

Handgestenerkennung für Media Browsing

Hendrik Radke (radke@rz.uni-potsdam.de)
Marko Tätzelt (mtaetzel@cs.uni-potsdam.de)

Showcase nichtlineare Medien, WS2005/06, Universität Potsdam

In diesem Projekt soll eine Steuerung des Computers mittels Handgesten ermöglicht werden. Die Handgesten des Benutzers werden dabei über eine Kamera eingelesen und in mehreren Schritten zu einer Gesten-Repräsentation weiterverarbeitet, die zur Applikationssteuerung genutzt werden kann.

Inhaltsverzeichnis

1	Projektbeschreibung	2
2	Vergleich mit bestehenden Lösungen	2
3	Lösungsidee	3
4	Durchführung	4
4.1	Verwendete Software-Bibliotheken	4
4.2	Segmentierung der Hand	4
4.3	Matching der Gesten	8
4.3.1	Aufnahme und Speicherung der Muster-Gesten	8
4.3.2	Vergleichsfunktion beim Matching	8
5	Beschreibung des erreichten Ziels	9
5.1	wxSpellcast-gesturized	9
5.2	Beschreibung des Lern- und des Erkennungsprogramms	10
6	Fazit und Ausblick	11
7	Quellenverzeichnis	12

1 Projektbeschreibung

In diesem Projekt geht es um Benutzerinteraktion mittels Handgesten. Als Fingerstellung bezeichnen wir hier eine bestimmte Anordnung bzw. Positionierung der Finger einer Hand innerhalb eines Einzelbildes. Diese ist (idealerweise) invariant gegenüber Position, Rotation und Skalierung. Das zu realisierende System liest Handgesten als Bilder von einer Kamera ein und wandelt diese in mehreren Schritten in eine Gesten-Repräsentation um, die folgende (voneinander unabhängige) Merkmale enthält:

1. Die Position der Hand im (2D-)Kamera-Koordinatensystem. Dabei wird der jeweilige Handmittelpunkt als Anker benutzt.
2. Den Winkel der Geste in Grad; 0° entsprechen dabei der Funktion $f(x) = 0$ im zweidimensionalen Kamera-Koordinatensystem.
3. Eine Zeichenkette, die die erkannte Fingerstellung (siehe oben) repräsentiert, z.B. „Zeigen“ oder „Faust“. Diese Zeichenkette wird aus dem Vergleich der Geste mit bereits abgelegten Mustern ermittelt.
4. Die Abweichung der erkannten Geste von dem Vergleichsmuster.

Diese Merkmale können nun als Eingabe für verschiedenste Software dienen. Im Rahmen dieses Projektes soll zunächst das Spiel Spellcast (siehe unten) auf Handgestensteuerung umgestellt werden. Weitere Anwendungen, so z.B. ein Browser-Plugin zur Steuerung von Mozilla Firefox oder ein „Maustreiber“, der die Maus als Eingabegerät ersetzt, konnten aus Zeitmangel leider nicht realisiert werden.

Das angefertigte Programmsystem sollte nach Möglichkeit plattformunabhängig (im Sinne von Quellcode-Kompatibilität) sein, zumindest aber unter Linux und Windows kompilierbar.

2 Vergleich mit bestehenden Lösungen

Zum Thema Gestenerkennung gibt es bereits diverse Lösungen mit unterschiedlichsten Ansätzen; es existiert sogar eine Linksammlung unter [1]. Der hier verwendete Ansatz stammt von Leonello Tarabella [2] und wurde 2004 auf der Konferenz für „Computer Music Modelling and Retrieval“ als intuitive Steuerung für Musikinstrumente vorgestellt. Die Webseite von Herrn Tarabella [3] zeigt einige Anwendungen seines Systems „Handel“.

Ein weiteres Anwendungsgebiet zeigt [4]: Hier können mit den Händen Objekte in einer virtuellen Welt aufgenommen und an anderer Stelle wieder platziert werden. In [5] hingegen werden eigentlich keine Gesten erkannt, sondern nur verfolgt und wiedergegeben; dennoch ist diese Arbeit bezüglich der Handsegmentierung von großem Interesse.

Das Projekt [6] hingegen verfolgt ein ähnliches Ziel: die Erkennung von bewegten Gesten und Fingerstellungen. Es bedient sich dabei jedoch eines ganz anderen Ansatzes als unser Projekt, nämlich der Principal Component Analysis in Verbindung mit Graph Matching.

Alle vorgestellten Verfahren haben jedoch den Nachteil, nicht rotationsinvariant zu sein. Genau dieses Problem wird aber durch Tarabellas Algorithmus gelöst. Damit werden nicht nur Schwierigkeiten bei der Erkennung beseitigt, sondern auch ein weiteres, unabhängiges Merkmal zur Steuerung – nämlich der Winkel – zur Verfügung gestellt.

3 Lösungsidee

Nach dem Einlesen der Hand über eine Video-Grabbing-Schnittstelle wird durch geeignete Filter und Segmentierungsalgorithmen ein Schwarz-Weiß-Bild erzeugt, das nur noch die Hand als weiße Silhouette auf schwarzem Grund enthält. Dabei kommen verschiedene Segmentierungsalgorithmen in Frage:

1. Handsegmentierung durch die Auswahl von Farbbereichen der Hand im RGB-Farbraum.
2. Handsegmentierung durch die Auswahl von Farbbereichen im HSV-Farbraum. Damit soll der Helligkeitsfaktor vernachlässigt werden können.
3. Subtraktion der Hand von einem vordefinierten Hintergrund („Bluescreen“).

Die ersten beiden Methoden sind dabei der dritten vorzuziehen, da keine Hilfsmittel, wie z.B. ein blauer Hintergrund oder weiße Handschuhe, benutzt werden müssen. Als einfachste Lösung bietet sich natürlich die in der Praxis sehr häufig und erfolgreich eingesetzte dritte Methode an, die deutlich einfacher zu filtern ist. Um kleinere weiße (oder schwarze) Pixelgruppen, die sich trotz schon erwähnter Filter immernoch im Bild befinden können, zu entfernen, soll ein Schwellenwert- oder ein Weichzeichner(Blur)-Filter zum Einsatz kommen. Ein Median-Blur-Filter entfernt kleinere punktförmige Gebilde und dünne Linien, ohne das Bild unscharf zu machen und ohne neue Grauwerte zu erzeugen (Rauschminderung – siehe [7]). Nun sollten alle weißen Pixel eine zusammenhängende Fläche bilden.

Danach wird der Mittelpunkt der Hand bestimmt. Von diesem Punkt wird dann strahlenförmig jeweils die (maximale) Entfernung zum Rand der Handfläche gemessen (Abb. 1). Nun ist das Ziel, eine Repräsentation der Fingerhaltung zu erhalten, die unabhängig von der Handposition, -Rotation und -Skalierung im zweidimensionalen Kamerakoordinatensystem ist. Unabhängigkeit von der Position wird erreicht, indem die Messstrahlen stets relativ zur Hand ermittelt werden. Um ebenfalls Unabhängigkeit gegenüber der Skalierung zu erreichen, müssen lediglich die Datenpunkte genormt werden, so dass skalierte Versionen der gleichen Fingerstellung die gleichen (normierten) Entfernungswerte ergeben.

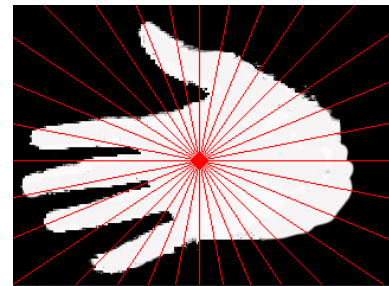


Abb. 1: Hand mit 32 radialen Scanlines

Invarianz gegenüber Rotation ist ebenfalls gegeben: Da die Entfernungen genau vom Zentrum der Hand aus ermittelt werden, entspricht eine Drehung der Hand einer Phasenverschiebung der Messpunkte, die herausgerechnet werden kann. Dazu bedienen wir uns

der Fast Fourier-Transformation (FFT – siehe z.B. [8], [9]). Für dieses Verfahren werden die gemessenen Werte als Funktion $f : N \rightarrow R$ interpretiert. Die Fourier-Transformation approximiert nun diese Funktion als Überlagerung von Sinusschwingungen verschiedener Frequenzen (Abb. 2). Da die Werte nun normiert und phasenunabhängig sind, ist die Geste somit unabhängig gegenüber Handposition, -Rotation und -Skalierung.

Diese Harmonien symbolisieren also eine bestimmte Fingerstellung und können mit vorliegenden Identifizierungsmustern verglichen werden. Dazu werden die Unterschiede der eingelesenen Geste G mit den Vergleichsmustern geeignet als Skalar $c, 0 \leq c \leq 1$ dargestellt. Die Geste mit der kleinsten Abstandsnorm ist dann die erkannte Geste.

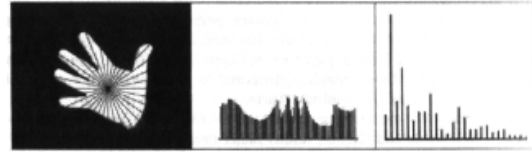


Abb. 2: Hand, Scanlines und Fourier-Transformation.

4 Durchführung

4.1 Verwendete Software-Bibliotheken

Das Projekt ist in C++ realisiert. Um die Plattformunabhängigkeit zu gewährleisten, werden ausschließlich Bibliotheken verwendet, die zumindest eine Unterstützung für Windows und Linux bieten.

Die Schnittstelle zum Einlesen der Bilder sowie zum Filtern und Segmentieren des Bildes bildet die openCV-Bibliothek [10]. Für die Anwendung der Fourier-Transformation auf die Datensätze wird die FFTW3 [9] benutzt. Eine mit der in OpenCV enthaltenen GUI-Bibliothek HighGUI implementierte Oberfläche dient zur Anzeige der Zwischenschritte des Erkennungsprozesses.

4.2 Segmentierung der Hand

Wie schon beschrieben, wollen wir die Segmentierung der Hand, also die klare Trennung der Silhouette einer Hand vom restlichen Bildinhalt, durch die Verwendung von Farbfiltern erreichen. Als ersten Ansatz untersuchten wir dazu die Farbwerte einer Hand auf mehreren Fotos. In einem der ersten implementierten Farbfilter dienten diese Werte zur Klassifizierung von Pixeln. In diesem eher trivialen Ansatz zählten wir alle Pixel zur Handfläche, die RGB-Werte besitzen, die gar nicht oder nur wenig von den vorher gemessenen RGB-Werten abweichen.

In Abbildung 3 ist die Ausgabe eines so implementierten Farbfilters veranschaulicht. Für die Ausgabe der Pixel, die nach diesem Filter der Hand entsprechen, wurde hier der ursprüngliche Farbwert beibehalten; in der Segmentierung werden dann alle nicht-schwarzen Pixel weiß gefärbt. Wie auf dem Bild gut zu erkennen ist, werden zum einen viele weitere Pixel als der Hand zugehörig erkannt, obwohl



Abb. 3: RGB-Filter

diese eigentlich zum Hintergrund gehören. Zum anderen fällt dieser Filter durch große Abhängigkeit von der Beleuchtung auf.

Verändert sich der Lichteinfall – dazu genügt schon geringer Schattenwurf in Bereichen der Hand – zieht dies unweigerlich auch eine Verschiebung der RGB-Werte betroffener Pixel nach sich. Dadurch fallen die betroffenen Hand-Bereiche nicht mehr in den Farbbereich, den der Filter einer Hand zuordnet (schwarz gefärbt). Ein Filter im RGB-Farbraum erscheint uns aus diesen Gründen ungeeignet.

Der Segmentierungsalgorithmus der Hand vom Hintergrund sollte ursprünglich allein über das Merkmal der Hautfarbe arbeiten. Wie in [11] beschrieben ist, finden sich die Hautfarben aller ethnischen Gruppen in einem kleinen Gebiet im HSV-Farbraum. Gestützt auf o.g. Arbeit und weitere([12], [13]) lässt sich somit ein einfacher Filter entwickeln. Der HSV-Farbraum (Abb. 4) teilt eine Farbe in 3 Komponenten auf: Den Helligkeitswert V (vertikale Achse), den Farbton H (Hue) als Winkel und den Sättigungswert S der Farbe (Radius). Der Mittelpunkt des Rot-Bereichs markiert den Hue Wert 0° . Der Bericht [14] zeigt, dass Variationen der Hautfarben verschiedener Menschen hauptsächlich auf Unterschiede in der *Intensität*(also Sättigung und Helligkeit) der Farbe und weniger auf den Farbwert zurückzuführen ist. Das macht die Arbeit eines Filters in diesem Farbraum erheblich einfacher da nur noch Intervalle der drei Komponenten berücksichtigt werden müssen. Nach der Diplomarbeit [5] liegen die Farbwerte(Hue) einer Hand im HSV-Farbraum im Bereich von 6° bis 38° . Auf diese Untersuchungen gestützt, sind alle weiteren Handfilter in diesem Farbraum realisiert.

Im Video 3 ist die Ausgabe eines verbesserten Filters dokumentiert, der mit Farbwerten im HSV-Raum arbeitet. Die Hand wird vollständig erkannt, jedoch auch sehr viele Anteile des Hintergrunds, die den Farbraum der Hand schneiden. Der Hintergrund kann also mit Ausnahme bestimmter Farben (die auch in Händen vorkommen) beliebig sein. Im Vergleich zu vorigen Versuchen wird hier auch die Sättigung der Farbe berücksichtigt. Grund dafür ist die Schwierigkeit, Farben mit geringer Sättigung noch einen Hue zuzuweisen – was standardgemäß durch Zuweisung des Farbwertes 0 (rot) geschieht. In Video 4 agiert ein Filter, der in dem Farbbereich arbeitet wie in [5] beschrieben (Hue: 6° - 38°). Es ist zu erkennen, dass auch hier die ganze Hand plus Teile des Hintergrunds als Hand erkannt wird. Auch ein Median-Blur-Filter der openCV-Bibliothek konnte hier keine Abhilfe schaffen, aber eine Weiterverwendung dieser Art der Segmentierung in unserem Programm kann nur erfolgen, wenn der Filter sehr wenig Rauschen erzeugt. Wenige, kurzweilige Blöcke von ein bis zwei Pixeln haben noch keine große Ungenauigkeit zur Folge – dieser Filter aber erzeugt im Durchschnitt wesentlich mehr.

Die vorläufigen Ergebnisse der Brauchbarkeit von Farbfiltern im HSV-Farbraum für unser Programm sind in Tabelle 1 veranschaulicht. Es wurden Versuche unter verschiedenen Lichtverhältnissen durchgeführt, wobei der Hue Wert periodisch verändert wurde.

Diese Tests zeigten uns, dass der Hue allein nicht ausreicht, um über das Merkmal Hautfarbe eine Hand zu erkennen.

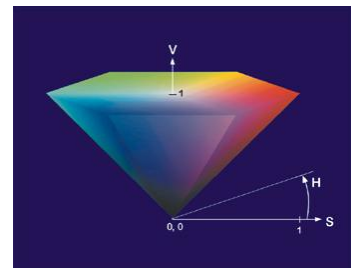


Abb. 4: HSV-Farbmodell

Hue[°]	Ergebnis
[6, 38]	schlecht, keine Segmentierung möglich, auch durch zusätzliche Filter keine Verbesserungen
[0, 26]	nur bei schlechten Lichtverhältnissen besser
[-10, 40]	besser, aber viel Rauschen, das sich auch durch weitere Filter nicht beseitigen ließ
[10, 80]	vorteilhafter für bestimmte Lichtverhältnisse, aber insgesamt nicht zufriedenstellend

Tabelle 1: Effekt verschiedener Hue-Intervalle

Der Ansatz, einen Farbfilter im HSV-Farbraum zu verwenden, erschien aber weiterhin sinnvoll und ausbaufähig. Daraufhin beschlossen wir, eigene Messreihen unter Berücksichtigung der Sättigung zu erstellen, um dann mit diesen Werten bessere Ergebnisse zu erzielen. Es wurden Fotos mit der Webcam von Händen vor verschiedenen Hintergründen gemacht, die im Anschluss mit einem Bildbearbeitungsprogramm analysiert wurden.

Als Beispiel sei hier der erste Test, unter schlechten Lichtverhältnissen, gegeben:

Umgebungs-Bedingungen: relativ dunkel, künstliches Licht, wenig Lichtstreuung, variierender und vielfarbiger Hintergrund

Einstellungen der Logitech Quick Cam: kein Weißabgleich, kein Restlichtfilter, keine Farbverstärkung (automatische Empfindlichkeit ließ sich nicht dauerhaft deaktivieren)

Vorgehensweise: Im ersten Schritt wurden die Hue-Werte erfasst, die in der Hand am häufigsten vorkommen. Hue-Werte, die weniger in der Hand, sondern eher im Hintergrund zu finden sind, wurden nicht in die Menge der erfassten Hues aufgenommen. Danach wurde ermittelt, welche Sättigungswerte in Kombination mit welchen Hue-Werten auftreten. Da bestimmte Sättigungswerte häufiger in einem bestimmten Intervall von Hue-Werten auftreten, wurden fortan nur noch Kombinationen von einem Hue- mit einem Sättigungs-Bereich betrachtet.

Tabelle 3 veranschaulicht die gemessenen Hue- und Sättigungswerte der Hautfarbe verschiedener Hände. Die Häufigkeitsverteilung blieb dabei weitestgehend unberücksichtigt.

Die Spanne von 46° bis 58° ist dabei besonders sensitiv: Ohne diesen Bereich ließ sich keine Hand vollständig erkennen; nur eine Vergrößerung des Sättigungsbereichs in diesem Hue-Winkel um 2 bis 3% verursacht aber sehr starkes Rauschen.

Einige Hue-Winkel besitzen fast gleiche Sättigungswerte, so dass diese Bereiche unter Einbeziehung von Toleranzen zusammengefasst werden konnten:

Trotz Verwendung dieser Werte sorgt die automatische Empfindlichkeitsregelung	Hue	[8, 21]	[22, 29]	[30, 39]	[40, 45]	[46, 58]
	Saturation min	33	30	18	18	18
	Saturation max	50	55	36	27	23

der Kamera für Bilder mit sehr viel Rauschen vor variablem Hintergrund. Insgesamt arbeitete das System mit diesen Werten dennoch viel besser als vorher. Das Rauschen hielt sich in Grenzen, ließ sich aber nicht ganz durch das Anwenden von Schwellwert- und Blur-Filtern der openCV-Bibliothek beseitigen. Unerwartet ist das Verhalten des

Tabelle 3:

Hue	[8, 13]	[14, 17]	[18, 21]	[22, 29]	[30, 39]	[40, 45]	[46, 58]
Saturation	38	34	47	33	25	[19, 22]	[19, 22]
	40	45	40	35	26	25	23
	37	49	39	37	28	26	18
	45	37	48	41	30		
			35	50	43	24	
			38	37	34	34	
			42	44	32	35	
			50		52	29	
			51		51	33	
			45		45	19	
min	37	34	37	33	19	19	18
max	45	49	47	52	34	26	23

Filters unter anderen als bei den Messungen vorhandenen Lichtverhältnissen: Wir erwarteten eigentlich eine Farbbestimmung, die bis zu einem gewissen Grad unabhängig von den Lichtverhältnissen sein würde, gestützt auf die Arbeiten [11], [12], [13], [14]. Jedoch stellten wir fest, dass unser Filter unter anderen Lichtverhältnissen nicht nur sehr viel Rauschen verursachte, sondern auch die Handsilhouette zerklüftete (Video 5). Eventuell ließe sich diese Art der Handsegmentierung weiter verwenden, wenn man einen speziellen kombinierten Filter implementiert, der kleine punktförmige Gebilde und kurze dünne Linien entfernt, den Farbbereich toleranter gestaltet und Einschränkungen bezüglich der Farbvariabilität des Hintergrunds vorgibt.

Bei weiteren Tests stellte sich aber heraus, dass dieser Filter eine gut segmentierte Hand liefert, wenn der Hintergrund uniform blau ist, weshalb wir uns dazu entschieden, ein einfaches Bluescreen-Verfahren zu entwickeln. Der neue Filter macht nichts weiter, als alle Pixel eines bestimmten Blau-Spektrums schwarz und die noch verbleibenden Pixel weiß zu färben. Auf das Ergebnis wird ein Median-Blur-Filter angewandt, um eine zusammenhängende Hand zu erhalten sowie Störungen zu glätten. Der Vorteil bei der Verwendung des Blaufilters ist die klar und zusammenhängend gefilterte Hand, die sich gut für die weiteren Berechnungen eignet. Die ursprüngliche Forderung, mit dem Softwaresystem ohne weitere Hilfsmittel interagieren zu können, geht jedoch verloren. Es wird ein blauer Hintergrund sowie grüne, dunkelblaue oder blaue Kleidung (Ärmel) vorausgesetzt. Vor einem vielfarbigen Hintergrund erzeugt der Blaufilter sehr viele weiße Flächen, die nicht zur Hand gehören, auf dessen Grundlage keine Gestenverarbeitung möglich ist.

4.3 Matching der Gesten

Wenn die Fourier-Koeffizienten der Geste ermittelt sind, müssen diese noch mit einem abgelegten Muster verglichen werden. Interessant sind hierbei zwei Aspekte: die Aufnahme und Speicherung der Muster-Gesten sowie die Vergleichsfunktion beim Matching.

4.3.1 Aufnahme und Speicherung der Muster-Gesten

Zu diesem Zweck wurde ein einfaches „Lernprogramm“ geschaffen, das Mustergesten von der Kamera aufzeichnet und, mit einem Namen versehen, in eine Datei speichert. Um den Einfluss von Störungen bei der Aufzeichnung zu verringern, wird jede Geste mehrmals aufgezeichnet. Im Moment werden jeweils $n = 5$ „Proben“ von einer Geste genommen. Alle aufgezeichneten Werte werden dabei normalisiert, so dass sie zwischen 0 und 1 liegen. Dies macht die Vergleichsmuster unabhängig von der Auflösung der Kamera bei der Aufzeichnung. Von diesen n Proben $[x_1, \dots, x_n]$ werden die im Folgenden gelisteten Werte aufgezeichnet. Dabei ist zu beachten, dass die x_i selber wieder Vektoren darstellen, so dass die o.g. Operationen elementweise durchgeführt werden müssen.

- Minimum: $\min([x_1, \dots, x_n]) = \min_{i=1}^n \{x_i\}$
- Maximum: $\max([x_1, \dots, x_n]) = \max_{i=1}^n \{x_i\}$
- Mittelwert: $\text{avg}([x_1, \dots, x_n]) = \frac{1}{n} \sum_{i=1}^n (x_i)$
- Median: $\text{med}([x_1, \dots, x_n]) = x_{n/2}$, wobei $[x_1, \dots, x_n]$ nach Größe geordnet ist
- Varianz: $\text{var}([x_1, \dots, x_n]) = \frac{1}{n} \sum (x_i - \text{avg}(x_1, \dots, x_n))^2$

4.3.2 Vergleichsfunktion beim Matching

Als letzter Schritt der Gestenerkennung müssen die Fourier-Koeffizienten der Scanline-Funktion mit den abgelegten Mustern verglichen werden, um ein Matching mit einem dieser Muster zu erzielen. Dabei gibt die Vergleichsfunktion einen Wert c zwischen 0 und 1 zurück, der den Unterschied mit dem Vergleichsmuster angeben soll; ein Wert von 0 bezeichnet hier ein ideales Matching, während ein Wert von 1 den größtmöglichen Unterschied darstellt. Das Matching-Verfahren vergleicht die Fourier-Koeffizienten einfach mit allen Mustern und wählt dann jenes mit dem geringsten Unterschied.

Um einen Koeffizientenvektor x mit einem solchen Muster y zu vergleichen, gibt es praktisch unzählige Möglichkeiten. Im Grunde genommen handelt es sich um ein „Skalarprodukt“, das aus zwei Vektoren eine positive, reelle Zahl berechnet, die für identische Argumente 0 ist. Jedes der im Folgenden gelisteten Verfahren berechnet den Vergleichswert als Summe von komponentenweisen Vergleichen: $c = \sum_{i=1}^n c_i$, der Einfachheit halber werden im folgenden nur die komponentenweisen Vergleichsfunktionen c_i angegeben.

Das einfachste Verfahren entspricht der euklidische Norm; der Abstand berechnet sich aus $c_{\text{avg}} = (x - y.\text{avg})^2$. Leider funktioniert dieses Verfahren nicht so gut wie erwartet, da die häufig vorkommenden „Ausreißer“ unter den Werten zu großen Abständen führen.

Um solche Fehler bereits bei der Aufzeichnung einzudämmen, kann statt des Mittelwertes auch der Median verwendet werden: $c_{med} = (x - y.med)^2$. Diese Variante ist etwas unempfindlicher gegen „Ausreißer“. Aber auch hier bleibt das Problem, der starken Schwankungen.

Als dritte Möglichkeit bleibt die Verwendung von Maximum und Minimum: $c_{minmax} = \begin{cases} 0 & \text{falls } y.min < x < y.max \\ \min(x - y.min)^2, (x - y.max)^2 & \text{sonst.} \end{cases}$ Hier wird der Abstand nur dann erhöht, wenn der Koeffizient außerhalb des durch Minimum und Maximum gegebenen Bereiches liegt; man vergleicht quasi mit dem Intervall $[y.min, y.max]$. Leider führt dieses Verfahren schon bei wenigen Gesten zu einem zu „optimistischen“ Matching, so dass praktisch eine beliebige Geste erkannt wird.

Alle drei Verfahren haben also das selbe Problem: Schwankungen und unterschiedlichen Größenordnungen der Koeffizienten sowohl bei der Musteraufnahme als auch im Erkennungsprozess bedingen eine zu große Ungenauigkeit für die Zuordnung. Wenn man jedoch die Koeffizienten bei der Aufzeichnung der Muster eine Zeit lang beobachtet, so fällt auf, dass einige Koeffizienten recht stark schwanken, während andere fast konstant bleiben; diese können als „charakteristisch“ für diese Geste angesehen werden. Diese Schwankungen schlagen sich in der Varianz des Koeffizienten über mehrere Aufnahmen nieder. Charakteristische Werte haben eine niedrige Varianz, während die schwankenden Werte eine hohe Varianz aufweisen: Die Varianz ist umgekehrt proportional zur Wichtigkeit des Koeffizienten bei der Erkennung. Man könnte also die bei der Berechnung der Unterschiede die Varianz mit berücksichtigen: $c = \sum_{i=1}^n (1 - y.var_i) c_i$. Auf diese Weise kann der Einfluss von Schwankungen der nicht charakteristischen Werte stark reduziert werden.

Was aber, wenn bei allen aufgezeichneten Mustern die gleichen Koeffizienten charakteristisch sind? Schwankungen der charakteristischen Werte sorgen dann immer noch für schlechte Erkennung. Dem kann abgeholfen werden, indem man die Werte einer Mustergeste, in denen sie sich von allen anderen Mustern deutlich unterscheidet, bevorzugt behandelt. Dies wird durch einen Vergleich beim Einlesen der Musterdatei erledigt; jedem Koeffizienten eines Musters wird ein Wert g_i zugeordnet, der um so größer ist, je größer der minimale Unterschied dieses Musters im Vergleich zu den anderen ist.

Als endgültige Formel zur Berechnung des Unterschiedes zweier Gesten ergibt sich also:

$$c = \sum_{i=1}^n g_i y.var_i c_i \text{ mit } c_i = c_{avg}[i] \text{ oder } c_i = c_{med}[i] \text{ oder } c_i = c_{minmax}[i].$$

5 Beschreibung des erreichten Ziels

5.1 wxSpellcast-gesturized

Spellcast ist ein Spiel, bei dem sich 2-8 Spieler mit einem breiten Sortiment an Zaubersprüchen bekämpfen. Die ursprüngliche Papier-und-Bleistift-Version dieses Spiels ist von Richard Bartle entwickelt worden, eine wxWidgets-basierte Computerversion wurde 2003 von Dennis Taylor geschrieben. Das Spiel läuft rundenweise ab, wobei jeder Spieler in je-

der Runde je eine Geste für die linke und rechte Hand angibt. Bestimmte Kombinationen von Gesten entsprechen bestimmten Zaubern.

Die Steuerung des Spiels über die Maus soll durch eine Steuerung mittels Handgesten ersetzt werden. Dazu muss das System zwei Gesten gleichzeitig verarbeiten, was mit verschiedenen Bildregionen für die linke und rechte Hand ermöglicht wird. Zusätzlich zu den Zaubergesten werden noch weitere Gesten benötigt, um die im Spiel auftauchenden Fragen zu beantworten und die Runde zu beenden.

Der wxSpellcast-Client wurde im Rahmen dieses Projektes um eine Gestensteuerung erweitert. Beim Starten des Programmes wird zusätzlich zum üblichen Steuerungsfenster ein Kamerafenster geöffnet, das ein bereits gefiltertes Kamerabild anzeigt. Das Fenster ist in 3 Regionen eingeteilt, wobei die linke bzw. rechte Region für die entsprechende Hand genutzt werden. Die untere Region ist für die Steuerung der restlichen Programmfunktionen, wie z.B. Rundenende und das Beantworten von Fragen, vorgesehen. Die zwei Regler am oberen Rand legen zwei Filter-Parameter fest (nämlich die obere respektive untere Grenze der zu ignorierenden Hue-Werte).

5.2 Beschreibung des Lern- und des Erkennungsprogramms

Zum Testen der Gestenerkennung sowie zur Aufzeichnung von Mustergesten wurden zusätzlich zwei Programme entwickelt: Das Lernprogramm (`reader.exe`) ermöglicht das Aufzeichnen und Abspeichern von Gesten in einer Datei. Der Gesten-Datensatz kann im Erkennungsprogramm (`tester.exe`) und in Spellcast als Grundlage zur Gestenerkennung dienen.

Im Testprogramm kommt o.g. Blaufilter zum Einsatz, dessen Parameter anhand zweier Regler, wie auch in wxSpellcast-gesturized, verändert werden können. Die blauen Linien am oberen Bildrand des Videofensters bilden die Messungen der Scanlines ab. Die roten Linien in der Bildmitte stellen die Fourier-Koeffizienten dar, die auf Grundlage der Scanlines ermittelt wurden.

Um mit dem Lernprogramm (Siehe Abb. 5) eine Geste aufzunehmen und im neuen Datensatz zu speichern, muss die selbe Geste 5 mal hintereinander durch jeweiliges Betätigen der Leertaste aufgenommen werden.

Nun bleibt das Bild im Videofenster eingefroren und es werden zusätzliche Werte über die Scanlines durch verschieden farbige Balken im Videofenster angezeigt. Alle Werte beziehen sich auf die jeweiligen Scanlines der 5 Schnappschüsse der aufgenommenen Geste:

- orange Balken: Minimum
- gelbe Balken: Mittelwert (average)
- grüne Balken: Median
- pinke Balken: Maximum

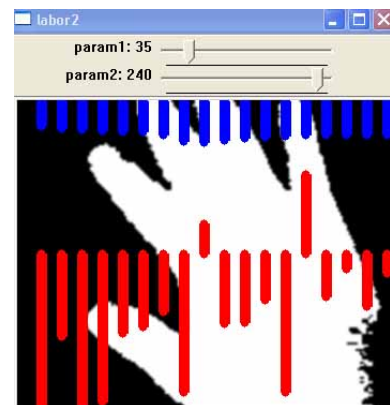


Abb. 5: Das Lernprogramm

Diese Werte werden zur weiteren Gestenerkennung benötigt. Nun muss in der Konsole

noch ein Name(ohne Leerzeichen, max. 15 Zeichen) für die neue Geste vergeben werden, bevor im Videofenster weitere Gesten aufgenommen werden können. Das Programm kann nur ordnungsgemäß durch das Drücken der Taste q im Videofenster-Fokus beendet werden.

6 Fazit und Ausblick

Leider konnte das gesteckte Ziel nicht ganz erreicht werden, obwohl die meisten Programmteile innerhalb eines gewissen Toleranzbereiches korrekt arbeiten. Das Matching der Gesten ist noch zu ungenau, um ein flüssiges und intuitives Umgehen mit der Gestenerkennung zu ermöglichen. Die Algorithmen funktionieren größtenteils gut, bis auf einige Ausnahmen, wo noch Forschungsbedarf besteht:

- Das Herausfiltern der Hand aus dem Kamerabild funktioniert vor uniform blauen Hintergründen sehr gut. Das Bluescreen-Verfahren versagt jedoch beim Ausfiltern von Hintergrund im rot-gelben Hue-Bereich sowie von reinen Grautönen. Des Weiteren ist der Algorithmus bedingt empfindlich bezüglich der Lichtverhältnisse: Wenn die Aufnahme nicht gut und vor allem gleichmäßig ausgeleuchtet ist, zeigen sich teilweise unerwünschte Störungen. Hier wäre durch Erweiterungen der Filter eine Verbesserung möglich, um auch vor nicht-blauem Hintergrund oder bei schlechtem Licht die Hand ohne Fehler identifizieren zu können.
- Die Fourier-Transformation unterliegt extrem starken Schwankungen, selbst bei geringfügigen Änderungen der ermittelten Scanlines. Hier muss eine andere rotations- und skalierungsinvariante Transformation als Ersatz gefunden werden. Im wesentlichen verursacht dieses Problem ein Abbildungsproblem von der Geste auf die für die Geste signifikanten Datensätze. Als möglicher Ersatz böten sich sogenannte Hu-invariante Momente an [Wikipedia:image moments].
- Ein weiteres großes Problem stellt das Matching der Gesten mit den Mustern dar. Die Schwankungen der Fourier-Transformation verhindern hier leider jede Möglichkeit zu einer gut funktionierenden Erkennung. Eine verbesserte Transformation wirkt sich sicherlich positiv auf das Matching aus. Verbleibende Probleme können dann mit den unter „Vergleichsfunktion beim Matching“ erläuterten Methoden gelöst werden.

7 Quellenverzeichnis

Literatur

- [1] *The Gesture Recognition Home Page*. <http://www.cybernet.com/~cohen/>, November 2005. Viele nützliche und weiterführende Links zum Thema Handgestenerkennung.
- [2] TARABELLA, LEONELLO: *Handel, a Free-Hands Gesture Recognition System*. In: WIIL, UFFE KOCK (Herausgeber): *Computer Music Modeling and Retrieval*, ISBN: 3-540-24458-1. Springer-Verlag, 2004. Dieser Artikel beschreibt das von uns verwendete Verfahren zur Gestenerkennung und war der Anstoß für die Projektidee.
- [3] TARABELLA, LEONELLO. <http://cnuce.isti.cnr.it/tarabella/music.html>, 11 2005. Die Webseite von Leonello Tarabella. Leider keine Artikel, dafür interessante Beispiele zur künstlerischen Verwendung der Handgestenerkennung.
- [4] *Vision-based hand gesture interface*. <http://www.cs.ucsb.edu/~matz/HGI/HandVu.html>. Ein Interface zur Verarbeitung von Gesten. Die Hände müssen dabei in einer Initialstellung aufgenommen werden. Interaktion mit virtuellen Objekten.
- [5] KONDRATYUK, YEVGENIYA: *Verfolgung und Segmentierung von Händen in Bildsequenzen zur Unterstützung der 3D-Analyse in einem Echtzeit-Videokonferenzsystem*. Diplomarbeit, TU Berlin, http://www.nue.tu-berlin.de/lehre/sa-da/DA_Kondratyuk.pdf, 2003. Arbeit zum Thema Verfolgung/Segmentierung von Händen – Voraussetzungen für die Gestenerkennung. Das Verfahren bedient sich dabei der Hautfarbe als Merkmal zur Segmentierung.
- [6] SHAMAIE, ATID, WU HAI und ALISTAIR SUTHERLAND: *Hand-Gesture Recognition for HCI*. http://www.ercim.org/publication/Ercim_News/enw46/shamaie.html. Ein anderer Ansatz zur Gestenerkennung, der mit Principal Component Analysis (PCA) und Graph Matching arbeitet. Das System kann auch zeitvariante Gesten erkennen.
- [7] STEINBRECHER, RAINER: *Bildverarbeitung in der Praxis*. ISBN: 3-486-22372-0. Oldenbourg Verlag, 2. Auflage, 1993. Ein Buch über die computergestützte Bilderkennung.
- [8] CORMEN, T., C. LEISERSON, R. RIVEST und C. STEIN: *Introduction to Algorithms*, Kapitel 30. ISBN: 0-262-53196-8. MIT Press, 2. Auflage, 2001. Ausführliches Kapitel zur FFT (Fast Fourier Transformation).
- [9] FRIGO, MATTEO und STEPHEN G. JOHNSON: *Fastest Fourier Transform in the West*. <http://www.fftw.org/>, November 2005. Wie der Name schon sagt, eine schnelle Bibliothek zur Fourier-Transformation.

- [10] *Intel OpenCV Library*. <http://sourceforge.net/projects/opencvlibrary>. Freie Bibliothek für Bilderkennung. Die Bibliothek enthält eine Vielzahl von nützlichen Algorithmen für Tracking, Segmentierung etc.
- [11] RAJA, Y., S.J. MCKENNA und S. GONG: *Tracking and segmenting people in varying lighting conditions using colour*. In: *IEEE International Conference on Face & Gesture Recognition*, 1998.
- [12] BRAUMANN, U.-D.: *Multi-Cue-Ansatz für ein dynamisches Auffälligkeitssystem zur visuellen Personenlokalisierung*. Doktorarbeit, Technische Universität Ilmenau, 2001.
- [13] XIAOMING, Y. und X. MING: *Finger identification in hand gesture based human-robot interaction*. In: *IEEE/RAS International Conference on Humanoid Robots*, MIT, Boston, USA, 2000.
- [14] YANG, J., W. LU und A. WAIBEL: *Skin-Color Modeling and Adaptation*, 1997.
- [15] FRITZSCH, KLAUS: *Maschinelles Sehen*. ISBN: 3-05-001785-6. Akademie Verlag, 1991. Etwas älteres Buch, das mathematische Grundlagen von Computer Vision erläutert.
- [16] OH, PAUL Y.: *Template for Real-Time Image Processing Development*. <http://www.boondog.com/tutorials/tripodBinarize/tripodBinarize.html>, 2001. Ein Template, welches den Entwurf von Echtzeit-Bildverarbeitenden Programmen erleichtern soll.
- [17] TAYLOR, DENNIS: *wxSpellcast - A game of duelling wizards*. <http://www.funkplanet.com/spellcast/>, 2003. wxSpellcast - das oben erwähnte Spiel, das auf Gestensteuerung umgestellt werden soll.
- [18] *wxWidgets*. <http://sourceforge.net/projects/wxwindows>, 2005. wxWidgets-Bibliothek - benötigt für wxSpellcast.