>> von Matthias Wölfel > *Kinetic Space ist ein Open Source Tool, das es jedermann ermöglicht, mit Hilfe einer Tiefenkamera wie z.B. der Kinect von Microsoft, individuelle Gesten zu trainieren, so dass sie vom Computer erkannt werden können. Hierbei wird auf eine Skelettstruktur des Menschen zugegriffen, die aus dem Tiefenbild extrahiert wird. Die Analyse dieses Skelettes erlaubt nun nicht nur einfache Gesten wie Schieben, Auswählen durch Deuten oder Winken zu erkennen, wie bereits in zahlreicher Software implementiert, sondern auch kompliziertere Bewegungsabläufe, wie sie zum Beispiel in Tanzperformances oder in der Gebärdensprache vorkommen. Das Softwaretool, welches unter Windows, Mac OS X und Linux lauffähig ist, wurde bereits weltweit von Entwicklern, Medienkünstlern, Tänzern und Musikern zur Steuerung von Drittanbietersoftware und Hardware verwendet.*

Einleitung

Kurze Zeit nachdem Microsoft Anfang November 2010 den Kinect-Sensor[1] als Steuerung für die Spielekonsole Xbox 360 auf den Markt gebracht hatte, entwickelte sich rasch eine rasant wachsende Community, die den Sensor, nicht wie von Microsoft vorgesehen an der Xbox, sondern am Computer verwendet, um die zahlreichen neuen Möglichkeiten der kostengünstigen Tiefenkamera für ihre Zwecke zu nutzen. Neben Hackern und Medienkünstlern bedienen sich auch immer mehr Universitäten und Forschungseinrichtungen dieser Technik. So gibt es inzwischen nicht nur viele künstlerische Arbeiten, sondern auch ein breites Spektrum an Anwendungen, von denen kurz ein paar vorgestellt werden.

Recht früh entstand eine Arbeit „be your own souvenir" von blablabLAB (www.blablablab.org), bei der es möglich war, sich auf einem öffentlichen Platz von dem Kinect Sensor scannen zu lassen und sich selbst als Figur aus einem 3D Druck mit nach Hause zu nehmen. Ein weiteres Projekt, das sich direkt mit der Auswertung des Tiefenbildes befasst, diesmal von Microsoft Research in England, ist KinectFusion (http://research.microsoft.com/apps/video/dl.aspx?id=152815). Es ermöglicht das schnelle Erfassen eines 3D-Modells einschließlich Textur aus einer realen Umgebung. Beispiele, die auf der Auswertung der Skelettstruktur beruhen, finden sich insbesondere in der Musik- und Performanceszene, wo die Koordinaten der Hand oder der Füße direkt dazu verwendet wurden, Töne zu manipulieren oder Lichteffekte zu steuern. So lässt sich sogar eine Luftgitarre spielen, die „echte Töne" erzeugt (www.youtube.com/watch?v=8DmOux4IdAE). Ein weiteres Beispiel der Verwendung der Skelettstruktur ist die Steuerung von virtuellen Figuren. Hier werden die einzelnen Körperteile und Körperbewegung auf einen 3D Charakter übertragen (http://www.ni-mate.com). Um die Abbildung des virtuellen Charakters zu vervollständigen, lassen sich durch den Abgleich von Gesichtsmasken mit dem Tiefenbild eines Gesichtes auch die Gesichtsmimiken auf ein 3D-Gesicht abbilden (www.faceshift.com).

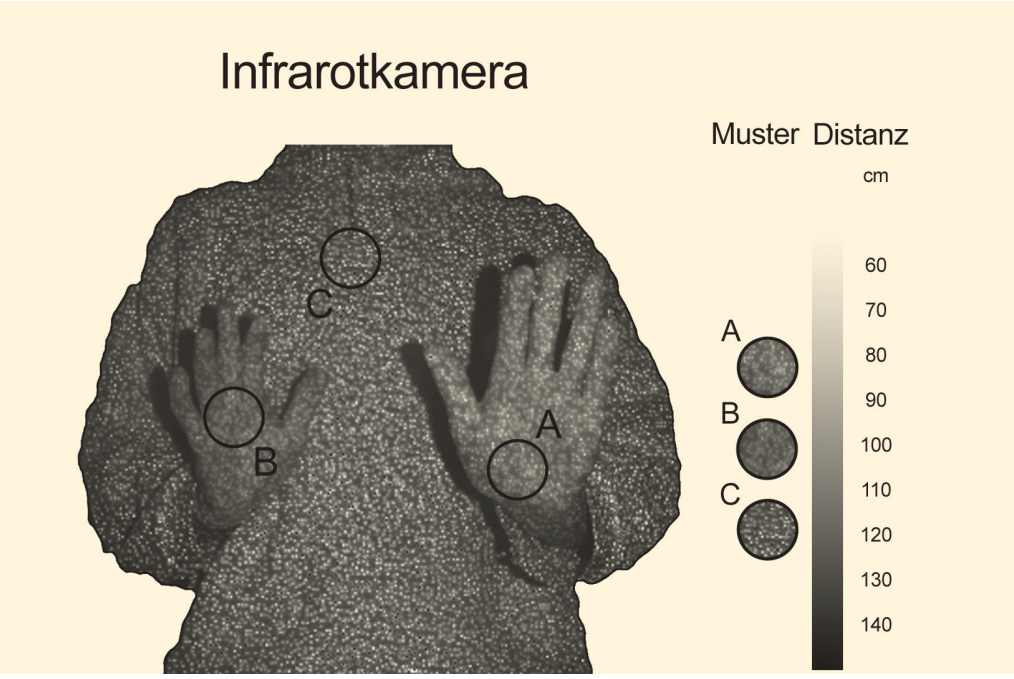
Eine gute Übersicht über weitere Kinect Hacks findet sich auf den Websiten www.kinecthacks.com und http://kinect.dashhacks.com.

Was jedoch fehlte, war die Möglichkeit, dass ein Computer Gesten lernen und erkennen kann. Kinetic Space entstand somit aus der Idee heraus, ein intuitives Tool – für alle Plattformen und als Open Source – zur Verfügung zu stellen, das es jedem Anwender ermöglicht, den Computer eigene Bewegungsabläufe durch simples Vormachen zu lehren, anstatt sie mittels Programmcode zu beschreiben. Mit dem hier vorgestellten Tool ist es zum ersten Mal möglich, dem Computer beliebige Gesten – ohne Programmierkenntnisse – beizubringen und diese für die eigene Anwendung zu nutzen.

Funktionsweise der Tiefenkamera

Im Gegensatz zu einer klassischen Kamera, die Schwarzweiß- oder Farbbilder aufnimmt, besteht der Kinect Sensor neben einer klassischen Kamera und einem Mikrofonarray aus einem Tiefensensor, wodurch viele neue Analysen – zum Beispiel die dreidimensionale Gestenerkennung – erst ermöglicht werden. Im Folgenden wird kurz auf die Funktionsweise des Tiefensensors, der im Kinect Sensor verwendet wird, eingegangen. Es sei angemerkt, dass es hier eine Vielzahl andere Verfahren gibt, mit systemspezifischen Vor- und Nachteilen, zum Beispiel mit Hilfe von Stereokameras oder Time-of-Flight Sensoren, die bei unserer Beschreibung (aus Platzgründen) außen vor gelassen werden.

Mit einem Verfahren, das Light Coding genannt wird, kann in dem Kinect die Tiefeninformation berechnet werden. Durch einen Laser, der im Infrarotlichtbereich arbeitet und somit für das menschliche Auge unsichtbar ist, werden Muster auf die Szene projiziert (lin-



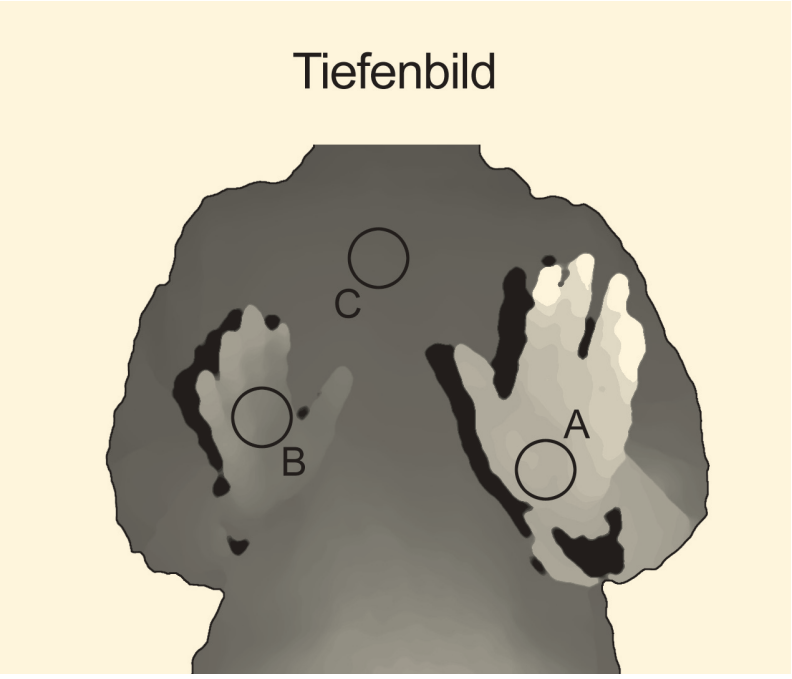
3D-GESTENERKENNUNG FÜR DICH UND MICH

kes Bild in *Abbildung 1*). Diese Muster verändern ihre Abbildung in der Kinect Infrarotkamera in Abhängigkeit von der Distanz zwischen Sensor und Objekt. Durch den Vergleich der Reflexion des lokalen Musters eines Objekts mit vorher gelernten Mustern (durch Projektion auf Flächen mit bekannter Distanz) kann dessen Distanz geschätzt werden. Drei Muster A, B und C sind in *Abbildung 1* mit ihrer dazugehörigen Tiefe dargestellt. Das Tiefenbild (rechtes Bild in *Abbildung 1*) ergibt sich durch einen Vergleich aller lokalen Muster im Infrarotbild und der entsprechenden Zuweisung. Zu den Bildern sei noch angemerkt, dass sich die Schatten durch den Versatz zwischen dem Laser und dem Sensor ergeben.

Verwendung des Kinect Sensors am Computer

Um den Kinect Sensor am Computer zu verwenden, gibt es zwei Treiber/Toolkits, die von unterschiedlichen Organisationen entwickelt werden. Dies ist zum einen eine Organisation namens OpenNI, der unter anderem PrimeSense[2], Willow Garage und Asus angehören, und zum anderen Microsoft mit dem Kinect Software Development Kit.

Abbildung 1: Funktionsweise von Light Coding.



OpenNI/NITE

OpenNI/NITE bietet eine Vielzahl von Funktionen, um auf die Sensordaten direkt zuzugreifen.

- *Das Tiefenbild* stellt eine Tiefenkarte der Szene,
- *das Farbbild* stellt ein „klassisches" RGB Bild,
- *das Infrarotbild* stellt die Rohdaten der Infrarotkamera und
- *Mikrofone* stellen eine vierkanalige Audioaufnahme zur Verfügung.

Es gibt auch eine Reihe von Funktionen, um Körperteile bzw. einfache Gesten im Raum zu erkennen.

- *Die Gestenerkennung* meldet, wenn bestimmte Gesten erkannt werden (hier handelt es sich um eine Reihe sehr einfacher, vordefinierter Gesten wie z.B. Zeigen oder Winken),
- *die Szenenanalyse* segmentiert den Vorder- von dem Hintergrund, identifiziert Personen und erkennt den Fußboden,
- *die Handerkennung* meldet, wenn Hände gefunden werden und kann diese verfolgen und
- *die Skelettextraktion* erzeugt eine vollständige oder teilweise Beschreibung des menschlichen Körperskelettes als 3D-Daten (auf diesen Daten basiert die in diesem Artikel beschriebene Gestenanalyse).

OpenNI und NITE sind seit Dezember 2010 als Open Source Treiber erhältlich (www.openni.org) und stehen für Windows, Mac OS X und Linux zur Verfügung.

Kinect Software Development Kit

Schon sehr kurze Zeit nach dem Verkaufsstart des Kinect Sensors bemerkte Microsoft, dass sich immer mehr Menschen – die nicht unbedingt Spielefans sind – mit dem Sensor beschäftigten und ihn sozusagen zweckentfremdeten, um ihn im wissenschaftlichen und künstlerischen Umfeld zu nutzen. Erst betrachtete Microsoft das sogenannte „Kinect Hacking" mit Argwohn, ohne jedoch gerichtlich gegen die Hacker vorzugehen. Dann aber erkannte auch Microsoft das große Potential hinter Kinect, welches bis dato unterschätzt wurde und entschloss sich dazu, selbst eine kostenfreie Entwicklungsumgebung anzubieten. Im Juni 2011 veröffentlichte Microsoft die erste Beta-Version, die für jedermann zugänglich war, und im Februar 2012 wurde die erste Version des Kinect Software Development Kit zum Download angeboten (natürlich nur für Windows). Nahezu zeitgleich mit dem Treiber (http://www.kinectforwindows.org) brachte Microsoft den Bewegungssensor auch in einer Version für Computer heraus. Die Besonderheit ist hier ein sogenannter Near-Mode, der bereits Objekte ab einem Abstand von 40 Zentimetern erkennen kann, wo bisher mindestens 80 Zentimeter nötig waren.

Das Kinect Software Development Kit bietet nahezu die gleichen Möglichkeiten wie OpenNI/NITE; im direkten Vergleich – beim aktuellen Softwarestand – besseren Support bei der Aufnahme von Video- und Audio-Streams sowie der Audioverarbeitung (Sprecherlokalisierung, Akustisches Beamformen und Automatische Spracherkennung). >

[1] Obwohl es eine ganze Reihe von alternativen Tiefensensoren gibt, ist der Kinect Sensor der bei weitem erfolgreichste und wurde bisher 18 Millionen Mal abgesetzt. Leider gibt es bisher keine Zahlen, wie viele dieser Sensoren tatsächlich an eine Xbox angeschlossen sind und wie viele zweckentfremdet wurden.

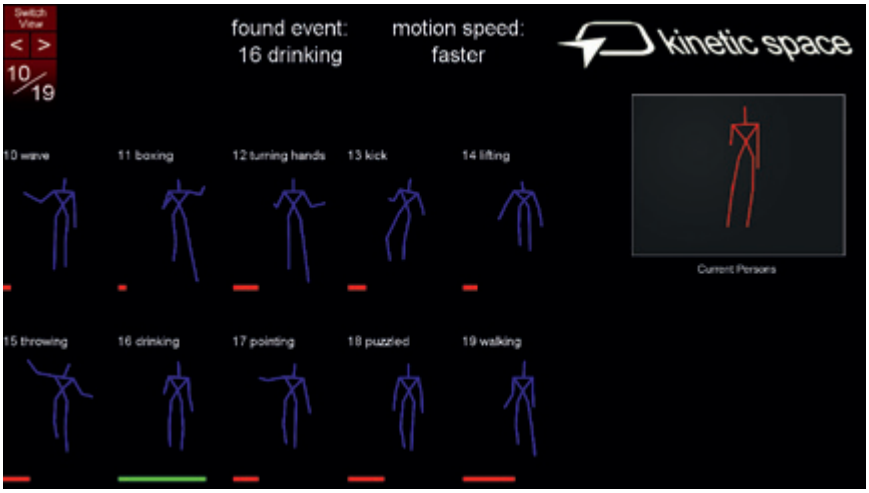
[2] PrimeSense entwickelte das Referenzdesign des in der Kinect verwendeten Tiefensensors.

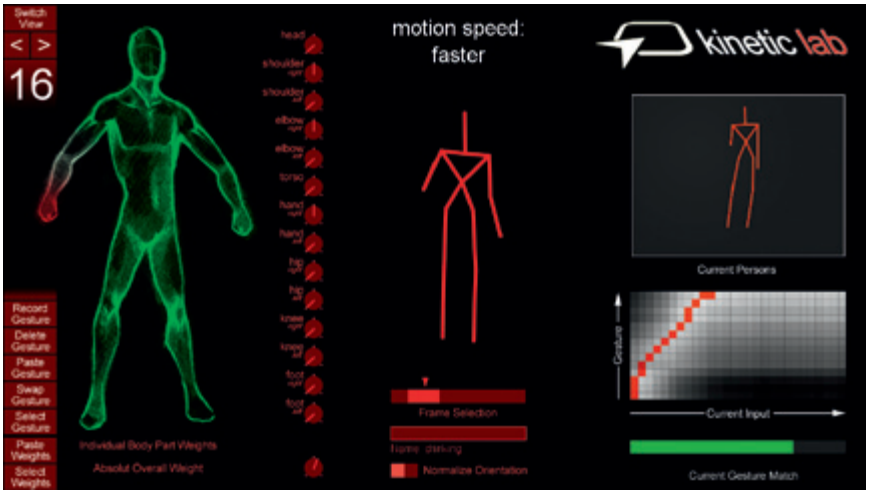
Überblick über Kinetic Space

Bevor wir uns intensiv mit dem technischen Hintergrund der Funktionsweise von Kinetic Space beschäftigen, wollen wir uns mit dem Interface vertraut machen, auf welches in der technischen Diskussion immer wieder Bezug genommen wird. Die Abbildungen 2 und 3 zeigen Screenshots der aktuellen Version von Kinetic Space in den zwei möglichen Ansichten „Space“ und „Lab“. Während die „Space“-Ansicht einen Überblick über je zehn Gesten gibt, erlaubt die „Lab“-Ansicht die genaue Analyse und das Training einer Geste.

Die „Space“-Ansicht (*Abbildung 2*) unterteilt sich in drei Bereiche:

- Tiefenbild (rechts oben in Abb. 2)
  Hier wird das von dem Tiefensensor aufgenommene Bild (helle Bereiche entsprechen näheren Objekten, dunklere Bereiche den etwas weiter entfernten Objekten) mit der extrahierten Skelettstruktur (rote Linien) überlagert dargestellt.
- Auswertung und Visualisierung der Gesten (links u. Mitte unten in Abb. 2)
  Hier werden je zehn bereits trainierte Gesten über ihren jeweiligen Bewegungsablauf als Skelettstruktur gezeigt. Der grüne bzw. rote Balken direkt unterhalb des jeweiligen Skelettes visualisiert die Ähnlichkeit der jeweiligen Geste mit dem aktuellen Bewegungsablauf des Benutzers, der im Tiefenbild des Sensors wahrgenommen wird. Neben der Farbe werden die Balken in Abhängigkeit der Übereinstimmung unterschiedlich lang dargestellt. Ein kurzer roter Balken bedeutet z.B., dass es nahezu keine Übereinstimmung des gesuchten und des erkannten Bewegungsablaufs gibt. Sobald der Balken länger wird und auf grün wechselt, wird symbolisiert, dass die Geste ähnlicher ist.
- Anzeige der gefundenen Geste (Mitte oben in Abb. 2) Falls eine Geste erkannt wurde, wird diese hier angezeigt. In Bild wurde z.B. festgestellt, dass der trainierte Bewegungsablauf mit ID 0 erkannt wurde und zwar mit einer Geschwindigkeit, die dem trainierten Ablauf ähnlich war (similar).





ganz oben:
Abbildung 2: Kinetic Space Programm in der „Space“ Ansicht.

oben:
Abbildung 3: Kinetic Space Programm in der „Lab“ Ansicht.

Die „Lab“ Ansicht (*Abbildung 3*) unterteilt sich in vier Bereiche:

- Tiefenbild (rechts oben in Abb. 3) wie bereits in der „Space“ Ansicht beschrieben
- Gestenanalyse und Best Match (rechts unten in Abb. 3) In diesem Bereich wird die Analyse des Dynamic-Time-Warping Algorithmus (Funktionsweise wird noch besprochen) grafisch dargestellt und der optimale Pfad zwischen der ausgeführten Bewegung (current input) und der gelernten Geste (gesture), in Rot, hervorgehoben. Direkt darunter befindet sich ein Balken, dessen Länge den Grad der Übereinstimmung zwischen der ausgeführten Bewegung und der gelernten Geste anzeigt.
- Visualisierung der Gesten (Mitte in Abb. 3) Hier wird die „aktuelle“ Geste (ID oben links) über ihren jeweiligen Bewegungsablauf als Skelettstruktur abgespielt.
- Analyse der einzelnen Körperteile (links in Abb. 3) Die Skizze einer Person zeigt anhand von Farbwerten für jeden Körperteil, ob die ausgeführte Bewegung der gelernten Geste entspricht (Körperteil in Grün dargestellt) oder davon abweicht (in Rot dargestellt).

Lernen, Erkennen und Analysieren von Gesten

Dieser Abschnitt gibt einen kurzen Überblick über die Funktionsweise der Gestenerkennung und beschreibt anhand der Software, wie die einzelnen Arbeitsschritte in der Software verwendet werden können.

Die Analyse und Klassifikation von Gesten ist ein Vergleich zwischen Mustern mit zeitlichem Verlauf. Die Arbeitsweise vieler Mustererkennungsalgorithmen besteht aus drei grundlegenden Verarbeitungsschritten, wie in *Abbildung 4* dargestellt, die sich in die Arbeitsphase und die Lernphase unterteilen lassen.

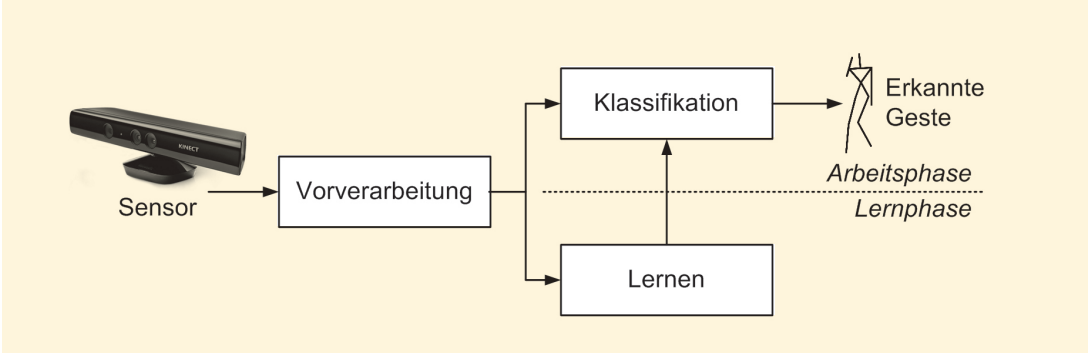


Abbildung 4: Grundlegende Verarbeitungsschritte vieler Mustererkennungsalgorithmen.

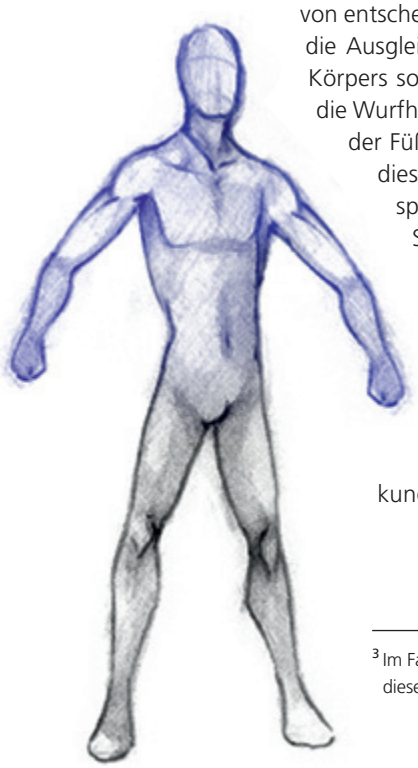
Die Mustererkennung besteht aus

- *der Vorverarbeitung,* die sowohl in der Arbeits- als auch in der Lernphase durchgeführt werden muss, um die zur Verfügung stehenden hochdimensionalen Rohdaten des Sensors in einen niedrigdimensionalen Merkmalsraum zu transformieren (bei unserer Anwendung ist dies die Überführung des Tiefenbildes in die 3D-Daten der Skelettstruktur)
- *dem Lernen,* das nur in der Lernphase benötigt wird, um anhand von markierten Daten, die im Merkmalsraum vorliegen, klassenrelevante Eigenschaften zu finden und zu speichern [3]
- *der Klassifikation,* die nur in der Arbeitsphase durchgeführt wird und die gelernte/gespeicherte Zuordnung der Merkmale mit den aktuell am Sensor zur Verfügung stehenden Daten abgleicht, um diese einer Klasse (oder keiner bei zu geringer Übereinstimmung aller Klassen) zuzuordnen.

Lernen: Trainieren und Justieren

Bevor eine Geste automatisch klassifiziert werden kann, muss diese erst einmal trainiert und justiert werden. Unter Training versteht man das Vormachen einer Geste. Unter Justieren das Verändern von freien Parametern mit Hilfe von Expertenwissen. So ist, um ein Beispiel zu geben, bei der Erkennung einer Wurfgeste die Wurfhand von entscheidender Bedeutung, die zweite Hand eventuell noch von Bedeutung, um die Ausgleichsbewegung zu berücksichtigen, aber die Stellung der Füße und des Körpers sollte hier vernachlässigt werden. So würde der Experte viel Gewicht auf die Wurfhand legen, etwas weniger Gewicht auf die zweite Hand und das Gewicht der Füße und des Körpers so einstellen, dass diese Körperteile bei der Analyse dieser Geste nicht berücksichtigt werden. Eine Einstellung für das Wurfbeispiel zeigt *Abbildung 5*. Blau steht hier für hohes Gewicht, Schwarz (in der Software Weiß) für niedriges Gewicht. Hier sei angemerkt, dass die Justage der Gewichtungen auch ein iterativer Prozess sein kann, d.h. die freien Parameter werden so lange angepasst und die Geste analysiert, bis man mit dem Ergebnis zufrieden ist oder die Geste verwirft.

Abbildung 5: Gewichtsfaktoren.



In einer idealen Welt sind die extrahierten Merkmale rauschfrei, zeitinvariant, personen- und orientierungsunabhängig und robust gegenüber anderen Störungen wie z.B. Verdeckung oder Lichteinfall. Natürlich existiert diese heile Welt nicht und man muss mit diesen Auswirkungen so gut wie möglich leben. In der Regel bedeutet dies, dass versucht >

[3] Im Fall von unmarkierten Daten müssen diese erst in Klassen unterteilt werden.

wird, diese unerwünschten Nebeneffekte mit Hilfe bestimmter Algorithmen zu kompensieren. So kann die Größe einer Person wie auch ihre Orientierung zum Sensor durch einfache Matrixtransformationen (Skalierung und Rotation) so normiert werden, dass die extrahierten Merkmale auf eine Durchschnittsperson, die frontal zum Sensor ausgerichtet ist, abgebildet werden.

Klassifizieren: Erkennen und Analysieren

Da eine Pose (statische Geste) ohne die Berücksichtigung des zeitlichen Verlaufs erkannt werden kann, ist eine einfache Implementierung unter Verwendung von Standard-Techniken der Mustererkennung wie einer Kostenfunktion, der Neuronalen Netze oder von Supportvektormaschinen möglich. Um jedoch Bewegungen (dynamische Gesten) des Körpers zu analysieren, müssen Verfahren zum Einsatz kommen, die zusätzlich den zeitlichen Verlauf mitberücksichtigen. Bekannte Verfahren sind hier das Dynamic Time Warping (DTW) oder die Hidden-Markov-Modelle (HMM), die die Ergebnisse einer Kostenfunktion, eines Neuronalen Netzes oder einer Supportvektormaschine weiterverarbeiten.

Aber bevor wir uns intensiver mit der Erkennung von Gesten beschäftigen, schauen wir uns noch eine Kostenfunktion an. Also ein Verfahren, welches die Ähnlichkeit zwischen Posen berechnet. Dieser Schritt ist insofern wichtig, als man eine Geste als Zusammensetzung vieler Posen betrachten kann. Eine der am weitest verbreiteten Kostenfunktionen zur Berechnung der Differenz zwischen dem Referenzsignal (gespeicherte Pose) und dem Testsignal ist der euklidische Abstand. Dies bedeutet, dass unter Voraussetzung des kartesischen Koordinatensystems der euklidische Abstand mit Hilfe des Satzes von Pythagoras berechnet werden kann. Er entspricht also dem Abstand zweier Punkte im Raum, wie wir es aus unserer Erfahrung/Umgebung kennen.

Das Skelett einer Person besteht aus einer Reihe von Bezugspaaren. Um für mehrere Bezugspaare einen gemeinsamen Abstand (und somit die Gesamtkosten) zu berechnen, können die einzelnen Abstände der Bezugspaare gewichtet aufsummiert werden. Die Gewichtung entspricht hier der im vorigen Abschnitt besprochenen Fokussierung auf bestimmte Körperteile. *Abbildung 6* zeigt die Auswertung der einzelnen Körperteile wie sie bei der Analyse einer Pose (Übereinstimmung der aktuellen Körperhaltung des Benutzers mit der ähnlichsten Pose innerhalb einer Geste) in der Software zu sehen ist. Grün symbolisiert eine gute Übereinstimmung und Rot eine schlechte.

Abbildung 6: Visualisierung der Abweichung zwischen zwei Posen.

Die Berechnung zwischen Posen soll nun so erweitert werden, dass zwei zeitliche Abläufe von Posen, die in ihrer Geschwindigkeit variieren, verglichen werden können. Das Dynamic Time Warping ist ein einfacher, aber leistungsstarker Algorithmus, der genau dies umsetzt. Die grundlegende Idee ist hier, die beiden Zeitreihen nicht als Ganzes zu betrachten, sondern den Vergleich der Reihen lokal (also für einen kleinen Abschnitt) zu optimieren und so Zeitunterschiede zu kompensieren. Durch das Übertragen der lokalen Ergebnisse in eine Matrix, die auf der einen Achse die Frames der Referenzgeste und auf der anderen die Frames der aktuell zu analysierenden Geste enthält, lässt sich der für zeitliche Unterschiede kompensierte Vergleich der zwei Gesten abbilden. In der Matrix, siehe *Abbildung 7*, entsteht so ein Muster aus hellen und dunklen Werten, die die jeweiligen minimalen Kosten (hier wird eine lokale Optimierungsentscheidung getroffen) pro Übergang repräsentieren. Erlaubte Übergänge sind entweder diagonal (gleiche Geschwindigkeit), horizontal (langsamer) oder vertikal (schneller). Durch Rückverfolgen des Weges entsteht ein Pfad, der den optimalen Weg durch die Matrix repräsentiert, was wiederum einem Vergleich der beiden Gesten bei geringsten Kosten (größter Ähnlichkeit) entspricht. Sind diese Kosten geringer (als Balkendiagramm visualisiert) als ein zuvor festgelegter Schwellwert, wird eine Geste erkannt, andernfalls nicht. Um gleichzeitig verschiedene Gesten auswerten zu können, wird dieses Verfahren auf alle zum Vergleich stehende Gesten angewendet.

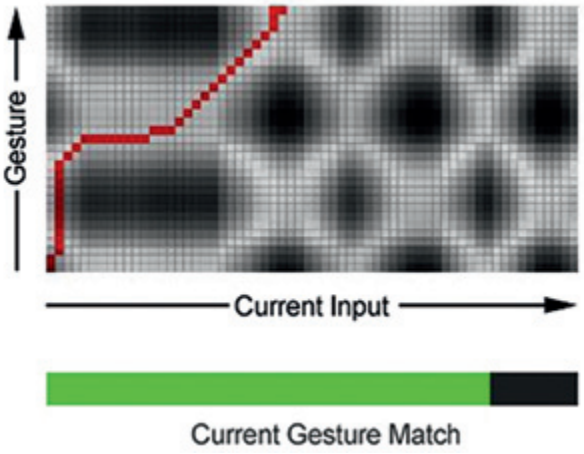


Abbildung 7: Kostenmatrix und Gesamtkosten beim Vergleich zweier Gesten mit optimalem Pfad.

Anwendungsbeispiele

In diesem Abschnitt stellen wir noch einige Anwendungsbeispiele vor, in denen die Software Kinetic Space bereits zum Einsatz kam – teilweise noch im Prototypenstatus, aber meistens bereits in Produkten und künstlerischen Installationen oder Performances.

Tanz, Musik und Performance

Durch die Analyse der Bewegung können akustische oder optische Ereignisse gesteuert werden. Entweder wird dies durch das Erkennen einer bestimmten Geste ausgelöst oder die Position eines Körperteils wird direkt abgebildet, um z.B. den Pitch oder den Beat zu verändern.

Spiel

Eigentlich die „klassische" Anwendung, für die der Kinect Sensor von Microsoft konzipiert wurde. Kinetic Space kam hier unter anderem bei einem Yeti Schneeballschlachtspiel mit zwei Kinects für bis zu acht Spielern zum Einsatz, um Wurfgesten zu erkennen und die Flugrichtung des Balls zu bestimmen.

Gesundheitswesen

Die detaillierte Analyse der Bewegung wird hier verwendet, um die Übungen in der Krankengymnastik zu kontrollieren, zu analysieren und gegebenenfalls steuernd einzugreifen.

Smartroom und Smartkitchen

Zu Hause werden hier durch verschieden Gesten Licht, Heizung und diverse Medien (TV, Beamer, Infoterminal, ...) gesteuert (http://www.openarch.cc). Eine weitere Anwendung ist das Auslösen eines Notrufes, so kann z.B. eine Person, die hingefallen ist, durch Ausführen einer bestimmten Geste mit der Notrufzentrale Kontakt aufnehmen.

Interaktion mit Humanoiden Robotern

Die Körpergestik der Zuschauer wird mit „klassischen" Posen, die aus der Kunstgeschichte bekannt sind, verglichen. Bei Auftreten einer ähnlichen Pose wird diese durch einen Humanoiden Roboter nachgebildet. (http://amorphicrobotworks.org/works/skelli/index.htm).

Hands On

Die Software wurde in Processing entwickelt und verwendet OpenNI und NITE. Sie ist unter Windows, Mac OS X und Linux lauffähig. Eine breite Palette von Drittanbietersoftware wie z.B. Max/MSP, Pure Data, VVVV und Resolume, als auch Hardware wurde bereits von Kinetic Space, über das OSC-Protokoll, gesteuert.

Das Programm einschließlich einer ausführlichen Dokumentation (auf Englisch) kann von der Seite http://kineticspace.googlecode.com heruntergeladen werden. Hier gibt es auch weitere aktuelle Beispiele, wofür Kinetic Space bisher eingesetzt wurde, sowie Informationen zu relevanten Vorträgen, Konferenzen und Festivals. Hier finden sich auch Verweise auf Videos, in denen die Funktionalität der Software im Detail vorgestellt wird.

Kinetic Space in der Presse

[…] What is it good for? Well, it can read a gesture from one person and register it on another and you can train it to register tiny movements and, potentially, allow for full motion control of your PC. Minority Report it isn't, but that future is getting closer and closer.

– www.techcrunch.com 2011/08/05

Map your gestures and have it ready for other people to use and make reference of. The Kinect Spaces is a gesture recognition and mapping software that gives programmers a reliable tool in creating and saving gestures via Kinect for their software use. This video by Matthias Wölfel shows us the nitty-gritty details of the program as well as how the Kinect Space can prove to be valuable to all Kinect hackers out there. […] This is indeed a great addition in further enhancing the Kinect controls!

– www.kinecthacks.com 2011/07/21

Ausblick

Kinetic Space ist eine Software, die bereits in der Lage ist, viele menschliche Gesten zuverlässig zu erkennen. Bisher werden hier aber weder die Orientierung des Kopfes berücksichtigt, noch die Position der einzelnen Finger analysiert. Beides sind wichtige Bestandteile, um die Gestenerkennung noch weiter zu verfeinern. Diese Funktionen sollen in einer neuen Version von Kinetic Space integriert werden. Um diese Ziele zu erreichen, ist zum einen die Integration von Algorithmen zur Analyse der Kopforientierung nötig, zum anderen eine Verbesserung in der Auflösungsgenauigkeit des Tiefenbildes, um die Finger besser extrahieren zu können. Da bei nur einem Sensor das Sichtfeld begrenzt ist, soll die Software auch dahin weiterentwickelt werden, dass sie die Information von mehreren Sensoren gleichzeitig analysieren kann

Dr. Matthias Wölfel

ist Professor für Interaction- und Interfacedesign im Studiengang Intermediales Design der Fakultät für Gestaltung.