

Haptisches Feedback eines Roboters durch virtuelle 3D-Modelle

Carl Gathmann
Universität zu Lübeck

Lübeck, Germany
carl.gathmann@student.uni-luebeck.de

Marten Buchmann
Universität zu Lübeck

Lübeck, Germany
marten.buchmann@student.uni-luebeck.de

Abstract—In dieser Studie wird ein Ansatz zur Erzeugung von haptischem Feedback im Bereich der Mensch-Roboter-Interaktion vorgestellt. Mittels Software wird eine sensorische Wahrnehmung simuliert, die durch frei gestaltbare, virtuelle 3D-Modelle gesteuert wird. Der Anwender erfährt eine Art abweisende Kraft, die durch die räumlichen Grenzen des virtuellen Modells definiert ist. Dadurch ist es möglich eine physische Interaktion mit dem immateriellen Modell nachzuahmen. Die vorgestellte Technologie findet viele Anwendungsbereiche. Von medizinischen Simulationen und Exploration in gefährlichen Zonen bis hin zur Erhöhung der Spielerfahrung in virtuellen Umgebungen.

Index Terms—Robotik, Haptisches Feedback

I. EINLEITUNG

Die Mensch-Roboter-Interaktion hat in den letzten Jahren zunehmend an Bedeutung gewonnen und sich als ein dynamisches Forschungs- und Anwendungsfeld etabliert. Im Zentrum dieser Interaktion steht die Verbesserung der Benutzererfahrung durch die Erweiterung der sinnlichen Wahrnehmung des Menschen.

In dieser Studie wird eine Methode präsentiert, der es dem Benutzer ermöglicht, virtuelle 3D-Modelle zu "ertasten", indem er über ein Roboterinterface mit diesen interagiert. Dies wird durch die Erzeugung einer abweisenden Kraft erreicht, die auf den Oberflächen der virtuellen Modelle basiert. Dieses haptische Feedback simuliert das physische Berühren eines realen Objekts.

Dieser Ansatz bietet zahlreiche Anwendungsmöglichkeiten, darunter die Simulation von Operationen für Ausbildungszwecke, die Erkundung von Objekten in gefährlichen oder unzugänglichen Umgebungen und die Erhöhung der Immersion in virtuellen Spielen. Die in dieser Arbeit vorgestellten Prinzipien und Implementierungen können als Grundlage für weiterführende Forschungen und Entwicklungen auf diesem Gebiet dienen.

Das vorliegende Paper beleuchtet die zugrundeliegenden Prinzipien, technische Details und potenzielle Anwendungen dieses Ansatzes.

II. METHODEN

Die Erzeugung des haptischen Feedbacks beruht auf der Berechnung einer abweisenden Kraft, die so konzipiert ist, dass dem Benutzer ein realistisches Gefühl für die Form des virtuellen 3D-Modells vermittelt wird. Die Modelle können als

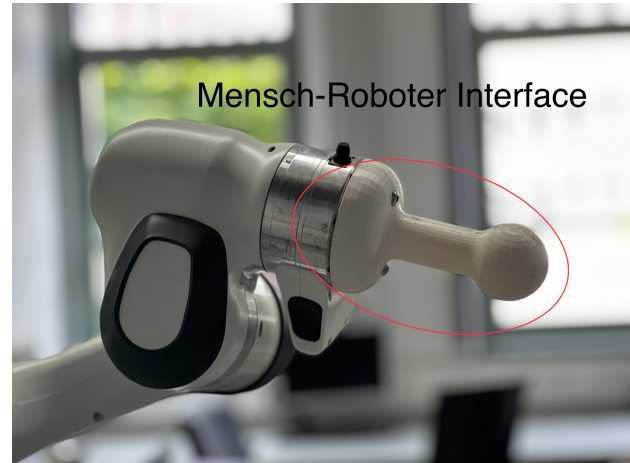


Fig. 1. Mensch-Roboter-Schnittstelle

Datei im STL-Standard übergeben werden. Der STL-Standard speichert 3D-Modelle als Dreiecksmesh, das aus Eckpunkten und den Normalenvektoren der Dreiecke besteht. Es ist keine besondere CAD-Software nötig um 3D-Modelle im STL Format zu erstellen. Außerdem ist das Format weit verbreitet und wird in der Industrie zur Speicherung und Übertragung von 3D-Modellen eingesetzt. Das sorgt für hohe Flexibilität und einfache Handhabung [1].

Als Roboterarm wurde Panda-Roboter von Franka Emika ausgewählt. Der Panda verfügt über 7 DOF und Kraftsensoren in allen sieben Achsen. So bietet der Manipulator eine hohe Präzision und Flexibilität, aber auch einen gewissen Sicherheitsstandard [2]. Eine benutzerfreundliche, 3D-gedruckte Schnittstelle (siehe 1) ist am Endeffektor des Roboters installiert, um die Führung durch menschliche Benutzer zu erleichtern.

Bei der entwickelten Software für dieses System ist die Leistungsfähigkeit äußerst kritisch, um eine nahtlose Benutzererfahrung zu gewährleisten. Es ist entscheidend, dass die Anwendung in Echtzeit ausgeführt werden kann, wobei eine maximale Verzögerung von weniger als 1 Millisekunde akzeptiert wird. Die Programmiersprache C++ eignet sich aufgrund ihrer hohen Performanz, um diesem Anspruch gerecht zu werden. Die Hardware-Konfiguration besteht aus einem Intel Core i7-3770 und 16GB RAM. Ein STL-Parser extrahiert

die Eckpunkte der einzelnen Dreiecke aus der STL-Datei und speichert diese in Matrizen. Damit eine einfache sowie effiziente Handhabung der 3D-Daten innerhalb des Programms möglich bleibt, werden Datentypen und Funktionen aus der "Eigen" Bibliothek verwendet.

Im folgenden Teil werden verschiedenen Ansätze zur Erstellung eines stetigen und kontinuierlichen Vektorfeldes aus einem Dreiecksmesh vorgestellt.

Für die Verwendung der Ansätze II-A und II-B muss vorerst das zweidimensionale- zu einem dreidimensionalen Problem reduziert werden. Erreicht wird dies durch die Erstellung eines neuen Koordinatensystems für jedes Dreieck, welches sich ergibt durch:

- 1) Eine Kante des Dreiecks als x-Achse.
- 2) Das Kreuzprodukt der entstandenen Achse mit einem weiteren Vektor als z-Achse.
- 3) Das Kreuzprodukt der beiden entstandenen Achsen als y-Achse.

Die Koordinaten des Punktes werden dann in diesem Koordinatensystem angegeben. Auf diese Art repräsentieren die x- und y-Koordinaten die Position und die z-Koordinate die Entfernung des Punktes von der Ebene des Dreiecks.

A. Edge-Function

Die "Edge-Function" oder "Kanten Funktion" ist eine häufig verwendete Funktion im Bereich der Computergrafik. Sie wird verwendet, um herauszufinden, wo sich ein Punkt relativ zu einer Kante befindet [3]. Definiert ist sie wie folgt:

$$E_i = (x_{i+1} - x_i)(y - y_i) - (y_{i+1} - y_i)(x - x_i) \quad (1)$$

x, y sind die Koordinaten des Punktes, x_i , y_i sind die Koordinaten des i-ten Eckpunktes des Dreiecks. Im Fall eines Dreiecks mit den Eckpunkten A, B und C und einem Punkt P ist der Punkt innerhalb des Dreiecks, wenn gilt:

$$E_{AB} \geq 0 \wedge E_{BC} \geq 0 \wedge E_{CA} \geq 0 \quad (2)$$

Mit dem Wissen, ob ein Punkt innerhalb des Dreiecks liegt und der Höhe des Punktes, die über die z-Koordinate des transformierten Punktes gegeben ist, kann ein Kraftvektor berechnet werden. Dieser Kraftvektor zeigt in Richtung der Normalen des Dreiecks und ist proportional zur Höhe des Punktes. Die entsprechende Formel ergibt sich als:

$$F = \frac{1}{z} \cdot n \quad (3)$$

z ist die z-Koordinate des Punktes, der innerhalb des Dreiecks liegt und n ist die Normale dieses Dreiecks.

B. Baryzentrische Koordinaten

Eine weitere Methode ist die Verwendung von baryzentrischen Koordinaten. Diese Methode ist sehr ähnlich zur "Edge-Function", da diese ebenfalls verwendet wird, um herauszufinden, wo sich ein Punkt relativ zu einem Dreieck

befindet. Ein wichtiger Unterschied ist allerdings, dass außerdem die relative Position innerhalb des Dreiecks bestimmt werden kann. Berechnet werden die Koordinaten wie folgt:

$$\begin{aligned} \lambda_b &= \pm \frac{|\Lambda(\Delta(Q, B, C))|}{|\Lambda(\Delta(A, B, C))|} \\ \lambda_c &= \pm \frac{|\Lambda(\Delta(A, Q, C))|}{|\Lambda(\Delta(A, B, C))|} \\ \lambda_a &= 1 - \lambda_1 - \lambda_2 \end{aligned} \quad (4)$$

A, B, C sind hierbei die Eckpunkte des Dreiecks und P die in das Koordinatensystem des Dreiecks transformierte Roboter Position. Λ repräsentiert eine Funktion für den Flächeninhalt eines Dreiecks. λ_1 , λ_2 und λ_3 sind die baryzentrischen Koordinaten des Punktes P, welche aussagen, wie weit der Punkt von den jeweiligen Eckpunkten entfernt ist. Wenn alle drei Koordinaten positiv sind, liegt der Punkt innerhalb des Dreiecks. Nun lässt sich ein Schwellwert festlegen, ab dem eine Interpolation zwischen den Kanten/Eckpunkten des Dreiecks stattfindet. Dazu ist es notwendig, für jede Kante zu wissen, welche Dreiecke sie enthält und welche Normalen diese Dreiecke haben. Das Gleiche gilt für die Eckpunkte. Anschließend kann die Kraft wie folgt berechnet werden:

$$\begin{aligned} F_{BC} &= \begin{cases} \frac{\lambda_a}{S} \cdot N_{\Delta} + (1 - \frac{\lambda_a}{S}) \cdot N_{BC} & 0 < \lambda_a < S \\ 0 & \text{sonst.} \end{cases} \\ F_{AC} &= \begin{cases} \frac{\lambda_b}{S} \cdot N_{\Delta} + (1 - \frac{\lambda_b}{S}) \cdot N_{AC} & 0 < \lambda_b < S \\ 0 & \text{sonst.} \end{cases} \\ F_{AB} &= \begin{cases} \frac{\lambda_c}{S} \cdot N_{\Delta} + (1 - \frac{\lambda_c}{S}) \cdot N_{AB} & 0 < \lambda_c < S \\ 0 & \text{sonst.} \end{cases} \end{aligned} \quad (5)$$

N_{Δ} ist dabei die Normale des betrachteten Dreiecks und N_{XY} sind dabei die Normalen der Dreiecke, die die jeweilige Kante XY enthalten. S ist ein Benutzerdefinierter Schwellwert, ab dem die Interpolation stattfindet. Die resultierende Kraft F_{XY} ist die Kraft, die auf den Benutzer wirkt, wenn er sich auf der Kante zwischen den Punkten X und Y bewegt.

Für die Eckpunkte gilt ein ähnlicher Ansatz. Entsprechend wird die Kraft wie folgt berechnet:

$$\begin{aligned} F_A &= \begin{cases} (1 - \frac{\lambda_a}{S}) \cdot N_{\Delta} + \frac{\lambda_a}{S} \cdot N_A & S < \lambda_a < 1 \\ 0 & \text{sonst.} \end{cases} \\ F_B &= \begin{cases} (1 - \frac{\lambda_b}{S}) \cdot N_{\Delta} + \frac{\lambda_b}{S} \cdot N_B & S < \lambda_b < 1 \\ 0 & \text{sonst.} \end{cases} \\ F_C &= \begin{cases} (1 - \frac{\lambda_c}{S}) \cdot N_{\Delta} + \frac{\lambda_c}{S} \cdot N_C & S < \lambda_c < 1 \\ 0 & \text{sonst.} \end{cases} \end{aligned} \quad (6)$$

C. Point-Triangle-Distance-Funktion

Die tatsächlich angewandte Methode zur des Kraftvektors basiert auf den Ergebnissen von Eberly. In der Arbeit wird eine Methode zur Berechnung der Punkt-Dreieck-Distanz präsentiert. Die Methode liefert dabei zwei wesentliche Werte zurück:

- 1) 'dist': Dieser Parameter repräsentiert den minimalen Abstand zwischen einem vorgegebenen Punkt P und dem Dreieck. Der Wert entspricht der kürzesten euklidischen Distanz, die von P bis zur nächsten Position auf dem Dreieck (seien es Kanten, Eckpunkte oder eine Flächenposition innerhalb der Dreiecksgrenzen) gemessen wird.
- 2) 'foot': Dieser Parameter repräsentiert den entsprechenden Punkt auf dem Dreieck, der P am nächsten liegt. Es handelt sich hierbei um den Punkt auf dem Dreieck, der den minimalen Abstand 'dist' zu P aufweist.

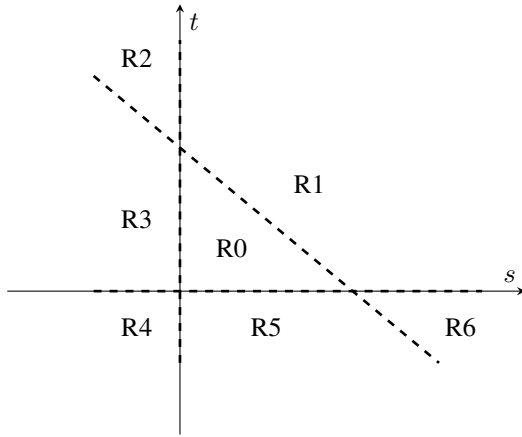


Fig. 2. Die Unterteilung des Dreiecks in Regionen.

Um die beiden Werte $dist$ und $foot$ zu berechnen wird zunächst die Darstellungsform des Dreiecks verändert. Die Eckpunkte eines Dreiecks werden in eine Ebenengleichung $T(s, t) = B + sE_0 + tE_1$ übertragen, wobei $(s, t) \in D = \{(s, t) : s \geq 0, t \geq 0, s + t \leq 1\}$ gelten muss. Die Entfernung $Q(s, t)$ zwischen einem Punkt P und dem Dreieck $T(s, t)$ lässt sich durch den quadrierten euklidischen Abstand als elliptischen Paraboloid darstellen:

$$Q(s, t) = |T(s, t) - P|^2 = as^2 + 2bst + ct^2 + 2ds + 2et + f \quad (7)$$

mit $(s, t) \in D$. Die Koeffizienten a, b, c, d, e, f lassen sich durch die Eckpunkte des Dreiecks und den Punkt P berechnen [4]. Somit kann die Punkt-Dreieck-Distanz durch die Minimierung von $Q(s, t)$ bestimmt werden. Durch die Annahme, dass E_0 und E_1 linear unabhängig sind, kann $Q(s, t)$ nur ein globales Minimum besitzen. Das Minimum von $Q(s, t)$ kann durch das Nullsetzen des Gradienten ermittelt werden. Daraus ergeben sich die Parameter $\tilde{t} = \frac{be - cd}{ac - b^2}$ sowie $\tilde{s} = \frac{bd - ae}{ac - b^2}$. Falls $(\tilde{s}, \tilde{t}) \in D$ befindet sich der sogenannte Fußpunkt 'foot' des Punktes P im Inneren des Dreiecks. Andernfalls ist der Fußpunkt auf den Kanten oder Eckpunkten des Dreiecks

lokalisiert. Um die konkrete Lage des Fußpunktes zu bestimmen wird das Dreieck wie in Bild 2 in unterschiedliche Regionen unterteilt. Der genaue Fußpunkt wird mithilfe der Ebenenkurven von $Q(s, t)$ bestimmt. Eine Ebenenkurve eines elliptischen Paraboloids fällt als einfache Ellipse aus. Diese Ellipsen liegen in der s, t -Ebene und $Q(s, t)$ hat dort eine konstante Höhe. Dies vereinfacht die Berechnungen, ob der Fußpunkt auf einer Kante oder einem Eckpunkt liegt.

Für die Regionen R1, R3 und R5 wird die Ebenenkurve ausgewählt, die die jeweilige Kante des Dreiecks tangential berührt. Das Minimum der Funktion $Q(s, t)$ ist dann in R1 durch (s, t) , in R3 durch $(0, t)$ und in R5 durch $(s, 0)$ gegeben.

Die Regionen R2, R4 und R6 stellen Sonderfälle dar. In diesen Fällen müssen sowohl die Richtung der Dreieckskante als auch die des Gradienten berücksichtigt werden. Das Vorzeichen des Skalarprodukts der beiden Richtungen wird betrachtet, um die spezifische Region des Fußpunkts zu bestimmen. Dabei gilt für R2 $(1, -1) \cdot \nabla Q(0, 1)$, für R4 $(1, 0) \cdot \nabla Q(0, 0)$ und für R6 $(-1, 0) \cdot \nabla Q(1, 0)$.

Auf diese Weise kann der Fußpunkt 'foot' des Punktes P auf dem Dreieck bestimmt werden. Der Abstand 'dist' ergibt sich aus der Länge des Vektors $\overrightarrow{P - foot}$. Um anschließend den resultierenden Kraftvektor zu ermitteln, wurde folgendes Verfahren implementiert:

Für jedes Dreieck der STL Datei:

- 1) Berechnung von $dist$ und $foot$ mithilfe der Point-Triangle-Distance-Funktion.
- 2) Berechnung des Vektors v zwischen $foot$ und der Roboterposition P .
- 3) Wenn das Skalarprodukt von v und der Normalen des Dreiecks positiv ist, wird die resultierende Kraft F wie folgt ergänzt:

$$F += v \cdot influence(dist) \quad (8)$$

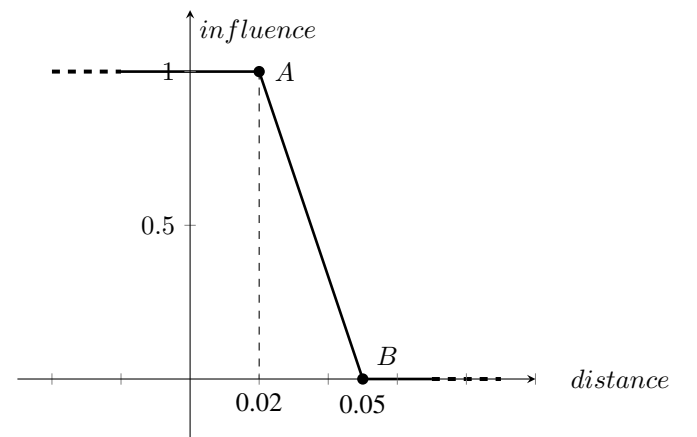


Fig. 3. Die Funktion $influence(dist)$, die den Einfluss der Distanz auf den resultierenden Kraftvektor F bestimmt.

Die Funktion $influence(dist)$ repräsentiert eine linear fallende Funktion innerhalb eines spezifischen Intervalls, das

von 1 auf 0 abfällt (siehe Bild 3). So können die verschiedenen Einflüsse der Kraftvektoren auf den resultierenden Gesamtkraftvektor variiert werden. Ein breiteres Intervall führt dabei zu einer weicheren und runderen Haptik des simulierten Objekts. In unserer Implementierung haben wir ein Intervall von 5cm bis 2cm festgelegt. Vektoren, die von dem Punkt der minimalen Distanz zur Roboterposition ausgehen und eine Länge von weniger als 2 cm aufweisen, werden mit voller Intensität in die Berechnung einbezogen. Vektoren mit einer Länge von 5 cm oder mehr werden aus der Berechnung ausgeschlossen. Für alle dazwischenliegenden Vektorlängen erfolgt eine lineare Interpolation.

Zur Visualisierung der Kräfte für verschiedene Punkte wurde ein Vektorfeld erstellt (siehe 4). Man kann klar erkennen, dass keine Unstetigkeiten entlang der Kanten und Eckpunkte auftreten.

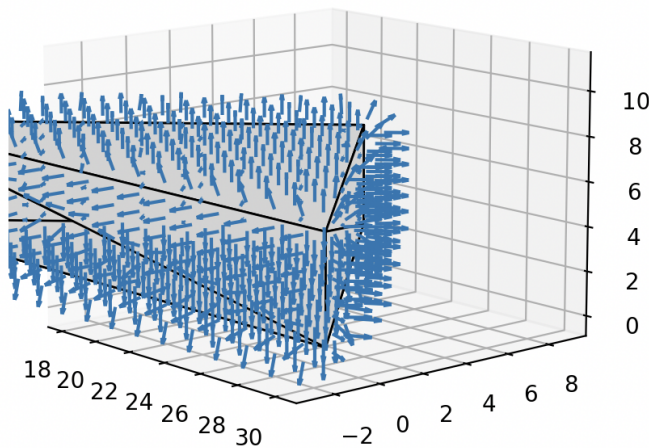


Fig. 4. Vektorfeld Visualisierung.

III. ERGEBNISSE

A. Effekt

Bei der Überprüfung unserer Implementierung konnten wir feststellen, dass der beabsichtigte Effekt der abstoßenden Kraft sowie das gewünschte Gefühl von Plastizität erfolgreich erreicht wurden. Die Kanten und Ecken der in der STL-Datei definierten Struktur waren deutlich wahrnehmbar. Ein kurzes Demonstrationsvideo, welches die Funktionsweise und das Resultat der angewandten Implementierung veranschaulicht, kann unter dem folgenden Link eingesehen werden: <https://youtu.be/lhARjkxuTvw>.

B. Performanz

Hinsichtlich der Performanz wurden ebenfalls zufriedenstellende Resultate erzielt. Bei der Verarbeitung einer STL-Datei, welche 160 Dreiecke beinhaltet, betrug die durchschnittliche Berechnungszeit für den Kraftvektor eines Punktes lediglich 0.714 Millisekunden bei der oben beschriebenen Hardware. Im Kontext einer Echtzeitanwendung stellt dieser Wert eine akzeptable Berechnungslast dar.

IV. DISKUSSION

Die Methode aus II-A zeichnet sich durch Einfachheit und Geschwindigkeit aus. Allerdings bietet diese keine Möglichkeit, zwischen den einzelnen Dreiecken zu interpolieren. Daher entstehen Unstetigkeiten an den Kanten der Dreiecke, was zu unintuitivem und sprunghaftem Verhalten führt, wenn sich der Benutzer entlang der Kante bewegt.

Der Vorteil von II-B gegenüber der II-A ist, dass die Unstetigkeit entlang der Kanten vermieden werden. Die Berechnung der baryzentrischen Koordinaten ist jedoch komplexer und langsamer. Außerdem ist zum Interpolieren der unterschiedlichen Dreieckflächen die Bestimmung der Nachbarschaftsbeziehungen notwendig. So ist der Rechenaufwand pro Roboterposition deutlich höher.

Die Methode II-C schafft es ein lückenloses Vektorfeld zu erzeugen und weist trotzdem eine gute Performanz auf. Mit unserer Implementierung liegt die Laufzeit allerdings in $O(n)$, was zu Skalierungsproblemen führt. Sollten die STL-Dateien eine bestimmte Größe überschreiten, kann die Anforderung an die Echtzeitfähigkeit nicht mehr zuverlässig eingehalten werden. Mögliche Ansätze zur Verbesserung der Performanz wären:

- 1) Das Parallelisieren der Berechnungen durch das Einsetzen der GPU.
- 2) Level of Detail (LOD) Modelle, wie sie in der Computergrafik verwendet werden.
- 3) Nutzung von Octrees.
- 4) Nutzung von leistungstärkerer Hardware.

Durch die Anwendung dieser Methoden könnte die Echtzeitanforderung des Systems auch bei umfangreicheren STL-Dateien erfüllt werden.

V. FAZIT

Zusammenfassend kann gesagt werden, dass es gelungen ist mit II-C eine effiziente Methode zu entwerfen, ein haptisches Feedback zu erzeugen. Durch das Verwenden des STL-Standarts können mit Beschränkungen in der Auflösung des 3D-Modells beliebige Objekte zu stetigen und kontinuierlichen Vektorfeldern konvertiert werden. Somit ist eine einfache und flexible Handhabung gewährleistet. Die Methode zur Erzeugung haptischen Feedbacks eines Roboters durch virtuelle 3D-Modelle kann demnach in unterschiedlichen Anwendungsbereichen Einzug finden.

REFERENCES

- [1] "Was ist eine STL-Datei?" 3D Systems. (), [Online]. Available: <https://de.3dsystems.com/quickparts/learning-center/what-is-stl-file> (visited on 08/13/2023).
- [2] F. Emika, *Franka emika panda technical data*, May 2018. (visited on 08/03/2023).
- [3] J. Pineda, "A Parallel Algorithm for Polygon Rasterization," 1988.
- [4] D. Eberly, "Distance Between Point and Triangle in 3D," Sep. 28, 1999.