Preis- und Wettbewerbsdruck sind Probleme, mit denen sich ein Automobilhersteller 2013 konfrontiert sieht. Darüber hinaus erhöhen eine Vielzahl von international präsenter Unternehmen den Konkurrenzdruck, den Bedürfnissen des Kunden gerecht zu werden.

Um diese Situation zu verbessern, geben die Automobilhersteller den Kostendruck an ihre Zulieferer weiter. So wirkt sich der Qualitäts - und Kostendruck und damit die Notwendigkeit, ein Bauteil schon während der Konstruktion fertigungsgerecht und somit kosteneffizient zu gestalten, direkt auf die Automobilzulieferer aus. Hieraus entsteht bei diesen Firmen die Nachfrage nach softwaretechnischen Werkzeugen, mit denen sich die Fertigungskosten eines Bauteils einfacher abschätzen lassen.

1.2 Problemstellung

In der Automobilindustrie wird die Konstruktion sämtlicher Bauteile eines Fahrzeugs und ihr Zusammenbau zu Baugruppen mit Hilfe von Computer Aided Design-Systemen (CAD) realisiert. Dies hat unter anderem den Vorteil, das wichtige Untersuchungen bereits in einem frühen Stadium der Produktentwicklung durchgeführt werden können. Das können beispielsweise dynamische Simulationen des Fahrzeugverhaltens oder eben Kostenanalysen für Bauteile sein.

Zur Kostenanalyse von CAD-Bauteilen ist es notwendig, eine Reihe von Eigenschaften, die mit dem Fertigungsverfahren variieren, zu erfassen. Mit Fertigungsverfahren oder Fertigungsart sind eine Reihe von hintereinander ausgeführten Prozessen gemeint, mit

denen ein Produkt aus anderen Gütern produziert wird. Beispielsweise wird beim Spritzgießen erhitzter Kunststoff in einen Hohlraum (Formwerkzeug) gespritzt, in welchem er erst verdichtet wird und dann erkaltet. Bei mit Spritzguss gefertigten Bauteilen sind Eigenschaften, die die Fertigungskostem maßgeblich erhöhen (sogenannte Kostentreiber) zum Beispiel die minimale und maximale Dicke des Körpers oder die Anzahl der sogenannten Hinterschnitte. Die Erfassung dieser Eigenschaften sind für den Benutzer teilweise sehr aufwendig, so das eine softwaretechnische Unterstützung wünschenswert ist. Im Rahmen dieser Arbeit sollen Werkzeuge entwickelt werden, wie bestimmte kostentreibende geometrische Strukturen, sogenannte Rippen und Dome, innerhalb von spritzgegoßenen Bauteilen einfacher ermittelt werden können. Im folgenden Abschnitt 1.3 wird kurz auf die das Spritzguß Fertigungsverfahren eingegangenen und die Anwendung GoCart vorgestellt, in die die im Rahmen dieser Arbeit entwickelten Werkzeuge integriert wurden.

1.3 Kostenabschätzung für spritzgussgefertigte Bauteile

Das Spritzgießsen (oft umgangssprachlich auch als Spritzguss oder Spritzgussverfahren bezeichnet) ist ein sogenanntes Urformverfahren, das hauptsächlich in der Kunststoffverarbeitung, aber auch beispielsweise beim Pulverspritzgießen in der Metallverarbeitung eingesetzt wird.

Mit Hilfe dieses Verfahrens lassen sich direkt nutzbare Teile in großer Stückzahl herstellen. Dazu wir der jeweilige verflüssigte Werkstoff in Pulverform in eine Werkzeugform eingespritzt. Diese Werkzeugform definiert die Form und die Oberflächenstruktur des zu fertigen Bauteils. Mit dem Spritzgußverfahren lassen sich Bauteile mit sehr geringen Toleranzen, also sehr hoher Fertigungsgenauigkeit, in großen Mengen wirtschaftlich produzieren.

Spritzgießsen ist allerdings ein aufwendiger Fertigungsprozess, der nur bei hohen Stückzahlen wirtschaftlich ist. Hierbei machen die Kosten der Werkzeughüllen ein großen Teil der Aufwände aus. So ist die Rentabilität von spritzgußgefertigen Bauteilen erst bei einer Stückzahl von einigen tausend Teilen erreicht. Maßgeblich für die Kosten der Werkzeughüllen sind unter anderem die Anzahl der Dom- und Rippenstrukturen des Bauteils.

1.3.1 Das Projekt GoCart

Der praktische Teil der Masterarbeit wurd im Auftrag der Firma Teraport GmbH erstellt. Dieses Unternehmen beschäftigt sich mit der Entwicklung einer Reihe von modularen Softwarebausteinen (dem DMU-Toolkit), mit denen sich computergeometische Untersuchungen an triangulierten Fahrzeugdaten durchführen lassen. Dies sind zum Beispiel die Berechnung von Montagepfaden oder die Überprüfung von umfangreichen Geometrien auf Kollisionen. Die Werkzeuge der Teraport GmbH beinhalten auch die Visualisierungsanwendung, in die Komponenten

Dome

Händisches Ausmessen von Domstrukturen

Das händische Messverfahren zur Platzierung eines parametrischen Zylinders erfordert zunächst die Eingabe einer beliebigen Anzahl von Punkten auf der planaren Oberfläche der Domstruktur. Hierfür wurde grafisches Werkzeug entwickelt, das es dem Benutzer erlaubt, eine Reihe von dreidimensionalen Kugelobjekten durch Linksklick im 3D-Ansichtsfenster zu erzeugen. Die Koordinate der jeweiligen Kugel wird bestimmt, indem der 3D-Selektionsmechanismus (eng. *Picking*) der Anwendung genutzt wurde, um den Schnittpunkt eines Strahls vom Klickpunkt im Ansichtsfenster auf eine Objektgeometrie zu bestimmen.

Diese Kugeln müssen vom Benutzer kreisförmig auf der Zylinderkappe angeordnet werden. Durch Rechtsklick kann dann ein interaktiver parametrischer Messzylinder, mit dem Höhe und Radius manipuliert werden kann, erzeugt werden. Das Zylinderobjekt ist an der Normalen der sogenannten *Best-Fitting Plane* ausgerichtet. Eine *Best-Fitting Plane* für eine dreidimensionale Punktemenge ist diejenige Ebene, die den aufsummierten orthogonalen Abstand der Punkte zur Ebene minimiert. Sie wird sowohl für die halbautomatische Erkennung von Domstrukturen als auch für die Segmentierung der Bauteile, die in Abschnitt 4 beschrieben wird, verwendet. Im Rahmen der Masterarbeit wurde das bestehende, auf Singulärwertzerlegung basierende Verfahren, durch Hauptkomponentenanalyse ersetzt. Es wird im Abschnitt 2.3.1 näher beschrieben.

Der Zylinder kann so im Schwerpunkt der Punktwolke und an der Normalen der Best-Fitting Plane ausgerichtet erzeugt werden. Als voreingestellter Radius wird der Abstand des am weitesten zum Schwerpunkt entfernten Punktabstand genutzt. Der Benutzer kann dann mit Hilfe eines Transformationsmanipulators die Höhe und den Radius weiter anpassen.

Rippen

Verstärkungsrippen sind ein wirksames Hilfsmittel, um die Steifigkeit und Festigkeit von Spritzgußteilen zu erhöhen. Der richtige Einsatz von Rippen kann Material und Gewicht einsparen, die Spritzzyklen verkürzen und dicke Querschnittbereiche vermeiden helfen, die beim Spritzgießen zu Problemen führen könnten. Wenn Einfallstellen auf der einer Rippe gegenüberliegenden Seite nicht akzeptabel sind, können sie durch strukturierte Oberflächen oder andere geeignete Unterbrechungen im Bereich der Einfallstelle kaschiert werden.

Händisches Ausmessen von Rippenstrukturen

1.4 Grundlagen der verwendeten computergrafischen Techniken

Dreidimensionale Objekte können in einem Rechner nur visualisiert und verarbeitet werden, wenn ihre Struktur durch ein geeignetes Modell beschrieben wird. Ein einfaches sol-

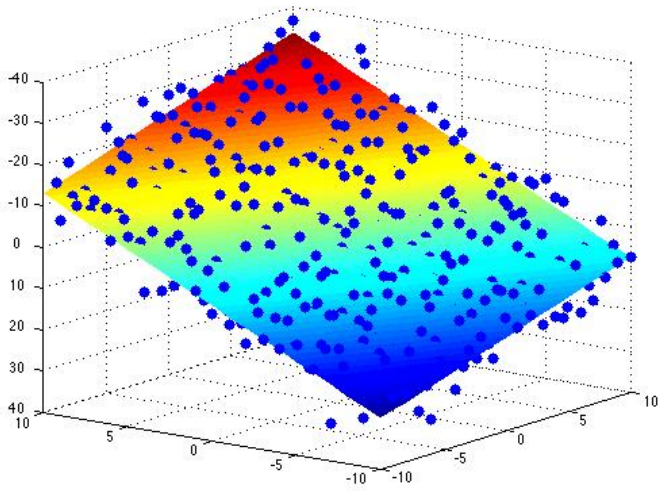


Abbildung 1.1: Händische Erzeugung eines parametrischen Zylinders.

ches Modell, auf denen die im GoCart-Projekt verarbeiteten Daten basieren, sind trinagulierte Geometrien. Sie werden im folgenden Abschnitt 1.4.1 näher beschrieben.

In computergrafischen Systemen ist es oft erforderlich, den minimalen Abstand zwischen zwei geometrischen Objekten zu bestimmen. Um dies zu realisieren, sind effiziente Kollisionsprüfungsroutinen notwendig, auf die im Abschnitt 1.4.2 eingegangen wird.

In Abschnitt 1.4.3 werden kurz dreidimensionale Auswahlverfahren beschrieben.

1.4.1 Triangulierte Geometrien

Eine einfache Möglichkeit zur Reprsentation von geometrischen Körpern sind Dreiecksnetze, sogenannte Triangulierungen. Eine Triangulierung approximiert die Oberfläche eines Körpers durch (sehr kleine) Dreieckeflächen. Ein Dreieck ist ein planares Polygon und wird durch drei sogenannte Scheitelpunkte, oder eng. Vertices, definiert. Jeder dieser Scheitelpunkte beschreibt mittels x-,y- und z-Koordinaten eine Position im dreidimensionalen Raum. Triangulierungen sind ein Spezialfall polygonaler Flächenrepräsentation geometrischer Körper, die Objekte mit Hilfe von planaren Vielecken (Polygone) beschreiben. Das Verfahren funktioniert exakt für ebene Strukturen, bei gekrümmten Oberflächen entsteht jedoch ein sogenannter Tesslierungsfehler, der umso stärker ausfällt, je weniger Dreiecke für die gekrümmte Oberfläche verwendet werden. In Abbildung 1.2 ist dieses Verhalten deutlich zu erkennen: Die Triangulierung mit der höheren Dreiecksanzahl wirkt an der Silhouette wesentlich runder als die mit geringerer Anzahl.

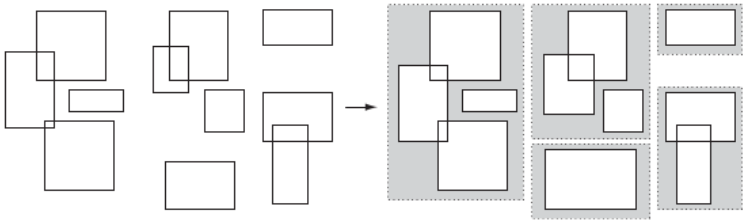


Abbildung 1.2: Triangulierung eines Zylinders mit unterschiedlicher Dreiecksanzahl.

Eine Triangulierung speichert neben den Punktdaten noch weitere Information für jedes Dreieck, wie beispielsweise Farbwerte oder sogenannte Normalenvektoren. Das sind Vektoren, die senkrecht auf der Netzoberfläche stehen und vom Körper „weg „ zeigen. Es wird zwischen Dreiecksnormalen und Punktnormalen unterschieden: Eine Dreiecksnormale ist der Vector, der orthogonal zur Dreicksfläche nach „aussen„ zeigt. Die Punktnormale ist die Mittelung der Dreiecksnormalen der drei Dreiecke, die an einen Scheitelpunkt angrenzen. Normalen haben in der Computergrafik verschiedenste Verwendungen, beispielsweise sind sie für essentiell für die Simulation von Oberflächen in der Bildsynthese.

Eine Triangulierung kann gespeichert und maschinell verarbeitet werden, indem für jedes Dreieck die drei Scheitelpunkte explizit gespeichert werden. Dieses Verfahren ist allerdings äusserst unpraktisch, da zum einen Punkte mehrfach gespeichert werden und zum anderen keine Möglichkeit existiert, einfach benachbarte Dreiecke zu finden. Ein besseres Verfahren sind sogenannte indizierte Polygonmengen, eng. *Indexed Face Set*. Hierbei werden alle Punkte in einer Liste gehalten. Ein einzelnes Dreieck kann dann durch drei Indizes in diese Liste beschrieben werden. Auf diese Weise ist zwar keine redundante Speicherung von Punktdaten mehr nötig, das Problem der fehlenden Nachbarschaftsbeziehungen besteht auch bei dieser Repräsentation.

Dieses Problem wird beispielsweise mit der sogenannten *Winged-Edge* Datesstruktur gelöst, die die gesamte Topologie durch Ecken beschreibt, die Referenzen auf benachbarte Flächen und Scheitelpunkte beinhaltet.

CAD-Fahrzeugmodelle liegen liegen zunächst in triangulierter Form vor. Diese kann mit Hilfe sogenannter Triangulierungungsverfahren aus der parametrischen Repräsentation (z.b. Nurbs oder Bezier-Patches), mit der innerhalb von CAD-Systemen gearbeitet wird, erzeugt werden. Die Algorithmen zur Triangulierung können unter Umständen fehlerhaft triangulierte Dreiecksnetze erzeugen. Oft auftretende Probleme sind beispielsweise Selbstüberschneidung von Dreiecken, kleine Löcher, doppelte Scheitelpunkte oder sogenannte T-Scheitelpunkte. T-Scheitelpunkte sind Scheitelpunkte, die auf einer Dreieckskante liegen (siehe Abbildung 1.3).

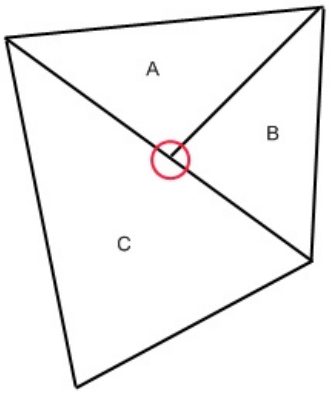


Abbildung 1.3: Triangulierung mit T-Scheitelpunkt.

Einige Fehler wie doppelte Punkte oder kleine Löcher lassen sich durch verschiedene Bereinigungsverfahren beheben, andere, wie T-Scheitelpunkte, nicht.

1.4.2 Kollisonsprüfung

Unter Kollisionsprüfung versteht man im Allgemeinen das Erkennen von Berührungen oder Überlappungen zweier oder mehrerer geometrischer Objekte im zwei- oder dreidimensionalen Raum. Ein geometrisches Objekt ist hier ein Körper, der durch ein Polygonnetz oder ein Freiformflächenmodell beschrieben werden kann.

Es existieren eine Reihe weiterer Anwendungen, für die eine effiziente Kollisionsprüfung unbedingt erforderlich ist:

- Beim *Virtual Prototyping* wird die Baubarkeit der transformierten Einzelkomponenten des Prototyps durch Kollisionstests gewährleistet [10].
- In der *Pfad- und Bewegungsplanung* gilt es, für einen Roboter mit beliebigen Freiheitsgraden einen Weg von einem Start- zu einem Zielpunkt zu finden, ohne das dieser mit Hindernissen kollidiert [8].
- *Starrkörper-Physiksimulationen* führen Kollisionsprüfung aus, um in jedem Zeitschritt der Simulation zu erfassen, ob ein dynamisches Objekt mit einem anderen Objekt der Szene kollidiert. So können natürliche Verhaltensweisen starrer Körper, wie beispielsweise das voneinander Abprallen von Billardkugeln, simuliert werden.

Diese Anwendungen stellen unterschiedliche Anforderungen an ein Kollisionsprüfungssystem. Zum einen müssen in kürzerster Zeit Überlappungen zwischen dynamischen Objekten erkannt werden, zum anderen muss eine große Menge Geometrie verarbeitbar sein. Ein einfacher Ansatz, die Kollisionen einer Szene zu finden, ist, alle Dreiecke paarweise gegeneinaner zu testen. Dies führt jedoch (bei n Szenenobjekten) zu quadratischem Aufwand:

$$\frac{n * (n-1)}{2} \in \mathcal{O}(n^2) \tag{1.1}$$

So sind keine echtzeitfähigen Überlappungstests realisierbar! Daher muss die Berechnungzeit mit Hilfe von Optimierungsmethoden minimiert werden. Eine Idee, den Aufwand zu reduzieren, ist, einen Divide-and-Conquer Ansatz zu verwenden [1].

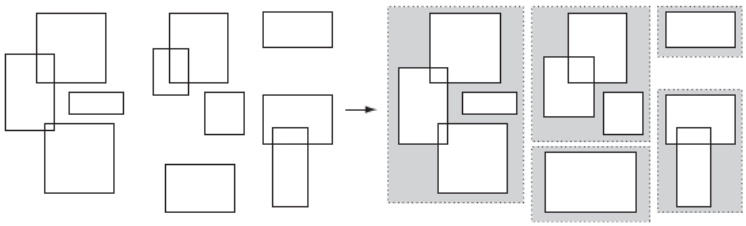


Abbildung 1.4: Erkennung von disjunkten Objekten.

In Abb. 1.4 sind auf der linken Seite um 11 Rechtecke auf Kollision zu prüfen 55 Tests notwendig. Nachdem in der „weiten Phase“ 5 disjunkte Teilmengen erkannt wurden, kann die Anzahl der Tests in der „nahen Phase“ auf 10 reduziert werden.

Ein Ansatz, die Anzahl der nötigen expliziten Dreieckstests zu reduzieren, ist, anstelle der Dreiecke zunächst einen Hüllkörper (*Bounding Volume*, kurz: *BV*) zu testen. Ein Hüllkörper ist ein einfaches geometrisches Objekt wie beispielsweise ein Würfel oder eine Kugel, das das Objekt komplett umhüllt. Die Idee ist, auszunutzen, dass die Schnitttests solcher Körper im Vergleich zu den eingehüllten Objekten weniger aufwendig sind. So können zunächst die Bounding Volumes zweier Objekte auf Kollision getestet werden und nur dann, wenn dieser Schnitttest positiv verläuft, muss die Dreiecksmenge der Objekte in der „nahen Phase“ explizit geprüft werden.

Die Nutzung von Hüllkörpern reduziert den quadratischen Aufwand der Kollisionspüfung einer Szene mit n Dreiecken soweit um einen konstanten Faktor k:

$$\forall k \in [0,1] : k * \frac{n * (n-1)}{2} \in \mathcal{O}(n^2) \tag{1.2}$$

Obwohl hierdurch der Rechenaufwand signifikant verringert werden kann, verbessert dies die Komplexitätsklasse des naiven Ansatzes nicht; die Anzahl der nötigen paarweisen Schnitttests zwischen den Hüllen wächst noch immer quadratisch bezüglich der Anzahl der Szenenobjekte.

Eine Möglichkeit, den Aufwand weiter zu reduzieren, ist die Nutzung von Hierachien aus Hüllkörpern (sogenannten *Bounding Volume Hierachien* oder *BVH*). Dieser Ansatz wurde 1996 in dem Artikel „*OBBTree: A Hierarchical Structure for Rapid Interference Detection*“ vorgestellt und hat seit dem die Forschung an Kollisionsprüfungsalgorithmen stark beeinflußt [4]. BVH's betten die Hüllkörper der Objekte rekursiv in größere BV's ein und ordnen diese in einer Baumstruktur an. So kann die Zeitkomplexität des Algorithmus auf logarithmischen Aufwand verringert werden. Abbildung 1.5 zeigt eine Hierachie weltachsenorientierter Hüllquader, mit der fünf Objekte eingehüllt werden.

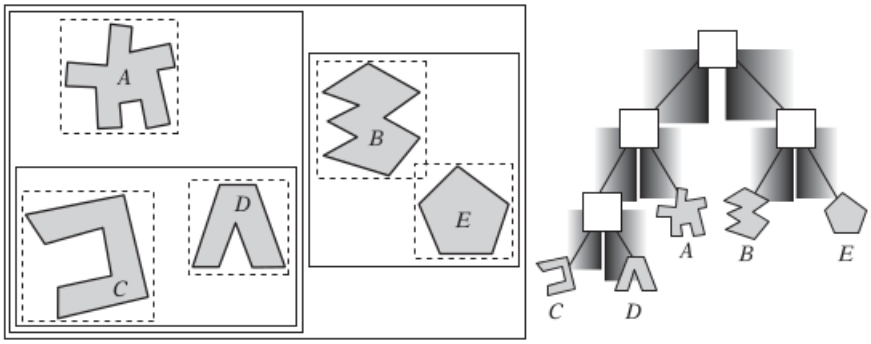


Abbildung 1.5: Bounding Volume Hierachie aus 5 Objekten. Auf der linken Seite wird die Szene rekursiv umhüllt so dass die Baumstruktur auf der rechten Seite daraus resultiert.

Der Ansatz rekursive Hüllkörper in einer Baumstruktur anzuordnen, lässt sich auch auf die Dreieckesmenge eines Objektes ausweiten. Hierbei wird die Dreiecksmenge eines Objektes als Punktewolke interpretiert, die dann mittels Top-Down Ansätzen so lange weiter unterteilt wird, bis ein Abbruchkriterium erreicht ist. Abbildung 1.6 zeigt eine solche Hierachie aus AABB's über der Dreiecksmenge eines Objektes.

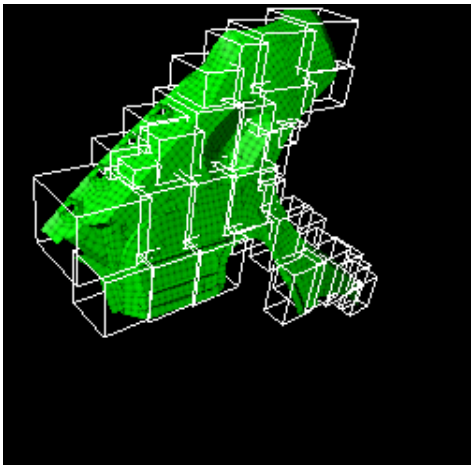


Abbildung 1.6: Bounding Volume Hierachie über den Dreiecken eines Objektes.

Um eine Szene, für die eine Hüllkörperhierachie erzeugt wurde, auf Kollision zu testen, muß die Hierachie traversiert werden. Auf jeder Baumebene, die während der Traversierung besucht wird, werden die BV's der Knoten gegeneinander auf Kollision geprüft. Wenn eine Überschneidung zwischen zwei Bounding Volumes gefunden wurden, wird die Baumsuche in diesen Ästen fortgesetzt. Auf diese Weise wird rekursiv bis in die Blattknoten vorgegangen. Sollten die Hüllen zweier Blattknoten überlappen, kann eine Kollision zwischen den eingehüllten Dreiecken bestehen, die dann in der „Narrow Phase" aufzulösen ist.

Im Rahmen dieser Arbeit wurde eine properitäre Implementierung der frei verfügbaren Bibliothek Proximity Query Package (PQP) verwendet [9].

1.4.3 3D-Selektion

Bei computergrafischen Anwendungen ist es oft notwendig, einen geometrischen Körper innerhalb einer Szene mit dem Mauszeiger zu selektieren.

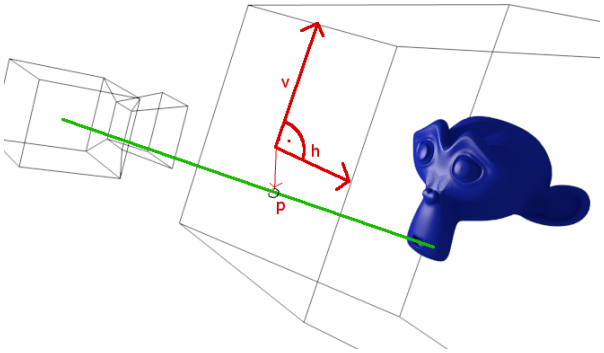


Abbildung 1.7: Bounding Volume Hierachie über den Dreiecken eines Objektes.

Eine typische Methode, dies umzusetzen, ist, einen Strahl vom Klickpunkt auf dem 3D-Ansichtsfenster der Anwendung über die virtuelle Kamera in die Szene zu projizieren und zu testen, ob der Strahl eine Geometrie schneidet (siehe Abbildung 1.7). Dieses Verfahren wird als eng. *Ray Casting* bezeichnet. Es hat den Vorteil, das es keine Methoden der zugrundeliegenden Grafikbibliothek benutzt und sich somit in jeder grafischen Anwendung auf die gleiche Weise verwenden lässt.

Um zu vermeiden, Schnittests zwischen jedem Dreieck der Szene und dem Teststrahl durchzuführen, werden bei der Selektion mittels Ray Casting die Objektgeometrien, ähnlich wie bei der Kollisonsprüfung, durch Hüllkörper ersetzt.

1.5 Zielsetzung der Arbeit

1.6 Aufbau der Arbeit

2 Halbautomatische Erkennung zylindrischer Strukturen

Ein Bild sagt mehr als 1000 Worte.

(deutsches Sprichwort)

2.1 Motivation

In diesem Kapitel soll das im Rahmen der Masterarbeit entwickelte Verfahren zur halbautomatischen Erfassung von zylindrischen Strukturen beschrieben werden. Ziel dieses Verfahrens ist es, die Aufwände für das Ausmessen von Domstrukturen auf Spritzgußteilen maßgeblich zu verringern (siehe Abschnitt 1.5).

Ein Dom ist, wie oben im Abschnitt 1.3.1 erläutert wurde, ein zylindrischer Vorsprung an einem Bauteil, der häufig ein Befestigungselement aufnimmt. Da solche Dome die Fertigungskosten eines Spritzgußteils erheblich steigern, ist es zur Abschätzung dieser Kosten notwendig, sowohl die Gesamtanzahl, als auch den Radius und die Höhe der einzelnen Zylinderstrukturen zu erfassen.

Das händische Messverfahren, das im Abschnitt 1.3.1 beschrieben wurde, erlaubt das Messen der Domparameter durch das Platzieren eines parametrischen Zylinders auf einer Domstruktur und des Definierens der Zylinderhöhe und des Radius durch grafische Manipulatoren. Der parametrische Körper wird also durch die Höhe, den Radius und die sechs Freiheitsgrade seiner Starrkörpertransformation[1] eindeutig beschrieben. Eine zulässige Betrachtungsweise der Erfassung einer Domstruktur ist somit die Bestimmung von sinnvollen Werten für diese acht Parameter. Um das Messverfahren zu beschleunigen wurde ein Werkzeug entwickelt, das einerseits die Verschiebungs- und Rotationswerte durch Analyse der Geometrie in der Umgebung einer Benutzereingabe und anderseits Höhe und Radius über heuristische Optimierung ermittelt.

Im Folgenden Abschnitt 2.2 wird kurz allgemein auf heuristische Optimierungsverfahren eingegangen. Danach wird in den Abschnitten 2.3 und 2.4 beschrieben, wie die Parameter des Optimierungsproblems im einzelnen ermittelt werden. Abschnitt **??** geht auf die Implementierung und die Ergebnisse der Vorgehensweise ein.

[1] Starrkörper besitzen sechs Freiheitsgrade: Sie können in drei Raumrichtungen verschoben und um die drei Weltkoordinatensystemachsen rotiert werden.

2.2 Heuristische Optimierung

Ein Optimierungsproblem im mathematischen Sinn ist die Aufgabe, für eine Funktion eine Menge aus Eingabewerten zu finden, so das der Funktion einen minimalen (oder maximalen) Wert annimmt, wobei in der Regel eine Beschränkung der Eingabewerte vorliegt. Die beste Vorgehensweise hierfür hängt von der Art der Bewertungsfunktion ab. Ist diese beispielsweise linear, lassen sich solche Probleme mit Hilfe von Verfahren, die auf dem sogenannten Simplex-Algorithmus basieren, effizient lösen.

Handelt es sich nicht um lineare Zielfunktionen, sind analytische Lösungsverfahren meistens sehr ineffizient und das systematische Durchsuchen des Löesungsraums ist aufgrund der Problemgröße nicht möglich. Für solche Probleme werden in der Informatik oft heuristische Optimierungsverfahren eingesetzt. Diese Verfahren generieren Lösungen anhand einer definierten Vorgehensweise und verbessern diese suzessive, bis ein Abbruchkriterium, beispielsweise die Anzahl der Iterationen, erreicht ist.

Im Gegensatz zu analytische Verfahren beinhaltet heuristische Optimierung immer die Nutzung von Zufallswerten, beispielweise beim Simulated Annealing, das im folgenden kurz erläutert wird, die initialen Punkte im Lösungsraum. Dies hat zu Folge, das heuristische Optimierungsverfahren nichtdeterministisch sind, also das bei komplexen Problemen mit mehreren lokalen Optima unterschiedliche Lösungen gefunden werden.

Im Folgenden werden drei heuristische Optimierungsverfahren kurz erläutert: *Simulated Annealing*, *Evolutionäre Algorithmen* und *Partikelschwarmoptimierung*.

Simulated Annealing (SA, siehe [6]) imitiert das Abkühlverhalten von Metallen. Kühlen diese langsam ab, haben die Atome ausreichend Zeit, sich zu ordnen und eine stabile Kristallstruktur zu bilden. Der Algorithmus betrachtet einen Punkt im n-dimensionalen Lösungsraum, der schrittweise in Richtung eines Optimums bewegt wird. Vor jeder Bewegung wird die Zielposition bewertet: Fällt die Bewertung besser aus, als die aktuelle Position, wird die Bewegung durchgeführt. Fällt sie schlechter aus, wird sie nur mit einer gewissen Wahrscheinlichkeit durchgeführt, die mit jedem Schritt sukzessive verringert wird.Diese Wahrscheinlichkeit entspricht der Temperatur des metallurgischen Abkühlungsprozesses und dient dazu, lokale Minima zu überwinden.

Evolutionäre Algorithmen **(EA)** sind eine Familie von Optimierungsverfahren, die sich das Prinzip der Selektion des Bestangepassten („*Survival of the fittest*“) der natürlichen Evolution zum Vorbild nehmen. Hierbei wird ein Punkt im Lösungsraum als Individuum und eine Menge von Punkten als Population bezeichnet. Zu Beginn des Optimierungsverfahren wird der Lösungsraum mit einer randomisierten Population „bevölkert“, die zunächst mit einer Zielfunktion (in diesem Zusammenhang oft Fitness-Funktion) bewertet werden. Die am schlechtesten bewerteten Individuen werden verworfen, wärend aus den Verbliebenen die nächste Population durch Anwendung von sogenannten „Evolutionären Operatoren“ erzeugt wird, beispielsweise Mutation (leichtes Verändern eines Individuums) und Rekombination (Mischen der Gene zweier Individuen, oft auch als Crossover bezeichnet). Dieser Zyklus aus Selektion, Crossover und Mutation wird solange wiederholt, bis ein vorgegebenes Abbruchkriterium (z. B. Anzahl der Generationen) erreicht ist. Es werden die Hauptforschungsrichtungen Evolutionäre Programmierung ([2]), Genetische Algorithmen ([3]) und Genetische Programmierung ([7]) unterschieden, die sich bezüglich der Datenstrukturen, Selektionsmethoden und der Operatoren unterscheiden.

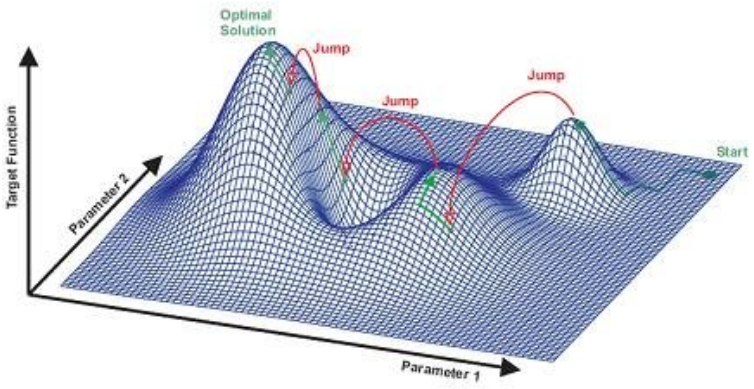


Abbildung 2.1: Simulated Annealing in einem zweidimensionalen Lösungsraum

Als ***Partikelschwarmoptimierung*** **(PSO)** wird ein Optimierungsverfahren bezeichnet, das nach dem Vorbild natürlichen Schwarmverhaltens von beispielsweise Fischen oder Vögeln eine Lösung für das Optimierungsproblem sucht. Die hier beschriebene Version der PSO wurde zuerst 1995 in einer Forschungsarbeit von J. Kennedy and R. Eberhart präsentiert [5]. Ähnlich den evolutionären Algorithmen wird bei der Partikelschwarmoptimierung initial eine zufällige Menge vom Punkten im Lösungsraum, die hier als Partikel bezeichnet werden, erzeugt. Der Partikelschwarm wird nun schrittweise durch den Lösungsraum bewegt, wobei benachbarte Partikel vor jedem Schritt die beste Position, die ein Partikel bisher gefunden hat, austauschen können. Welche Partikel miteinander kommunizieren können wird durch die sogenannte Nachbarschaftstopologie bestimmt. Ein Partikel hält somit zusätzlich zur Position $\vec{x}$ noch eine Bewegungsrichtung $\vec{v}$, die bisher beste gefundene Position als $\vec{p}_{best}$ und die beste Position seiner Umgebung als $\vec{n}_{best}$. Nach jedem Zeitschritt wird die Position und die Bewegungsrichtung (der Vektor $\vec{v}$ ist nicht normalisiert, er beinhaltet die Geschwindigkeit und Richtung) eines Partikels anhand folgender Formeln angepasst:

$$\vec{v}_{i,t+1} = \omega \cdot \vec{v}_{i,t} + c_1 \cdot \vec{U}_1[0,1] \bigotimes (\vec{p}_{best,i,t}) - \vec{x}_{i,t}) + c_2 \cdot \vec{U}_2[0,1] \bigotimes (\vec{g}_{best,i,t}) - \vec{x}_{i,t}) \quad (2.1)$$

$$\vec{x}_{i,t+1} = \vec{x}_{i,t} + \vec{v}_{i,t+1} \quad (2.2)$$

Hierbei ist t der Interationszähler, $\vec{U}_k[0,1]$ Zufallsvektoren und die Parameter ω, c_1, c_1 bestimmen den Einfluß der Position des vorigen Zeitschritts und der kognitiven und sozialen Komponenten.

Im Rahmen dieser Arbeit wird heuristische Optimierung benutzt, um für einen bereits ausgerichteten Zylinder Höhe, Radius und Rotation zu ermitteln. Dieses Problem ist nicht seperabel, d.h. einzelne Dimensionen des Lösungsraums können nicht oder nur begrenzt unabhängig voneinander untersucht werden. Darüber hinaus gibt es im Lösungsraum des Problems mehrere lokale Optima; es ist multimodal.

Für die Implementierung des Optimierungsproblems wird Partikelschwarmoptimierung eingesetzt. Hierfür wird ein Framework der Firma Teraport benutzt, das es ermöglicht, verschieden Optimierungsverfahren zu nutzen und Zielfunktionen zu implementieren.

2.3 Ermittlung der Position und Ausrichtung einer Domstruktur

Der relativ aufwendige Prozess des händischen Ausmessens einer Domstruktur, der in Abschnitt 1.3.1 beschrieben wurde, kann zunächst beschleunigt werden, indem der Benutzer anstatt eine Punktwolke zu definieren, nur ein Dreieck der Oberfläche selektiert. Ausgehend von diesem Dreieck wird dann ein sogenannter Region Growing Prozess gestartet, der sukzessive benachbarte Dreiecke, die eine bestimmte Eigenschaft besitzen, einer Suchmenge hinzufügt. Algorithmus 1 beschreibt diese Vorgehensweise im allgemeinen.

Algorithm 1: Region Growing

```
Data: Dreieck D_S
Result: R: Menge an Dreiecken mit einer bestimmten Eigenschaft
Erzeuge Stack S;
Füge D_S zu S hinzu;
while S ist nicht leer do
    Nehme oberstes Element D vom Stack S;
    Erzeuge Liste N_D der Nachbarn von D;
    for alle Dreiecke D_i aus N_D do
        if D_i erfüllt die Zieleigenschaft und D_i ist nicht in R enthalten then
            Lege D_i auf S;
        end
    end
    Erzeuge Liste N_D der Nachbarn von D;
    Füge D zu R hinzu;
end
```

Die gesuchte Eigenschaft ist hier eine ähnlich ausgerichtete Dreiecksnormale. Auf diese Weise kann, wenn der Benutzer ein Dreieck der planaren Zylinderoberseite selektiert, mit einem Klick die ganze Kappe ausgewählt werden. Als Position des zu platzierenden Zylinders wird der Schwerpunkt der Zylinderkappe genutzt, der sich folgendermaßen berechnen läßt:

$$v = \frac{\sum_i a(v_i)v_i}{\sum_i a(v_i)} \tag{2.3}$$

Hier ist v_i der Dreiecksmittelpunkt und $a(v_i)$ die Fläche des Dreiecks. Anschließend können die Scheitelpunkte der Dreiecke als Punktwolke interpretiert werden, für die mit Hilfe der Hauptkomponentenanalyse eine Best-Fitting Plane erzeugt werden, deren Normale die Rotation des Zylinders definiert.

2.3.1 Best-Fitting Plane mit Hauptkomponentenzerlegung

Eine *Best-Fitting Plane* ist diejenige Ebene, die den aufsummierten orthogonalen Abstand einer Menge von Punkte zur Ebene minimiert (siehe Abbildung 2.2). Eine Ebene kann geometrisch auf verschiedene Weise beschrieben werden. Im Rahmen dieser Arbeit wird die sogenannte Hessische Normalenform zur Definition von Ebenen genutzt, die hierfür einen Punkt P und einen Normalenvektor $\vec{n}$ benötigt.

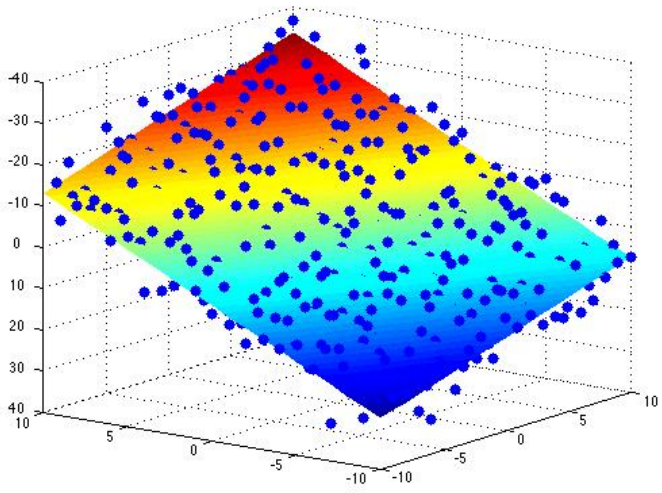


Abbildung 2.2: Best Fitting Plane für eine dreidimensionale Punktwolke.

Die *Best-Fitting Plane* wird in dieser Arbeit mit der klassischen Methode, die auf der Hauptkomponentenanalyse (eng *Principal Components Analysis*, kurz PCA) basiert, ermittelt. Dieses Verfahren basiert auf der Kovarianzmatrix Cov_p, die folgendermaßen aus der Punktemenge p_i erzeugt wird:

$$Cov_p = \sum_i a(p_i)(p_i - p)(p_i - p)^T \tag{2.4}$$

Der Punkt P der Ebenengleichung ist der Schwerpunkt der Punktewolke P. Er wird durch das gewichtete Mittel der Ortsvektoren $\vec{p}_i$ der Punkte berechnet.

$$P = \frac{\sum_i \vec{p}_i}{i} \tag{2.5}$$

Die Ebenennormale $\vec{n}$ kann mittels Eigenvektorzerlegung von Cov_v bestimmt werden: sie entspricht dem Eigenvektor, der mit dem kleinsten Eigenwert von Cov_v korrespondiert. Wenn der kleinste Eigenwert 0 ist, bedeutet das, die Punktemenge P koplanar ist. Da Eigenwertzerlegungen relativ aufwendig zu implementieren sind, wird hierfür eine Bibliotheksfunktion verwendet. Der Vektor $\vec{n}$ kann einfach in eine Rotationsmatrix M_R umgewandelt werden.

Das Verfahren kann einfach auf eine Menge von Dreiecken angewandt werden, indem die Vertices der Dreiecke als Punktewolke interpretiert werden. Der zu ermittelnde Zylinder kann dann im (gewichteten) Schwerpunkt (eng. *Center of Cravity*, kurz COG) P_T der Dreiecksmenge platziert werden.

$$P_T = \frac{\sum_i a(p_i)\vec{p}_i}{\sum_i a(p_i)} \tag{2.6}$$

Die Starrkörpertransformation (also die Positionierung und Ausrichtung im 3D-Raum) für den Zylinder ist durch von P_T und M_R vollständig bestimmt. Die Ermittlung der Höhe und des Radius wird dann als mehrdimensionales Optimierungsproblem aufgefasst, für dessen Lösung *PSO* eingesetzt wird.

2.4 Zielfunktion des Optimierungsproblems

Kernidee der Zielfunktion ist die Nutzung einer virtuellen Messlehre. Eine Lehre ist in der Technik ein Gerät, das für vorher festgelegte Maße und Formen ein Bezugsnormal darstellt, beispielsweise ein Messschieber oder ein Haarwinkel. Diese virtuelle Lehre ist ein degenerierter parametrischer Zylinder, der möglichst nah an die zu prüfende Bauteilgeometrie geschoben werden soll, ohne mit dieser zu kollidieren. Im folgenden wird der Messzylinder *Prüfzylinder*, die Bauteilgeometire *model* genannt.

Der Prüefzylinder besteht aus drei degenerierten Dreiecken, die zylindrisch angeordnet sind und in Höhe, Radius und Drehwinkel parametrisiert sind. Dies hat zum einen den Vorteil, das Kollisonsberechnungen mit drei degenerierten Dreiecken wenig aufwendig sind, und zum anderen, das mittels der Prüfstäbe auch mit Stützrippen ausgestattete Dome korrekt vermessen werden können (siehe Abbildung 2.4). Er wird initial korrekt ausgerichtet im Dreiecksschwerpunkt des Zylinderdeckels positioniert.

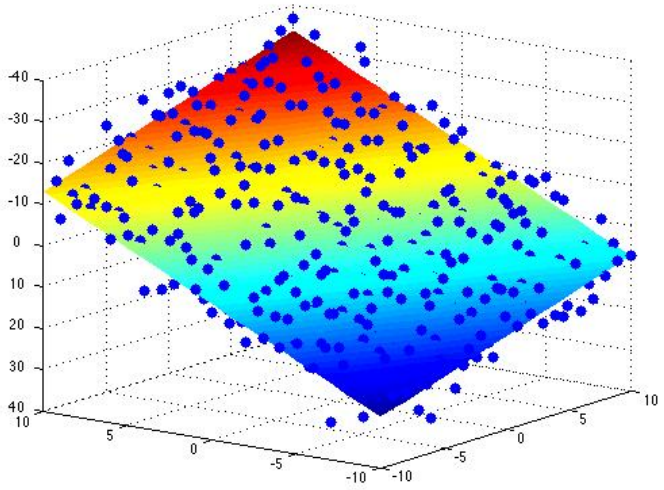


Abbildung 2.3: Parametrischer Messzylinder für Bauteilgeometrie.

Kollisionsvermeidung

Bei der Positionierung des Prüfzylinders ist es vor allem wichtig, das die Prüfgeometrie und das Modell nicht kollidieren. Hierfür wird eine die properitäre Implementierung der freien Kollisionsbibliothek Proximity Query Package (PQP) verwendet. Die Kolliosonsprüfung wird in Form eines Strafwertes in Bewertungsfunktion integriert: Kollidiert der Prüfzylinder mit dem Model wird ein sehr hoher Strafwert zurückgegeben.

Ermittlung von Höhe und Radius

Wenn die Prüfgeometrie und das Model nicht kollidieren, werden die Werte für Radius, Höhe und Drehwinkel so interpretiert, das der Radius r minimiert und die Höhe h maximiert wird. Folgende Gleichung bestimmt den Rückgabewert V bei nicht koolidierenden Geometrien:

$$V = \lambda_1 r + \lambda_2(-h) \tag{2.7}$$

Hierbei sind λ_1 und λ_2 Gewichtungsfaktoren, die den Radius oder die Höhe stŕker werten können. Auf diese Weise kann das System so konfiguriert werden, das Dome mit Phasen an der Unterseite variabel erfasst werden können:

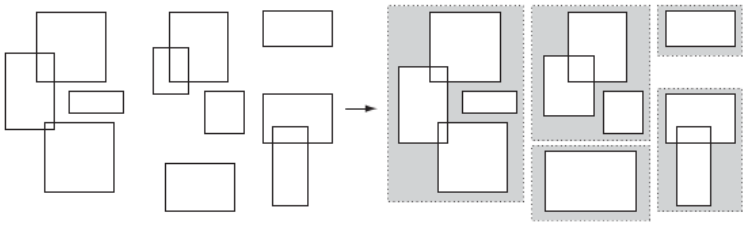


Abbildung 2.4: Angephaster Zylinder mit unterschiedlichen λ-Werten.

Algorithmus 2 fasst die Implementierung Zielfunktion noch einmal zusammen:

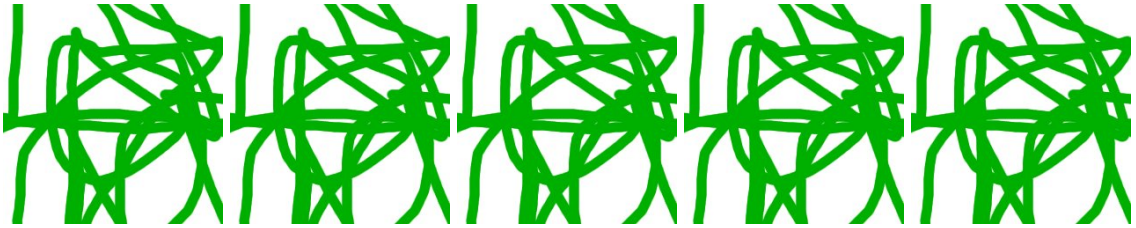
Algorithm 2: Zielfunktion PSO

Data: Bauteilgeometrie *Model*, Höhe h, Radius r, Winkel a,λ_1, λ_2
Result: Höhe und Radius
Erzeuge Prüfgeometrie mit h, r, a;
Erzeuge PQP-Modell mit Prüfgeometrie;
Prüfe Prüfgeometrie und Modell auf Kollison;
if *Prüfgeometrie und Modell kollidieren* **then**
  return 10^9;
end
return $V = \lambda_1 r + \lambda_2(-h)$

2.5 Ergebnisse

Zur Evaluierung des implementierten Werkzeugs wurden einige Versuche mit verschieden Bauteilen durchgeführt. Bei diesen Bauteilen handelt es sich zum einen um originale Fahrzeugdaten, zum anderen um eigens für die Domanylse konstruierte Modelle. Bei den folgenden Tests beziehen sich die Laufzeiten auf einen Intel Core2 Duo 6600 mit 2.4 GHz und 3 GB RAM. Als Betriebssystem kam Windows XP (32 Bit), Service Pack 3 zum Einsatz.

Abbildung 2.5 zeigt verschiedene Domtypen, die mit dem implementierten Verfahren unterschiedlich gut erfasst werden können:



(a) einfacher Dom (b) Seitenrippen (c) Schräge Unterseite (d) Schräge Oberseite (e) Runde Oberseite

Abbildung 2.5: Verschiedene Domtypen

Ein einfacher Dom wie in der Abbildung *a* ganz links kann problemlos erfasst werden; abhängig von den Werten für λ_1 und λ_2 wird die Bodenphase mit eingefasst oder nicht. Dome mit Seitenrippen wie in Abbildung *b* kommen häufig vor und können gut erfasst werden (siehe Abbildung 2.4). Bei Domen mit schräge Unterseiten funktioniert das Verfahren teilweise: Je nach Drehwinkel können die Messstäbe der Prfgeometire bis zur unteren Kante reichen. Da die Erfassung der Domhöhen nur im Bereich von 10mm Schritten erfolgt, ist diese Ungenauigkeit in der Praxis akzeptabel.

Dome mit schräger Oberseite können nur für sehr geringe Neigungswinkel korrekt erfasst werden, denn die Ausrichtung des Zylindersdeckels wird benutzt, um den Prüfzylinder zu positionieren. Dies funktioniert nur, solange die Zylinderoberseite orthogonal zur Mittelachse der Domgeometrie steht (siehe Abbildung 2.6).

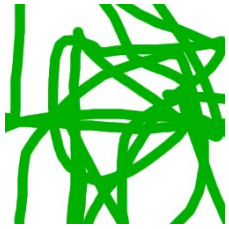


Abbildung 2.6: Dome mit schräger Oberseite.

Ein Dom mit runder Oberseite können mit dem implementierten Verfahren nicht erfasst werden. Hier schlägt die Ermittlung der Position und der Rotation fehl, da das im Abschnitt 2.3 vorgestellt wurde, keine planare Dreiecksmenge bestimmen kann.

Für die ersten drei in Abbildung 2.5 gezeigten Dome wurden mit dem umgesetzten Verfahren folgende Laufzeiten benötigt:

	händisches Ausmessen	halbautomatisches Verfahren
Bauteil *a*	12s	2300ms
Bauteil *b*	8s	2500ms
Bauteil *c*	8s	2200ms

2.6 Fazit und Ausblick

Es wurde gezeigt das das vorgestellte halbautomatische Messverfahren eine wesentliche Verbesserung gegenüber der händischen darstellt, da die Eigenschaften der am häufigsten vorkommenden Domtypen in einem Bruchteil der Zeit erfasst werden können. Für Domstrukturen mit planarer Oberseite findet der Region-Growing Ansatz die Position und die Ausrichtung des Prüfzylinders stabil und der gewählte Ansatz der heuristischen Optimierung löst das restliche Optimierungsproblem zuverlässig. Die Positionierungsergebnisse für den Prüfzylinder wurden visuell überprüft und waren durchgängig plausibel. Es gab keine Durchdringungen und offensichtlich bessere Werte für Höhe und Radius war nicht auszumachen.

Das Verfahren funktioniert nur teilweise fr̈ Domstrukturen mit schr̈agem Boden. Dies könnte insofern erweitert werden, dass die drei Messstäbe des Prüfzylinders mit jeweils einem Höhenparameter versehen werden. Diese Parameter können als weitere Dimensionen für das Optimierungsproblem aufgefasst werden, wobei die Zielfunktion um diese drei Werte erweitert werden müsste. Das Optimierungsverfahren würde eine Stellunng des Prüfzylinders liefern, in der ein Messstab die maximale Höhe liefert.

Bei Domen mit runden Oberseiten kann mit dem halbautomatischen Messwerkzeug kein Ergebnis gefunden werden. Dies kann gelöst werden, indem man die gesamte Starrkörpertransformation als achtdimensionales Optimierungsproblem auffasst. Dabei ist das Problem zu lösen, dass die Dimensionalität des Suchraums weiter ansteigt und damit der Aufwand für die Berechnung drastisch erhöht wird.

3 Segmentierung von triangulierten Geometrien

Das Ganze ist mehr als die Summe seiner Teile.

(Aristotels)

3.1 Motivation

3.2 Notation und Klassifizierung von Segmentierungstechniken

3.3 Geometrische Eigenschaften und Unterteilungskriterien

3.4 Hierachisches Clustering

3.5 Filtern und Verfeinern der Segmentstruktur

4 Halbautomatische Rippenerkennung

Das Ganze ist mehr als die Summe seiner Teile.

(Aristotels)

4.1 Ermittlung der Rippenoberfläche

4.2 Nachbearbeiten der ermittelten Rippenoberfläche

4.3 Ermittlung der Rippentiefe

4.4 Implementierung und Ergebnisse

Literaturverzeichnis

[1] Christer Ericson. *Real-Time Collision Detection*. Morgan Kaufmann, 2005.

[2] A.Ji; Walsh M.J Fogel, L.J.; Owens. *Artificial intelligence through simulated evolution*. Wiley and Sons, New York, USA, 1st edition, 1966.

[3] D.E Goldberg. *Genetic algorithms in search, optimization, and machine learning*. Addison-Wesley, New York, USA, 1st edition, 1989.

[4] Stefan Gottschalk. *Collison Queries using Oriented Bounding Boxes*. Chapel Hill, 2000.

[5] R.C. Kennedy, J.; Eberhart. Particle Swarm Optimization. In *Proc. of IEEE Int. Conf. on Neural Networks*, pages 1942–1948, 1995.

[6] S. Kirkpatrick. Optimization by simulated annealing. *Science*, pages 671–680, 1983.

[7] R.J Koza. *Artificial intelligence through simulated evolution*. MIT Press, 1992.

[8] Steven M. LaValle. *Planning Algorithms*. Cambridge University Press, 2006.

[9] UNC Research Group on Modeling Physically-Based Simulation andApplications. Pqp - a proximity query package. http://www.cs.unc.edu/~geom/SSV/, 05/2009.

[10] Gabriel Zachmann. *Virtual Reality in Assembly Simulation — Collision Detection, Simulation Algorithms,and Interaction Techniques*. PhD thesis, Technische Universität Darmstadt., December 2010.