Bildbasierte Modellierung SS 2018
Übungsblatt 8

TU Braunschweig
Prof. Dr.-Ing. Marcus Magnor
Institut für Computergraphik

JP Tauscher
tauscher@cg.cs.tu-bs.de

13.6.2018

Abgabe: Präsentation der bearbeiteten Aufgaben in der Übung am 19.6.2018.

Für die Programmieraufgaben kann in Gruppen von max. 3 Leuten zusammengearbeitet werden. Dabei muss aber jeder einzelne in der Lage sein, alle Teile des Programms zu erklären. Die Materialien für die Programmieraufgaben sind jeweils erhältlich unter:

`https://graphics.tu-bs.de/teaching/ss17/bbm`

In dieser Aufgabe soll ein Algorithmus implementiert werden, um aus Videodaten eines einfachen Laserscanners 3D-Punktwolken zu erzeugen. Der Scanner besteht aus einem Linienlaser, wie sie beispielsweise an Scannerkassen Verwendung finden, einer Kamera und zwei Hintergrundebenen. Der Laser überstreicht das Objekt, das vor den in einem Winkel zueinander stehenden Hintergrundebenen platziert ist. Objekt und Ebenen werden dabei von der Kamera aufgenommen. Die Geometrie der Ebenen wird als bekannt vorausgesetzt; sie kann beispielsweise durch ein optisches Kalibrierverfahren bestimmt werden. Das Licht des Lasers bildet zu jedem Zeitpunkt eine weitere Ebene, die die beiden Hintergrundebenen in jeweils einer Linie schneidet. In Verbindung mit der bekannten Geometrie der Ebenen kann aus diesen Linien die durch den Laser aufgespannte Ebene im Koordinatensystem der Kamera bestimmt werden. Die Koordinaten der Punkte, an denen der Laser das Objekt trifft, können auf diese Ebene projiziert werden, um 3D-Koordinaten zu erhalten. Wird nun der Laser über das Objekt bewegt, kann aus den Einzelbildern jeweils ein Linienprofil des Objekts rekonstruiert werden; zusammen ergeben diese Profile eine 3D-Punktwolke des Objekts.
Ein Laserscanner, der nach diesem Prinzip funktioniert, wurde am Institut für Robotik und Prozessinformatik der TU Braunschweig entwickelt. Aufbau und Funktionsweise sind im dem Paper *Low-Cost Laser Range Scanner and Fast Surface Registration Approach* erklärt.
Unser Programm zur 3D-Rekonstruktion erhält als ersten Parameter den Namen des Eingabevideos, als zweiten den der Ausgabedatei (jede Zeile ein 3D-Punkt, Koordinaten durch Komma getrennt, Endung `.asc`). Als dritter Parameter kann optional eine Ganzzahl angegeben werden, die bestimmt, jeder wievielte Frame des Videos rekonstruiert wird; dadurch lässt sich die Laufzeit beliebig auf Kosten der Auflösung reduzieren. Die entstehenden Punktwolken können mit Meshlab gerendert werden.

8.1 Vorüberlegungen (10 Punkte)

Um für jeden Punkt eine Tiefe berechnen zu können, muss aus den beiden Linien, an denen die Laserebene die Hintergrundebenen schneidet, die Laserebene im Kamerakoordinatensystem rekonstruiert werden. Vereinfachend können wir annehmen, dass die Kamera sehr weit vom Objekt entfernt ist, die Projektion ist also näherungsweise orthographisch. Anstelle einer automatischen Kalibrierung der Hintergrundebenen soll angenommen werden, dass diese senkrecht sowie symmetrisch um die horizontale Bildmitte angeordnet sind und sich dort orthogonal treffen.

- Skizziere eine Draufsicht der beiden Hintergrundebenen.
- Beschreibe z als Funktion von x jeweils für die linke und rechte Bildhälfte. Diese Funktion wird später verwendet, um 3D-Koordinaten aus den 2D-Bildkoordinaten zu berechnen.

8.2 Kalibrierlinien (5 Punkte)

Zunächst müssen im Eingabebild die beiden Kalibrierlinien gefunden werden. Dafür wird das Bild binarisiert und in den linken und rechten Teil separiert. Finde in einem solchen Binärbild die dominanteste Linie und gib zwei Punkte zurück, die auf dieser Linie liegen.

8.3 Laserscanner (15 Punkte)

Führe folgende Schritte für jeden Frame des Videos durch:

- Binarisiere das Bild, so dass nur die Laserlinie erhalten bleibt.
- Finde die Kalibrierlinie im linken Bild. Berechne die z-Koordinaten gemäß deiner theoretischen Überlegungen.
- Finde die Kalibrierlinie im rechten Bild. Berechne die z-Koordinaten gemäß deiner theoretischen Überlegungen. Beachte dabei, dass die von `find_line` zurückgegebenen Koordinaten relativ zum Ursprung der rechten Bildhälfte sind.
- Finde eine Ebene, die durch beide Kalibrierlinien geht. Für jeden Punkt $\vec{x}$ auf der Ebene soll gelten: $\vec{n} \cdot \vec{x} = d$.
- Projiziere jeden hellen Punkt des Binärbildes auf die Ebene und schreibe die 3D-Koordinaten in der Form x, y, z nach `outfile`. Da im OpenCV-Koordinatensystem y nach unten hin wächst, solltest du die y-Koordinate invertieren, um die korrekte Darstellung im Renderer zu erreichen. Abhängig von der Wahl deines Koordinatensystems musst du möglicherweise auch z invertieren.
- Stelle dein Ergebnis mit Meshlab dar. Nutze nicht die Grid Triangulation und verwende einen geeigneten Shader. Entspricht das Ergebnis deinen Erwartungen? Warum?