Computer Vision

Praktikum SS2009

1. Zwischenbericht

Tobias Braun, XXYYZZ

Sven Eisenhauer, 707173

Vorbemerkungen

Die Installation des bereitgestellten WISP-Setups verlief größtenteils problemlos. Einzig bei der WISP-Version 2008 trat auf dem privaten Rechner von Sven Eisenhauer ein Kompilier-Fehler auf. Dieser lies sich leicht durch einfügen der Zeile

#define _STLP_STATIC_CONST_INIT_BUG 1

in der Datei \STLport-5.1.6\stlport/stl/config/user_config.h beheben.

Weiterhin trat auf einem anderen Rechner ein Laufzeit-Fehler bei der Verwendung von WISP auf. Nach einiger Zeit kam es zu einem Deadlock, vermutlich innerhalb von WISP. Das System meldete keinen Fehler, es kam zu keinem Absturz. Auch konnte der Fehler nicht genau reproduziert werden. Nach Installation der neuesten ATI-Grafikkarten-Treiber für das betroffene System trat der Fehler nicht mehr auf. Seitdem stellt WISP eine stabile Entwicklungs-Umgebung dar. Leider hat dieser sporadisch auftretende Fehler unnötig viel Zeit verschlungen.

Die Durchführung des Praktikums erfolgt auf Basis des virtuellen Aibos aus dem WISP-Paket.

Aktueller Stand

Verfolgung des Pfads

Wir haben uns dazu entschieden, mit einem einfachen Ansatz zur Verfolgung des vorgegebenen Pfads zu beginnen, um schneller zu ersten Erkenntnissen zu gelangen. Auf Basis dieser Erkenntnisse werden wir gegebenenfalls diesen Ansatz weiter verfeinern.

Bei diesem einfachen Ansatz werten wir nur eine Bildzeile aus, wie in den Tipps zum Praktikum erwähnt. Die betrachtete Bildzeile befindet sich bei ¾ der Bildhöhe, also im unteren Viertel des Bildes. Wie ebenfalls in den Tipps erwähnt, transformieren wir das Kamerabild in das HSV-Modell und extrahieren die Helligkeitskomponente. Die weiteren Schritte verwenden dieses Grauwert-Bild als Ausgangspunkt.

Zum Auffinden des Linien-Mittelpunkts betrachten wir von links jedes Pixel der Bildzeile. Das erste Pixel mit dem Helligkeitswert der Pfadlinie stellt den linken Rand der Farblinie fest. Von hier detektieren wir das Pixel, das nicht mehr diesen Helligkeitswert besitzt. Dieses stellt den rechten Rand dar. Reicht die Pfadlinie bis an den rechten Rand des Bilds, stellt der rechte Rand des Bilds die rechte Begrenzung der Pfadlinie dar. Dementsprechend ergibt sich daraus die Mitte des Pfades

als $m = linker\ Rand + (\frac{rechter\ Rand - linker\ Rand}{2})$

Auf Basis des Wertes m wird der Drehungswinkel des Aibos vor dem nächsten Schritt berechnet. Die berechnete Drehung wird durchgeührt und dein ein Schritt von 0,5 vorwärts gemacht. Sollte in der betrachteten Bildzeile die Pfadlinie nicht erkannt werden, führt der Aibo keine Drehung aus und bewegt sich nur um den Wert 0,5 vorwärts.

Kollisionsvorhersage / -vermeidung

Zur Vermeidung von Kollisionen des Aibos mit Objekten auf dem Parcours müssen diese zuerst erkannt und in einen räumlichen Bezug zur Position des Aibo gebracht werden. Als räumlichen Bezug planen wir die Entfernung des Aibo zum Objekt zu verwenden. Wenn erkannt wurde, dass eine Kollision bevorsteht, benötigt der Aibo eine entsprechende Strategie, um dem Hindernis rechtzeitig auszuweichen und danach weiter der Pfadlinie zu folgen.

Als ersten Schritt zur Erkennung von Objekten im Kamerabild wenden wir den Kirsch-Kantendetektor aus dem WISP-Paket auf das Helligkeitsbild an. Zur besseren Bewertung der

Ergebnisse, zeigen wir dieses Bild in einem zusätzlichen WISP-Fenster an. Dieses Grauwert-Bild zeigt deutlich die Umrisse der verschiedenen Objekte im Bild. Zusätzlich liefert die Kirsch-Filter-Funktion von WISP auch noch ein Farbbild mit den Richtungen der gefundenen Kanten. Allerdings kam es zu Abstürzen von WISP bei dem Versuch dieses indizierte Farbbild anzuzeigen. Dabei sollte ein zusätzliches Objekt der Klasse AbPointer<AbWindow> in Kombination mit einem Zeiger auf ein Objekt der Klasse AbImageComponent verwendet werden. Dieser Mechanismus wird schon innerhalb des vorgegebenen Projekts verwendet und erschien uns daher plausibel. In einem WISP-Beispiel für den Kirsch-Kantendetektor werden Instanzen der Klasse Main::Window zur Visualisierung der Filterergebnisse verwendet. Mittels einer globalen Instanz von Main::Window konnten wir dann auch dieses Farbbild anzeigen. Hier sind die Kanten der Objekte deutlich erkennbar und die unterschiedlichen Kantentypen besitzen unterschiedliche Farben.

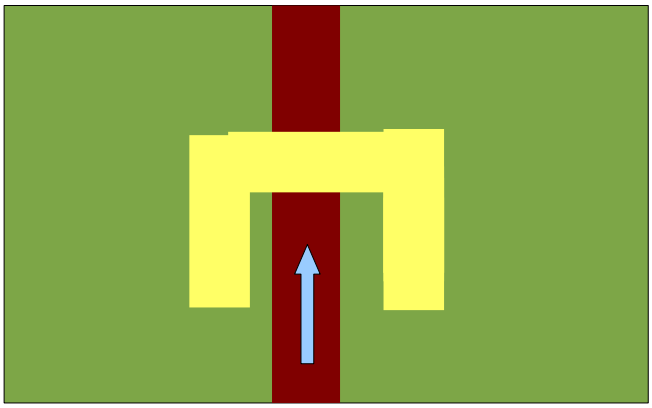
Dieses Bild speichern wir bei jedem Durchlauf, um im nächsten Durchlauf die Änderungen im Bildinhalt feststellen zu können.

Geplante Aktivitäten

Ausgehend von diesen beiden Bildern zum Zeitpunkt i und i-1, planen wir die Entfernung des Aibos zu einem Objekt zu bestimmen. Kommt ein Objekt näher, verändert sich seine Größe. Aus dieser Größenänderung lässt sich die Entfernung des Objekts bestimmen.

Deshalb besteht unser nächster Schritt darin, räumliche Objekte als solche in einem Bild zu erkennen. Danach müssen diese Objekte in einem anderen Bild wiedergefunden und verglichen werden. Dafür planen wir die Termine 3 und 4 ein. Parallel dazu werden wir unseren Abschnitt des Parcours mit Objekten versehen. Wir werden auch versuchen unseren Ansatz zur Verfolgung der Pfadlinie zu verbessern, um Abzweige zu finden. Sollte der Aibo in eine Sackgasse laufen, so muss er zum letzten Knoten zurückkehren und dem anderen Abzweig folgen.

Am 5. und 6. Praktikumstermin werden wir uns mit der Umgehung von einfach Hindernissen, wie z. B. Würfel, befassen. Dies beinhaltet auch das Wiederauffinden der Pfadlinie. Sollten wir danach noch Zeit zur Verfügung haben, werden wir versuchen, auch komplexere Hindernisse, wie z. B. in der nachfolgenden Abbildung zu sehen, zu umgehen.



Zeichnung 1: Komplexes Hindernis, Ansicht von oben