Handgestenerkennung in einem CAVE Automatic Virtual Environment unter Verwendung mehrerer Tiefensensor-Kameras

Thomas Jung, Franz Simon

HTW Berlin,
Wilhelminenhofstraße 75A , D-12459 Berlin
eMail: t.jung@htw-berlin.de
URL: http://www.htw-berlin.de

Zusammenfassung Das Cave Automatic Virtual Environment (CAVE) der HTW Berlin wird im Unterschied zu anderen CAVE-Systemen durch ein Natural User Interface (NUI) gesteuert. Mehrere Tiefensensor-Kameras der Firma Microsoft („Kinect“) ermöglichen die Erkennung eines (fusionierten) Skeletts. Im Rahmen dieser Arbeit wird untersucht, wie einfache Handgesten auf Basis der relativ niedrig aufgelösten Tiefenbilder detektiert werden können. Es wird gezeigt, dass die Handgestenerkennung geeignet ist, Interaktionsszenarien zu realisieren.

1 Einleitung

CAVE-Systeme [1] werden schon seit den 1990er Jahren zur Erzeugung immersiver Virtueller Umgebungen verwendet, die Interaktion erfolgte dabei zunächst über magnetische, später über optische Tracking-Systeme, bei denen Benutzer in der Regel jedoch mit aktiven oder passiven Markern instrumentiert werden. Natural User Interfaces vermeiden weitestgehend sichtbare Bedienelemente, um die Bedienung dadurch natürlicher zu gestalten. Tiefensensor-Kameras, die die Entfernungen zu Bildpunkten bestimmen, ermöglichen dabei die markerlose Erkennung von Gesten. Durch die Entwicklung kostengünstiger Geräte (Microsoft Kinect, Asus Wavi Xtion) gewinnt die Entwicklung von NUIs an Bedeutung.

Die CAVE der HTW besitzt seit 2011 ein NUI basierend auf zunächst einer Kinect-Kamera. Dieses NUI war bereits geeignet, um Head Tracking und einfache Navigationsinterfaces zu realisieren [2]. Ein großer Schwachpunkt war dabei jedoch der stark eingeschränkte Interaktionsraum innerhalb der CAVE. Seit 2012 werden zwei Kinect-Kameras verwendet, deren Daten fusioniert werden, so dass der Interaktionsraum nun nahezu verdoppelt werden konnte [3]. Dabei wurde bereits eine sehr einfache Handgestenerkennung beschrieben, die basierend auf einer einzelnen Kamera zwischen zwei Handgesten (geschlossen, offen) unterscheiden konnte. Sofern die Hand hinreichend gut von der Kamera erfasst werden konnte, lieferte der Erkennungsalgorithmus bereits zufriedenstellende Ergebnisse [3]. Im Rahmen dieser Arbeit wird nun untersucht, wie eine bessere Erkennungsgenauigkeit durch die Verwendung einer zweiten Kamera und leistungsstärkerer Klassifizierungs-algorithmen erreicht werden kann.

2 Handgestenerkennung mit Tiefenbildkameras

Frati und Prattichizzo verwenden die konvexe Hülle der Handkontur zur Erkennung von Gesten. Pixel, deren Tiefenwerte nicht zu weit von der Handposition des Kinect-Skeletts entfernt sind, werden als zur Hand gehörend klassifiziert. Die Fingerspitzen bilden dabei die Eckpunkte der konvexen Hülle. Überschreiten Distanzen von Konturpunkten zur konvexen Hülle einen Schwellwert, gilt ein Finger bzw. die Hand als ausgestreckt [4]. Trindade, Lobo und Barreto verwenden zur Klassifizierung der Handpixel noch zusätzlich einen Farbfilter, mit Hilfe dessen sie zunächst hautfarbige Pixel aus dem Farbbild der Kinect segmentieren. Über einen zusätzlichen Lagesensor werden die zugehörigen Tiefenpixel in der Nähe der Handposition normalisiert. Nach Anwendung eines Voxelfilters werden die verbliebenen Tiefenpixel voxelweise mit Vorlagen verglichen, wobei die Geste mit der größten Übereinstimmung gewählt wird [5].
Tang klassifiziert Handpixel ebenfalls über Farb- und Tiefenbild der Kinect. Er geht davon aus, dass die Handfläche der Kamera zugewandt ist. Ein 64x64 Pixel großes Fenster wird dann in 64 jeweils 8x8 große Teilfenster geteilt, für die jeweils vier Bildmerkmale (SURF-Deskriptoren) berechnet werden, wodurch insgesamt 256 Merkmale generiert werden. Die Handgestenerkennung erfolgt dann durch das Trainieren einer Support Vector Machine anhand dieser Merkmalsvektoren [6].

3 Verwendung mehrerer Tiefenkameras

Die Farbbilder der Kinect können nicht verwendet werden, da die Beleuchtungssituation in der CAVE nicht kontrolliert werden kann, deshalb stehen für die Erkennung der Handgesten nur die Tiefenbilder der beiden Kinect-Kameras zur Verfügung.

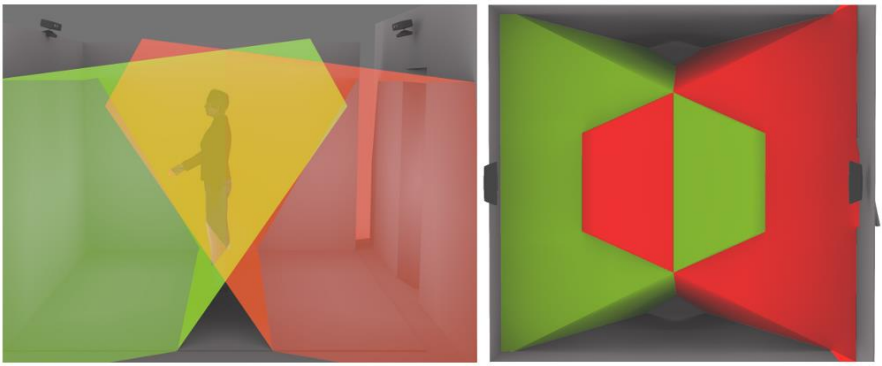

Abb. 1: Darstellung des von den Kinect-Sensoren erfassten Raumes der CAVE von der Seite (links) und von oben (rechts), mit den Sichtbereichen der vorderen Kinect-Kamera (rot) und der hinteren Kamera (grün)

Zwei Kinect-Kameras werden wie in Abbildung 1 zu sehen angeordnet, um einen möglichst großen Interaktionsbereich der CAVE abzudecken. Grundsätzlich könnten

Punktwolken aus beiden Tiefenbildern gewonnen und fusioniert werden. Leider erlaubt es das Kinect-SDK nicht, die beiden Kameras hinreichend genau zu synchronisieren. Beide Kameras erfassen den Raum zwar mit 30 Bildern pro Sekunde, der zeitliche Versatz ist derzeit jedoch nicht beeinflussbar und variiert bis hin zu 33ms. Bei schnellen Bewegungen des Benutzers kann also keine konsistente Punktwolke aus beiden Bildern generiert werden, deshalb wird jeweils nur ein Tiefenbild zur Handgestenerkennung verwendet.

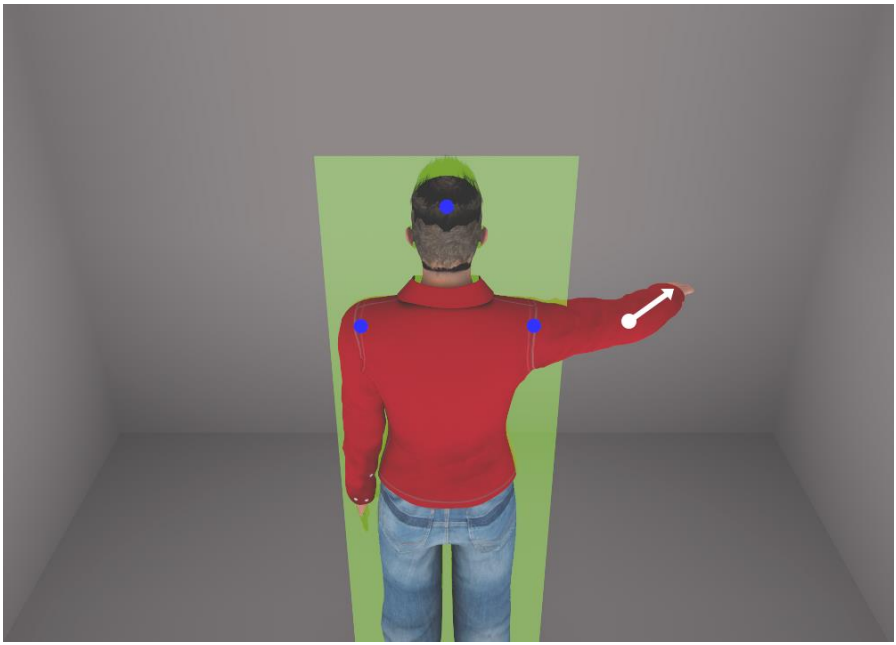

Abb. 2: Gelenkpunkte des Unterarms (weiß), potentielle Verdeckung durch den eigenen Körper (grün).

Es wird jeweils das Tiefenbild der Kinect-Kamera verwendet, die die Hand des Benutzers am besten sieht. Dazu wird zunächst das Sichtvolumen beider Kameras abzüglich der jeweiligen Verdeckung durch den Benutzer bestimmt, um zu entscheiden, ob die Hand im jeweiligen Tiefenbild sichtbar ist (siehe Abbildung 2). Ist die Hand in beiden Tiefenbildern sichtbar, wird das Tiefenbild gewählt, in dem die Hand eher seitlich abgebildet ist (Größter Winkel zwischen Unterarm und Sichtstrahl).

4 Detektionsalgorithmus

Da die Handgelenkposition durch den Algorithmus des Kinect-SDKs in manchen Situationen nicht korrekt erkannt wird (siehe Abbildung 3 rechts), wird ein eigenes Verfahren verwendet, um die Handgelenkposition aus dem Tiefenbild zu bestimmen.

Dabei wird zunächst ein gültiges Saatpixel im Bereich zwischen Ellenbogen und Hand bestimmt, mit Hilfe dessen dann mit dem Flood-Fill-Algorithmus weitere Tiefenpixel des Unterarms bestimmt werden, die einen 3D-Abstand von nicht mehr als 50 cm zum Saatpixel besitzen und eher in Richtung der Hand liegen (siehe Abbildung 3 links).

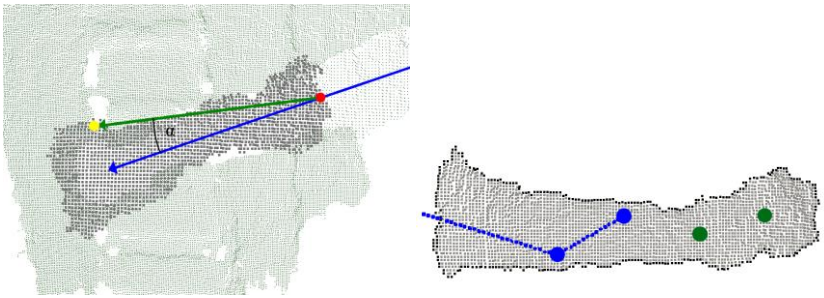

Abb. 3: links: Segmentierung eines Unterarmpixels (gelb) im Tiefenbild, Verbindungsvektor Ellenbogen Hand (blau), der Winkel α muss kleiner 90 Grad sein; rechts: fehlerhafte Gelenkpunkte des Skeletts (blau), korrigierte Punkte (grün) innerhalb der detektierten Kontur

Liegt eine eher seitliche Darstellung der Hand vor, wird das Handgelenk auf der Höhe des geringsten Abstands ähnlich wie bei Doliotis und Kollegen [6] zwischen zwei gegenüberliegenden Pixeln der geglätteten Unterarmkontur bestimmt (siehe Abbildung 4). Betrachtet werden dann anschließend nur Pixel zwischen Handgelenk und Handspitze. Ist die Hand eher zur Kamera hin ausgerichtet (siehe Abbildung 5 untere Reihe), werden Pixel im Abstand von maximal 20cm zur Handspitze (Tiefenpixel mit größtem Abstand zum Ellenbogen) betrachtet.
Die segmentierten Tiefenpixel der Hand werden dann in beiden Fällen horizontal ausgerichtet. Die Ausrichtung entspricht einer Rotation der projizierten Achse des Unterarms auf die x-Achse (siehe Abbildung 5). Um die segmentierten Tiefenpixel der Hand wird ein quadratisches Fenster gelegt und in 8x8 Teilbereiche unterteilt. Zuletzt wird mit dem Verfahren von Tang [7] klassifiziert, wobei jedoch nur das Tiefenbild verwendet wird. Dabei werden zum Training der Support Vector Machine Trainingsdaten verwendet, in denen einerseits eine geöffnete andererseits eine geschlossene Hand durch die CAVE geführt und aufgezeichnet wurde.

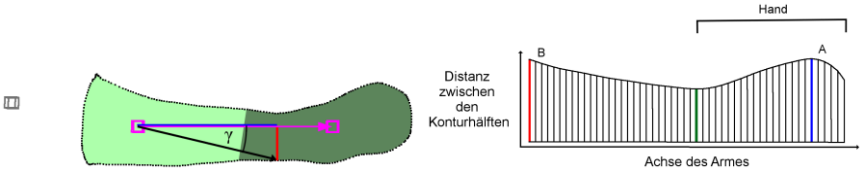

Abb. 4: links: Hauptachse (blau) zwischen den Mittelpunkten der beiden Tiefenpixelmengen für Hand und Unterarm, rechts: Abstände zwischen gegenüberliegenden Konturpunkten bezogen auf die Hauptachse, Handgelenk (grün)

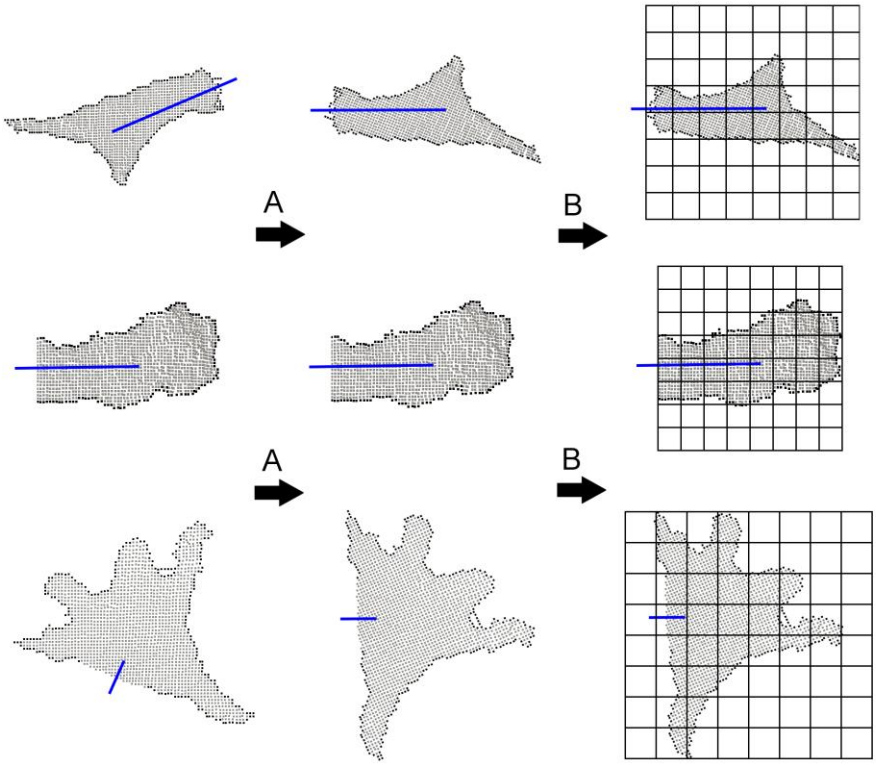

Abb. 5: linke Spalte: segmentierte Handtiefenpixel und Hauptachse (blau), mittlere Spalte: ausgerichtete Tiefenbilder, rechte Spalte: skalierungsinvariante Fenster für die Segmentierung

5 Training der Support Vector Machine

Insgesamt wurden 12842 Datensätze aufgenommen. Ungefähr 60% der Datensätze (n=7705) wurden als Trainingsdaten verwendet, die restlichen 5137 Datensätze dienten zum Test. Mit Hilfe fünffacher Kreuzvalidierung wurden in der Trainingsphase optimale Parameterwerte log2(C) = 3 und log2(γ) = -3 für den verwendeten RBF-Kernel bestimmt. Die Genauigkeit des trainierten Modells wurde dann mit Hilfe der Testdatensätze bestimmt. Sie betrug 98,9% und war damit leicht höher als die der Trainingsdaten von 98,7%. Somit wurde die Erkennungsgenauigkeit gegenüber der einfachen Implementierung aus [3], die bei ca. 84% lag, deutlich verbessert.

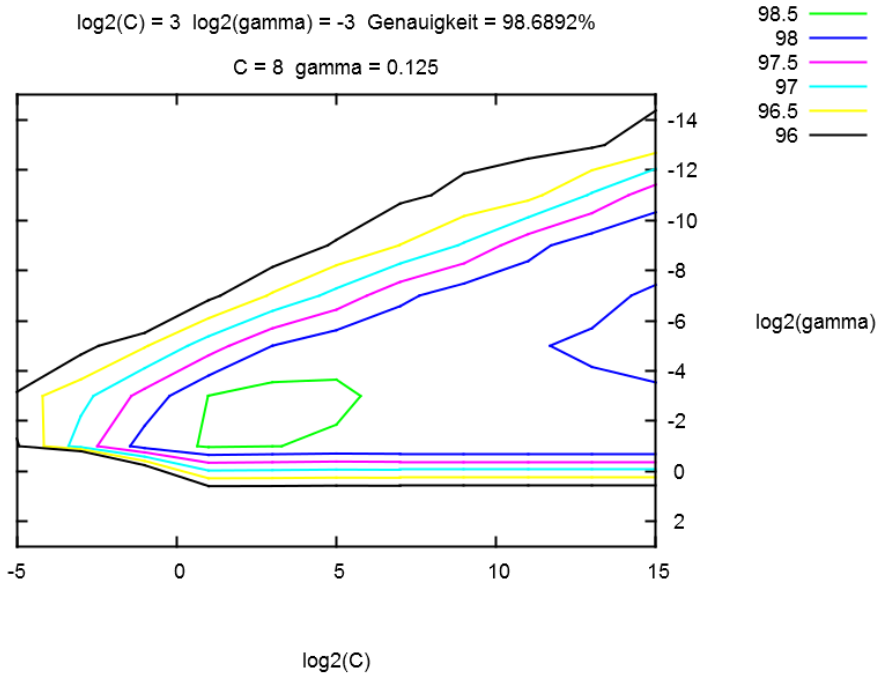

Abb. 6: Visualisierung der Ergebnisse einer Rastersuche nach optimalen Kernel-parameterwerten, mit dem optimalen Bereich (grün), in welchem die Genauigkeit mehr als 98,5% beträgt, sowie der optimalen Kombination der Parameterwerte.

6 Benutzertest

In einer Testanwendung sollten Benutzer Würfel greifen und verschieben. Durch die Verwendung beider Hände konnten die Benutzer die Würfel außerdem skalieren. Es konnte gezeigt werden, dass es nun durch das Öffnen und Schließen der Hand ohne weitere Eingabegeräte möglich ist, die Operationen auszuführen. Schwierigkeiten ergaben sich dabei lediglich durch das relativ ungenaue Tracking der Kinect bei der Selektion und Positionierung der Objekte.

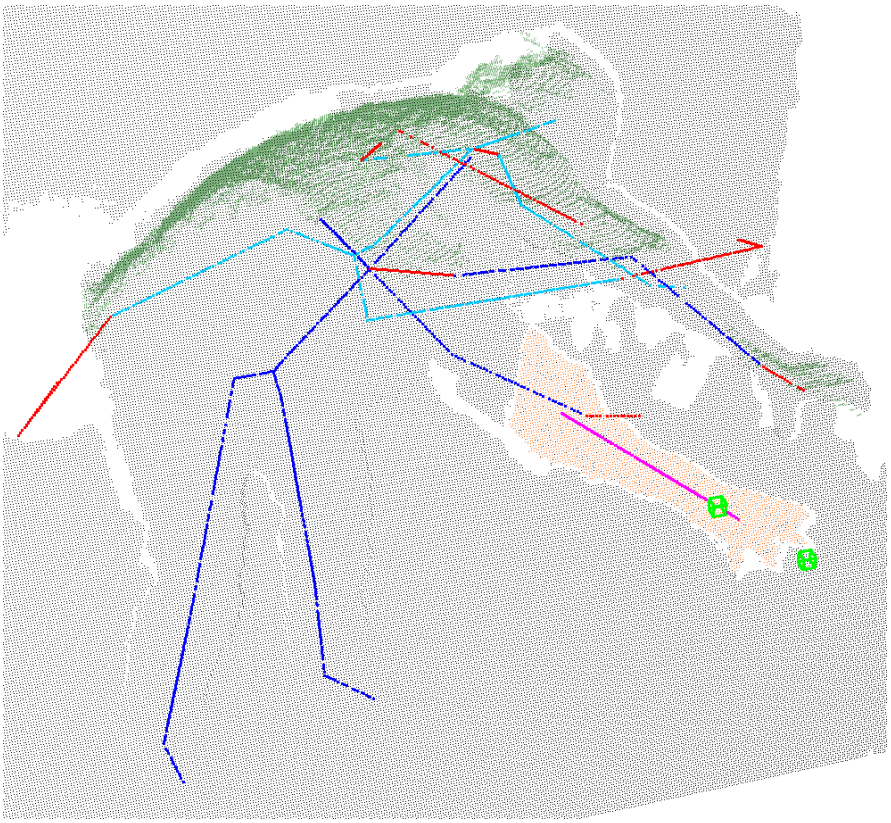

Abb. 7: Darstellung der Punktwolke beider Kinect-Kameras mit den Skelettdaten der hinteren (türkis) und der vorderen Kinect-Kamera (dunkel blau), sowie Knochen (rot) zu Gelenkpositionen die auf geschlossenen Informationen beruhen, den Tiefendaten der vorderen (grau) und der hinteren Kinect-Kamera (dunkel grün), sowie der tatsächlichen Position von Handspitze und Handgelenk (grün)

In Abbildung 7 ist eine relativ ungünstige Situation zu erkennen, in der die Skelettdaten der vorderen Kinect verwendet werden, da die Daten der hinteren Kinect nicht als gemessen gekennzeichnet wurden. Da sich der Benutzer bückt, sind die Gelenkdaten der vorderen Kinect darüber hinaus sehr ungenau. In der Abbildung ist zu erkennen, dass selbst in dieser schwierigen Situation durch den hier beschriebenen Algorithmus die Handposition richtig ermittelt wird und damit auch die korrekte Handgeste erkannt werden kann.

Literatur

[1] Cruz-Neira C., Sandin D. J. und DeFanti T. A., “Surround-Screen Projection-Based Virtual Reality: The Design and Implementation of the CAVE”, Computer Graphics, SIGGRAPH Annual Conference Proceedings, 1993.
[2] Jung T., Krohn S., Schmidt P., “Ein Natural User Interface zur Interaktion in einem CAVE Automatic Virtual Environment basierend auf optischem Tracking”, In Proc Workshop 3D-NordOst 2011, Berlin, Germany, December 2011, pp 93-102
[3] Jung T., Simon F., Interaktion in einem CAVE Automatic Virtual Environment unter Verwendung mehrerer Tiefensensor-Kameras, In Proc Workshop 3D-NordOst 2012, Berlin, Germany, December 2012, pp 199-208
[4] Frati V., Prattichizzo D., “Using Kinect for hand tracking and rendering in wearable haptics” Proc. IEEE World Haptics Conference (WHC) 2011 Istanbul, Turkey, 2011, pp. 317-321.
[5] Trindade P., Lobo J., Barreto J. P.. “Hand gesture recognition using color and depth images enhanced with hand angular pose data” Proc. IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI) 2012 Hamburg, Germany, 2012, pp. 71-76.
[6] Doliotis P., Athitsos V., Kosmopoulos D. und Perantonis S., "Hand Shape and 3D Pose Estimation using Depth Data from a Single Cluttered Frame". In: International Symposium on Visual Computing (ISVC) Rethymnon, Crete, Greece, 2012
[7] Tang M., “Recognizing Hand Gestures with Microsoft’s Kinect”,. Department of Electrical Engineering Stanford University, 2011, http://www.stanford.edu/class/ee368/Project_11/Reports/Tang_Hand_Gesture_Recognition.pdf., Abrufdatum Mai 2013