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*—!!!NOCH GPT!!! In dieser Studie wird ein Ansatz zur Generierung von haptischem Feedback in Mensch-Roboter-Interaktionen vorgestellt. Mittels maßgeschneiderter Software erzeugen wir eine sensorische Wahrnehmung, 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, wodurch eine physische Interaktion mit dem immateriellen Modell simuliert wird. Die vorgestellte Technologie findet breite Anwendungsbereiche, von medizinischen Simulationen und Exploration in gefährlichen Zonen bis hin zur Erhöhung der Spielerfahrung in virtuellen Umgebungen. In Verbindung mit Virtual-Reality-Ausrüstung eröffnet unsere Methode neue Wege zur Verbesserung der Benutzerimmersion durch die Vermittlung eines realistischeren Gefühls für die Form und Textur virtueller Objekte. Die in dieser Arbeit vorgestellten Prinzipien und Implementierungen können als Grundlage für weiterführende Forschungen und Entwicklungen auf diesem aufstrebenden Gebiet dienen.**

***Index Terms*—component, formatting, style, styling, insert**

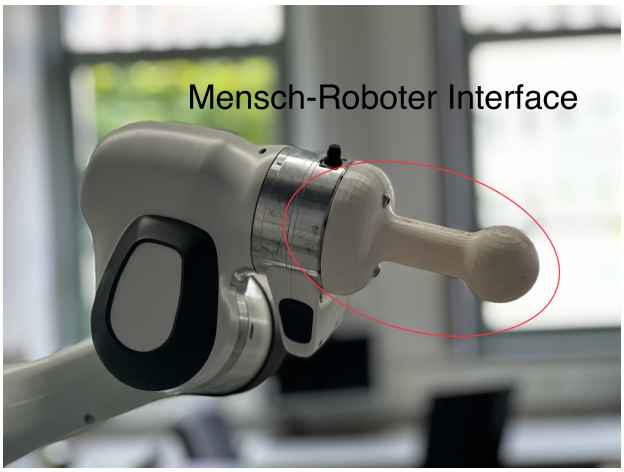

Fig. 1. Mensch-Roboter-Schnittstelle

I. Einleitung

!!! GPT =¿

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 präsentieren wir einen Ansatz, der es dem Benutzer ermöglicht, virtuelle 3D-Modelle zu ”ertasten”, indem er über ein Roboterinterface mit ihnen interagiert. Dies wird durch die Erzeugung einer abweisenden Kraft erreicht, die auf den Grenzen der virtuellen Modelle basiert. Dieses haptische Feedback simuliert das physische Berühren eines realen Objekts, obwohl kein tatsächlicher physischer Kontakt mit dem virtuellen Modell besteht.

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. Darüber hinaus kann die Kombination unserer Technologie mit Virtual-Reality-Brillen zu einem verbesserten Gefühl von Präsenz und Realismus in virtuellen Umgebungen führen.

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

¡= GPT !!!

II. Methods

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 Datei im STL-Standart übergeben werden. Der STL-Standart speichert 3D-Modelle als Dreiecksmesh, das aus Eckpunkten und den Normalen der Dreiecke besteht. Es ist keine besondere CAD Software nötig um 3D-Modelle im STL Format zu erstellen. Außerdem ist das Format weit verbeitet und wird in der Industire zur Speicherung und Übertragung von 3D-Modellen eingesetzt. Das sorgt für hohe Flexibilität und einfache Handhabung. !!!Quelle zu STL Standart!!!

Für die physische Implementierung der Mensch-Roboter-Interaktion wurde der Panda-Roboter von Franka Emika ausgewählt. Der Panda zeichnet sich durch seine 7-achsige Struktur aus, die hohe Präzision und Flexibilität bietet. Eine benutzerfreundliche, 3D-gedruckte Schnittstelle (siehe 1) wurde am Endeffektor des Roboters installiert, um die Führung durch menschliche Benutzer zu erleichtern.

Bei der Entwicklung der Software für dieses System stand die Leistungsfähigkeit im Vordergrund, um eine nahtlose Benutzererfahrung zu gewährleisten. Dazu war es entscheidend, dass die Anwendung in Echtzeit ausgeführt werden kann, wobei eine maximale Zeitverzögerung von weniger als 1 Millisekunde akzeptiert wurde. Die Programmiersprache C++ wurde aufgrund ihrer hohen Performance ausgewählt, um diesem Anspruch gerecht zu werden. Die Hardware-Konfiguration bestand aus !!!HARDWARE!!!. Die Verarbeitung und Nutzung der 3D-Modelle wurden durch die Umwandlung der STL-Daten in Eigen-Matrizen optimiert. Für diesen Prozess wurde durch einen STL-Parser benutzt. Dadurch ist eine einfache und effiziente Handhabung der 3D-Daten innerhalb des Programms möglich.

Während der Recherche des Projekts wurden verschiedene Ansätze zur Erzeugung der abstoßenden Kraft, welche in den folgenden Abschnitten vorgestellt werden.

Als Vorbereitung auf die Verwendung der Ansätze II-A und II-B muss zuerst ein zweidimensionales Problem aus dem dreidimensionalen Problem gemacht 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 Funktion

Die Edge-Funktion ist eine häufig verwendete Funktion im Bereich der Computer-Grafik. Sie wird verwendet, um herauszufinden, wo sich ein Punkt relativ zu einem Dreieck befindet. 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:

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

!!!BILD!!!

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 Kraft kann dann wie folgt berechnet werden:

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

z ist die z-Koordinate des Punktes, der innerhalb des Dreiecks liegt und n ist die Normale dieses Dreiecks. Die großen Vorteile dieser Methode sind die Einfachheit und die Geschwindigkeit. Die Berechnung der Edge-Funktion ist sehr einfach und schnell, bietet aber keine Möglichkeit, zwischen den einzelnen Dreiecken zu interpolieren. Daher enstehen Unstetigkeiten an den Kanten der Dreiecke, was zu unintuitivem und sprunghaftem Verhalten führen kann, wenn der Benutzer sich entlang der Kante bewegt. Aus diesem Grund haben wir die Methode verworfen und uns an einer weiteren Methode versucht.

B. Baryzentrische Koordinaten

Eine weitere Methode, die wir untersucht haben, ist die Verwendung von baryzentrischen Koordinaten. Diese Methode ist sehr ähnlich zur Edge-Funktion, da sie auch 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:

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

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:

$$F_{BC} = \begin{cases} \frac{\lambda_a}{S} \cdot N_\triangle + (1 - \frac{\lambda_a}{S}) \cdot N_{BC} & 0 < \lambda_a < \text{S} \\ 0 & \text{sonst.} \end{cases}$$

$$F_{AC} = \begin{cases} \frac{\lambda_b}{S} \cdot N_\triangle + (1 - \frac{\lambda_a}{S}) \cdot N_{AC} & 0 < \lambda_b < \text{S} \\ 0 & \text{sonst.} \end{cases}$$

$$F_{AB} = \begin{cases} \frac{\lambda_c}{S} \cdot N_\triangle + (1 - \frac{\lambda_a}{S}) \cdot N_{AB} & 0 < \lambda_c < \text{S} \\ 0 & \text{sonst.} \end{cases}$$

$N_\triangle$ 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:

$$F_A = \begin{cases} (1 - \frac{\lambda_a}{S}) \cdot N_\triangle + \frac{\lambda_a}{S} \cdot N_A & S < \lambda_a < 1 \\ 0 & \text{sonst.} \end{cases}$$

$$F_B = \begin{cases} (1 - \frac{\lambda_b}{S}) \cdot N_\triangle + \frac{\lambda_b}{S} \cdot N_B & S < \lambda_a < 1 \\ 0 & \text{sonst.} \end{cases}$$

$$F_C = \begin{cases} (1 - \frac{\lambda_c}{S}) \cdot N_\triangle + \frac{\lambda_c}{S} \cdot N_C & S < \lambda_a < 1 \\ 0 & \text{sonst.} \end{cases}$$

Der Vorteil gegenüber der Edge-Funktion ist, dass so die Unstetigkeit entlang der Kanten vermieden wird, die Berechnung der baryzentrischen Koordinaten ist jedoch etwas komplexer und langsamer als die Berechnung der Edge-Funktion. Daher wurde auch diese Variante letztendlich nicht verwendet.

C. Point-Triangle-Distance-Funktion

Die tatsächlich angewandte Methode zur des Kraftvektors basiert auf den Untersuchungen von Eberly. Diese präsentieren eine Methode zur Berechnung der Punkt-Dreieck-Distanz und liefern 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) 'pp0': 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.

Die Berechnungsmethode beruht auf der Anwendung von Vektoren und Skalarprodukten und integriert speziell die Berücksichtigung verschiedener 'Regionen'. Diese 'Regionen' sind durch die relativen Positionen des Punktes 'p' und des Dreiecks definiert und erlauben eine effiziente Bestimmung des kürzesten Abstandes und des nächstgelegenen Punktes auf dem Dreieck. Die genaue Definition und Charakterisierung dieser 'Regionen' kann aus 2 entnommen werden.

Die Anwendung dieser Methode ermöglicht somit eine präzise und effiziente Bestimmung der Punkt-Dreieck-Distanz im dreidimensionalen Raum.

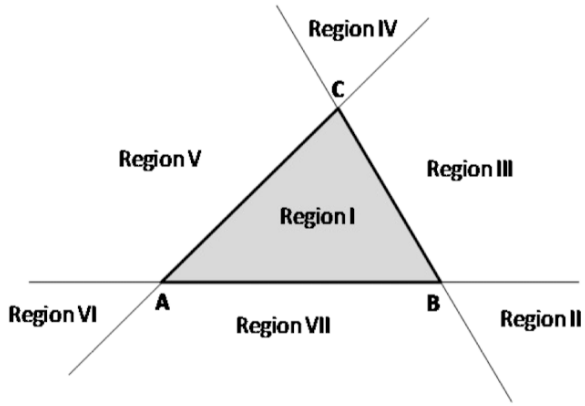

Fig. 2. Die verschiedenen Regionen, die durch die relative Position des Punktes und des Dreiecks definiert sind [2].

Um den resultierenden Kraftvektor zu ermitteln, wurde folgendes Verfahren implementiert:

für jedes Dreieck der STL Datei:

1) Berechnung von $dist$ und $pp0$ mit Hilfe der Point-Triangle-Distance-Funktion.
2) Berechnung des Vektors v zwischen $pp0$ und der Roboterposition p.
3) Wenn das Skalarprodukt von v und der Normalen des Dreiecks positiv ist, wird die resultierende Kraft um
$$F \mathrel{+}= v * f(dist)$$
ergänzt.

Die Funktion $f(dist)$ ist dabei eine linear fallende Funktion, die den Einfluss der Distanz auf die Kraft bestimmt.

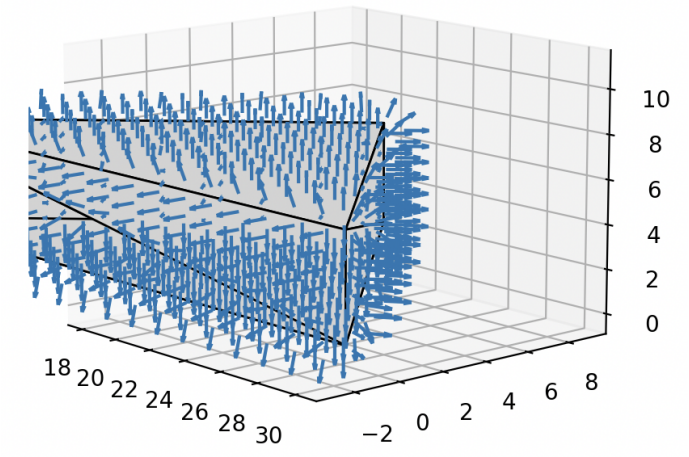

Fig. 3. Vektorfeld, das resultierende Kräfte für verschiedene Punkte visualisiert.

Zur Visualisierung der Kräfte für verschiedene Punkte wurde ein Vektorfeld erstellt (siehe 3). Man kann klar erkennen, dass keine Unstetigkeiten entlang der Kanten und Eckpunkten auftreten.

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/IhARjkxuTvw.

B. Performanz

Hinsichtlich der Performance 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. Discussion

Sollten die STL-Dateien jedoch eine bestimmte Größe überschreiten und es und es wird keine leistungsfähigere Hardware eingesetzt, könnte die Anforderung an die Echtzeitfähigkeit nicht mehr zuverlässig erfüllt werden. Eine potenzielle Lösung hierfür könnte die Anwendung von Level of Detail (LOD) Modellen sein. Diese Modelle, die bei größerer Entfernung des Roboters zum Objekt genutzt werden können, zeichnen sich durch eine reduzierte Anzahl an Dreiecken aus und können demzufolge effizienter verarbeitet werden. Auch die Nutzung von Octrees könnte die Performance weiter verbessern, da durch deren Einsatz die Anzahl der zu verarbeitenden Dreiecke verringert werden kann. Ein Octree ist eine spezielle Art von Baumstruktur, die im Kontext der 3D-Computergrafik verwendet wird und eine effiziente räumliche Aufteilung ermöglicht. Durch die Anwendung dieser Methoden könnte die Leistungsfähigkeit des Systems auch bei umfangreicheren STL-Dateien erhalten bleiben und die Echtzeitanforderungen sicherstellen.

V. Conclusion

References

[1] David Eberly. “Distance Between Point and Triangle in 3D”. In: *28.09.1999* (Mar. 2023). (Visited on 03/08/2023).

[2] Dr. rer. nat. Jan Ehrhardt. *Uebungsblatt4*. June 2023. (Visited on 08/03/2023).