Jonglieren mit der Kinect

ein Softwareprojekt im Projekt Bildverarbeitung

Projektbericht

Rolf Boomgaarden
Florian Letsch
Thiemo Gries

9. April 2014

Unter Aufsicht von: **Benjamin Seppke**
Arbeitsbereich Kognitive Systeme
Fachbereich Informatik, Universität Hamburg

Inhaltsverzeichnis

1 Einleitung

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Sed aliquam, ligula vitae condimentum malesuada, turpis nisi placerat eros, vel facilisis mi neque quis nulla. Aenean eleifend risus id dolor ultricies scelerisque. Phasellus venenatis libero enim, vel lacinia massa interdum nec. Quisque a euismod ligula. In eget mattis orci. Integer vitae enim ac nisl scelerisque luctus ut et nibh. Quisque ut odio ultrices, consequat mi vel, accumsan metus. Donec faucibus, nulla vel mattis euismod, felis leo accumsan tortor, et congue turpis leo et elit. Proin gravida mollis facilisis. In enim nisi, pellentesque id tincidunt a, accumsan eget elit. Aliquam erat volutpat. In quam ante, accumsan eu est a, molestie euismod neque. Proin porta rhoncus nisl sed dignissim. Aliquam lacinia sed libero et eleifend. Ut placerat tortor eget augue pellentesque rutrum.

2 Motivation

Mit den technischen Möglichkeiten eines Tiefen- und Bilddaten liefernden Systems (konkret: Microsoft Kinect) soll in dieser Arbeit versucht werden, das Wurfmuster eines mit Bällen jonglierenden Akteurs zu analsieren.

Ein Jongleur wirft Jonglierbälle in einem Muster, das möglichst gleichmäßig ist. So ist der Höhepunkt der Flugbahn idealerweise konstant auf der gleichen Höhe. Zum Analysieren des Jongliermusters wäre dies also bereits ein erstes Kriterium, die *Güte eines Jongliermusters* automatisiert zu bewerten.

Denkbar sind auch weitere Anwendungen, wie etwa das automatische Zählen von erfolgreich gefangenen Würfen. Eine computergesteuerte Erfassung der insgesamten Wurfzahl ist ein einfaches Kriterium für eine *Leistungsbewertung des jonglierenden Benutzers*.

Die genaue Anwendung ist jedoch nicht Ziel dieser Arbeit. Stattdessen verfahren wir in einem bottom-up Herangehen, um von den rohen Bild- und Tiefendaten der Kinect ausgehend Informationen über sich im Bild befindliche Objekte (Jonglierbälle) zu erfassen und deren Bewegung zu erkennen. Das Ergebnis ist dann ein Fundament, auf dessen Grundlage konkrete Anwendungen entwickelt werden können.

3 Zielsetzung

Am Ende dieser Arbeit soll eine Anwendung stehen, die mit Hilfe der Kinect Daten über die Flugbahnen dreier jonglierter Bälle liefert.

Ein Akteur befindet sich hierbei im Bildzentrum in einem wohl definierten Abstand zur Kinect. Es werden drei matte Bälle beliebiger Farbe jongliert. Um das Ergebnis unabhängig von der Szenenbeleuchtung zu halten, sollen die Tiefendaten ausreichend Information für das eindeutige Identifizieren der Bälle liefern.

4 Möglichkeiten der Kinect

Die Kinect ist eine von Microsoft zur Spielekonsole Xbox 360 vertriebene Erweiterung, die den Spieler mit einem RGB- und einem Tiefensensor erfasst und diese beiden Datenströme an die Konsole liefert. Da die Kinect über einen USB-Anschluss verfügt, kann sie an konventionellen Rechnern angeschlossen und betrieben werden. Eine quelloffene Implementierung zur Unterstützung der Kinect ist das freenect Projekt, das für Linux, Windows und MacOS zur Verfügung steht und im Rahmen dieser Arbeit als Bibliothek für Python verwendet wurde. FIXME: Quellen.

Die Videoquelle der Kinect liefert standardmäßig 30 (JA?) Bilder pro Sekunde mit einer Auflösung von 640x480 und 8 bit Farbtiefe. Die Tiefendaten stammen von einer Infrarot-Kamera und liefern bei gleicher Bildfrequenz 2048 verschiedene Tiefenwerte (11bit). Aus der Funktionsweise der Infrarotkamera ergibt sich, dass die Kinect in Umgebungen starker Infrarotstrahlung (beispielsweise im Tageslicht) nur beschränkt einsatzfähig ist.

FIXME: Erklären wie das mit diesem projizierten Punktemuster funktioniert.

5 Recherche: Ein jonglierender Roboter

(paper...)

6 Lösungsidee

image Processing Pipeline

7 Umsetzung

7.1 Programmfluss

7.1.1 Schritt 1: Tiefendaten vorverarbeiten

Normieren aux XXX Tiefenwerte

7.1.2 Schritt 2: Regions Of Interest isolieren

Hintergrund entfernen, Bälle freistellen. Zwei Ansätze:

A. keine Objekte auf Tiefenebene zwischen Spieler und Kinect, auch nicht am Rand. Tiefenwerte aber einem gewissen Wert einfach abschneiden. Annahme: Spieler steht auf Linie oder ähnlich. Tiefendaten binarisieren.

B. Mit temporalem Filtering sich bewegende Regionen isolieren. Erlaubt auch störende Objekte wie Stühle am Rand, Erfahrung aber nicht so gut, da das Verfahren bei schneller Bewegung (Ballwurf) nicht zuverlässig ist. Außerdem Probleme mit unscharfen Objekträndern und Rauschen. Außerdem technische Hürden (Vigra als weitere Abhängigkeit).

Bewertung: Ansatz A völlig ausreichend für unsere Zwecke. Einschränkung der Spielerpositition nicht störend, da dies sogar interkativ durchgeführt werden (so lange nach vorne gehen, bis System vernünftige Werte liefert - kein aufwändiges Abmessen nötig).

7.1.3 Schritt 3: Bälle in Frame-Folgen einander zuordnen

Kurze Vorverarbeitung: Rechtecke erkennen aus binarisiertem Bild. Annahme: der Mittelpunkt jedes Rechtecks ist ein Kandidat für eine Ballposition. Die Ausdehnung und somit der Ballradius werden vorerst ignoriert.

Dies ist der aufwändigste Schritt wie sich herausgestellt hat, zumindest der, mit dessen Lösung wir die meiste Zeit verbracht haben.

Erste Idee: Regionen mit Tiefenbild bestimmen, tatsächliche Bälle von Händen etc unterscheiden, indem Kreise in den RGB Bildern gesucht werden. Dies war aber rechenaufwändig und wegen Bewegungsunschärfe sehr unzuverlässig (auch noch unterschiedlich stark je nach Fortschritt des Ballwurfs).

Problemquellen:

- Hände sind auch als Rechtecke enthalten
- Bälle fliegen sehr nah, teilweise überschneiden sich die Rechtecke zweier Bälle, so dass nur ein großes zu sehen ist und als eine mögliche Ballposition untersucht wird
- Ball legt in einem Frame (1/30 Sekunde) unterschiedlich lange Strecken zurück, teilweise sehr große (Pixelanzahl angeben?)
- Mindestabstand zur Kinect resultiert in kleinem Jongliermuster, das verstärkt die problematischen Faktoren
- teilweise fehlt eine Region in einem erkannten Frame

Ansätze:

Konsumierende Ansätze, feste Ballanzahl:

1. Nächste Punkte in zwei aufeinander folgenden Frames werden als der identische Ball aufgefasst. Nicht so zuverlässig, vor allem wegen schneller Ballbewegung

und nah aneinander fliegender Bälle. Schwierig auch, wenn ein erkannter Ball fehlt -> Beachtung von ßpringenden"Bällen.

2. Verbesserungsansatz: erwartete Ballposition wird approximiert mit vorheriger Bewegung (linearer Bewegungsvektor). Teilweise besser, aber schwierig, die initiale Bewegung zu Erkennen, auch weiterhin Probleme mit Lücken in den Informationen.
3. Verbesserung: Nicht linear, sondern Flugbahn vorberechnen. Linearer Bewegungsvektor wird als Tangente an Steigung der Wurfparabel zu Grunde gelegt. (Verbesserung nochtmal gut angucken, aber gefühlt hat das erstaunlich wenig unterschied gebracht)

Nicht konsumierende Ansätze, variable Ballanzahl:

1. wenn langsame Aufwärtsbewegung in aufeinander folgenden Frames erkannt wird: als Beginn eines Wurfes auffassen und an dieser Stelle einen Ball mit identischer Geschwindigkeit starten und dessen Flugbahn ab dort schrittweise simulieren. In jedem Schritt mit aktuell vorhandenen Bällen abgleichen und Wurfparameter anpassen. (hier schrittweise Bilder zeigen: Feuerwerk etc)

7.1.4 Schritt 4: Bereinigte Wurfparabel

Das fehlt uns noch. Aber aus den gelieferten Daten wollen wir dann höher-levelige Informationen abstrahieren. Objektanzahl, Wurfhöhen, Würfe zählen, etc.

7.2 Erläuterung verwendeter Bildverarbeitungsverfahren

7.2.1 Kalman Filter

FIXME: alles überarbeiten, mehr, nochmal nachlesen, wie's wirklich ist, Schaubilder

Der Kalman Filter kann sich bewegende Objekte beobachten und Schätzungen zur aktuellen oder auch zukünftigen Position machen.

In diesem Projekt wird er dazu verwendet, die Bälle zu beobachten und die weitere Flugbahn zu bestimmen. Damit kann eine voraussichtliche Flugbahn in die Ausgabe gezeichnet werden, aber auch in der internen Verfolgung und Zuordnung der Bälle spielen die Ergebnisse eine wichtige Rolle.

Der Kalman Filter besteht aus zwei Funktionen, einem `predict` und einem `update`. Im `predict` wird mit Informationen zur Art der Bewegung und mit mindestens einer Ortsinformation zum Zeitpunkt `t` eine Schätzung zum Ort im Zeitpunkt `t+1` gemacht. Ein `predict` kann beliebig oft hintereinander ausgeführt werden.

Im `update` wird die aktuell gespeicherte Ortsinformation mit extern gewonnen Daten aktualisiert. Die hier gemessene Abweichung zwischen Schätzung und tatsächlichen

Daten kann natürlich auch zur weiteren Schätzung eingebracht werden. Ein `update` wird nicht mehrmals hintereinander ausgeführt.

7.3 Herausforderungen

Probleme aus den Ansätzen noch mal aufgreifen. Noch irgendwas abstrakteres dazu schreiben? Vielleicht dass wir uns nicht doll genug getracked haben dier Projektzeit über?

7.4 Bewertung der Umsetzung

Robustheit.

Effizienz. Speedup-Möglichkeiten?

Anwendungsrelevanz.

8 Anwendungsmöglichkeiten

Projekte nehmen als Grundlage für einfache Programme / Spiele.

- Objekte zählen

- Würfe zählen

- ... ?

9 Fazit

Lerneffekt, Frustration, Bewertung des Endprodukts

Quellen

[1] Paul Viola, Michael Jones,
Robust Real-time Object Detection
Vancouver, Canada, 13.07.2001.
http://research.microsoft.com/en-us/um/people/viola/Pubs/Detect/violaJones_IJCV.pdf

[2] Ole Helvig Jensen,
Implementing the Viola-Jones Face Detection Algorithm
IMM-M.Sc.: ISBN 87-643-0008-0 ISSN 1601-233X
Technical University of Denmark, Informatics and Mathematical Modelling
Kongens Lyngby, Denmark, 2008.
http://www.imm.dtu.dk/English/Research/Image_Analysis_and_Computer_Graphics/Publications.aspx?lg=showcommon&id=223656