Echtzeitbildverarbeitung — Übungsblatt 4

www.informatik.uni-bremen.de/agebv/de/VeranstaltungEBV15 Abgabe: 29.06.2015

Allgemeine Hinweise

Benennt alle `.sample-cc` Dateien in `.cc` um. Die noch fehlenden Teile des Programms sind im Sourcecode mit `TODO` markiert, dahinter steht die jeweilige Punktzahl für den Teil. Implementiert diese Teile. Kopiert die bearbeiteten Teile des Codes in ein `.pdf` zusammen mit Euren Bearbeitungen der anderen Aufgaben. Dieses `.pdf` wird ausgedruckt, korrigiert und zurückgegeben.

Gebt dann die veränderten Sourcen zusammen mit der `.pdf`-Datei in einer `zip`-Datei bei Stud.IP ab. Die `.zip`-Datei soll den Namen `ebvassignment??-<namen>.zip` haben.

Gebt Namen und Email aller Gruppenmitglieder an. Gebt ausserdem die Zeit (max. aller Gruppenmitglieder) an, die Ihr für den Zettel benötigt habt. Diese Angabe ist nur als Rückmeldung für uns und geht nicht in die Bewertung ein.

Aufgabe 10 Ballspiele II (10 Punkte)

Diese Aufgabe, sowie Aufgaben 4 und 7 haben als Gesamtziel, die Flugbahn eines geworfenen Balles vorherzusagen. Das heisst, nach drei bis vier Bildern der Bildsequenz (Abb. 1) sagt das Programm die Flugbahn des Balles zutreffend vorher. Mit weiteren eintreffenden Bildern wird die Vorhersage verbessert. Die so berechnete Position könnte z.B. an einen Roboter kommandiert werden, der den Ball mit einem Netz auffängt oder mit einem Schläger zurückschlägt. Nachdem Ihr in Afg. 7 den Ball im Bild gefunden soll nun als letzter Schritt die eigentlich Flugbahnvorhersage mit einem Partikelfilter erfolgen. Wir gehen dabei von einer vorhandenen und im Programm fest kodierten Kamerakalibrierung aus.

Ihr benötigt dazu einen funktionierenden Kreiserkenner (`houghCircle.cc`) der auf einem funktionierenden Sobel-Filter aufbaut (`sobel.cc`).

Implementiert zuerst die Funktionen in `cameraCalibration.cc`. Sobald ihr `CameraCalibration::project`), also die Kameragleichung korrekt habt solltet Ihr an den Ecken des Schachbrettes grüne Kreuze sehen. Die anderen beiden Funktionen werden automatisch getestet, die entsprechende Fehlermeldung sollte verschwinden sobald sie richtig sind.

Implementiert dann den Partikelfilter `ParticleFilter` startend mit einer sinnvollen Initialisierung in `ParticleFilter::observeInit`. Dann sollten rote Kreuze auf dem Ball zu sehen

Abbildung 1: Aufeinanderfolgende Bilder (Abstand 1/25s) eines geworfenen Balles in einer natürlichen Szene. Dank geht an das Congress Center Bremen für die Drehgenehmigung. Der Ball ist ein Standard Volleyball.

sein.

Ergänzt die Initialisierung, so dass sie die Geschwindigkeit berechnet. Ein Problem ist die Initialisierung der Partikel von den ersten beiden Messungen. Dies würde zu einer hohen Unsicherheit in der Geschwindigkeit führen und daher enorm viele Partikel benötigen um die Verteilung so fein zu repräsentieren, dass der Filter funktioniert. Dem könnt Ihr entgegenwirken, indem ihr nicht aufeinanderfolgende Bilder, sondern zwei Bilder in einem gewissen Abstand (`waitUntilSecondObservation`) verwendet um die Partikel zu initialisieren.

Implementiert nun das Dynamikmodell (`ParticleFilter::dynamic`). Hier reicht eine einfache Euler-Integration eines Modells mit Gravitation ohne Luftwiderstand. Einen Bonuspunkt gibt es für das Verwenden einer analytisch korrekten Lösung dieser Differentialgleichung. Nun sollte eine Schar blauer Kurven vom Ball ausgehen und sich mit dem Ball mitbewegen.

Ergänzt nun das eigentliche Messmodell in (`ParticleFilter::Particle::observeRegular`. Die Schar der blauen Kurven sollte mit der Zeit enger werden und immer genauer am späteren Auftreffpunkt enden.

Läuft der Tracker insgesamt in Echtzeit?

Ziel der Aufgabe ist, die oben genannten Methoden selbst zu implementieren, d.h. Ihr dürft nicht die in OpenCV vorhandenen Funktionen verwenden. Das Program benötigt eine Lösung zu Aufgabe 4 und 7. Idealerweise könnt Ihr Eure eigene verwenden, indem Ihr den Rahmencode in Euer Verzeichnis kopiert. Ansonsten müsst Ihr die Musterlösung hineinkopieren. Das Programm wird gestartet mit `ballgame3 images/ball.???.jpg`.

Aufgabe 11 Freistoss (4 Punkte)

Bei einem Freistoss darf sich kein Gegenspieler innerhalb 9.15m Entfernung vom Ball aufhalten, bis der Ball gespielt worden ist. Für die Fernsehübertragung eines Fussballspieles soll mit Hilfe von Bildverarbeitung der verbotene Kreis eingezeichnet werden. Die Kamera ist auf einem schwenk-/nickbaren Stativ montiert und kann zoomen, es sind aber keine Sensoren an der Kamera angebracht, die Schwenken, Nicken oder Zoomen messen könnten.

Beschreibt einen kurzen Lösungsansatz (1-2 Seiten). Erklärt, welche Merkmale im Bild erkannt werden sollen und wie daraus der einzublendende Kreis berechnet wird. Macht es Sinn, die Aufgabe vollautomatisch zu lösen oder sollte manuell eingegriffen werden. Wie viele Merkmale werden im Bild benötigt, damit der Ansatz funktioniert?

Aufgabe 12 Der einäugige Ballfangkönig (1 Bonuspunkt)

Wir hatten diskutiert, dass eine Kamera ein Winkelmessgerät ist und damit Längenangaben über die Szene nur ermittelt werden können, wenn mindestens eine Länge in der Szene bekannt ist (sogenannter Skalenmassstab). Für unsere Ballflugbahnvorhersage ist dies der Balldurchmesser. Was wäre, wenn wir den Durchmesser des Balles nicht wüssten?

Gesucht ist eine Analyse der Situation, also ob wir den Skalenmassstab aus den Messungen ohne Balldurchmesser ermitteln können, keine technische Lösung.