Aufgaben zu Stereoskopie

Lösen Sie die nachfolgenden Aufgaben und bereiten Sie diese bis zum nächsten Lehrveranstaltungstermin vor. Unterstrichene Aufgaben sind nach Möglichkeit während der Lehrveranstaltung zu lösen.

LB-S 01.

a) Schreiben Sie ein Programm, dem ein linkes und ein rechtes Bild (vereinfacht als bereits rektifiziert angenommen) als Argumente übergeben werden, das aus beiden mit dem semiglobalen Stereo-Matcher (Klasse `cv::StereoSGBM` mit deren Methode `compute`) ein Disparitätsbild erzeugt. Verwenden Sie die in der *OpenCV*-Dokumentation empfohlenen Parameterwerte (bzw. einen der Werte aus den empfohlenen Bereichen) und insgesamt 64 verschiedene Disparitäten. Geben Sie das Disparitätsbild abschließend in einem Fenster aus. Achten Sie dabei auf dessen korrekte Skalierung, d.h. konvertieren Sie das Tiefenbild mittels `cv::Mat::convertTo` in ein Graustufenbild, das 8 Bit Farbtiefe voll ausschöpft.

Benötigte Header: `opencv2/calib3d.hpp`, `opencv2/highgui.hpp`

b) Passen Sie Ihr Programm aus a) über die Parameter des Stereo-Matchers derart an, dass das erzeugte Disparitätsbild Ihrer Meinung nach besser wird. Wenden Sie Ihr Programm außerdem, vorbereitend auf das Folgebeispiel, auf die zur Verfügung gestellten, nicht rektifizierten Testbilder an.

LB-S 02.

a) Schreiben Sie ein Programm, dem Pfade zu zwei *yml*-Dateien (verwenden Sie die zur Verfügung gestellten Beispieldateien) mit intrinsischen und extrinsischen Kameraparametern (in dieser Reihenfolge) als Argumente übergeben werden. Das Programm soll die in den Dateien gespeicherten Matrizen mit Hilfe der `cv::FileStorage`-Klasse auslesen, in einer geeigneten Datenstruktur speichern und anschließend ausgeben. Achten Sie hierbei auf die Wiederverwendbarkeit Ihrer Funktionen und Datenstrukturen.

b) Erweitern Sie Ihr Programm aus LB-S 01. b) derart, dass es zwei zusätzliche Argumente entgegennimmt, die Pfade zu *yml*-Dateien mit den intrinsischen und extrinsischen Parametern der verwendeten Kameras (in dieser Reihenfolge) darstellen. Lesen Sie diese mit Hilfe der in a) implementierten Funktionen aus und verwenden Sie sie anschließend zur Rektifizierung der beiden Kamerabilder, die an Stelle der originalen Kamerabilder zur Disparitätsbilderzeugung verwendet werden. Nutzen Sie zur Rektifizierung die Funktionen `cv::stereoRectify`, `cv::initUndistortRectifyMap` und `cv::remap` mit Standardparametern und zeigen Sie zur Kontrolle die rektifizierten Bilder an.

Zusätzlich benötigte Header: `opencv2/imgproc.hpp`

c) Modifizieren Sie Ihr Programm aus b) derart, dass es keine Bilder mehr anzeigt, aber zusätzlich eine 3-D-Rekonstruktion berechnet. Verwenden Sie dazu die Funktion `cv::reprojectImageTo3D`, um das linke Stereobild mit Hilfe des **un**skalierten Disparitätsbildes und der Kameraparameter in entsprechend eingefärbte Punkte mit 3-D-Koordinaten umzurechnen. Geben Sie die 3-D-Koordinaten dreier Punkte, die gültige Werte besitzen, testweise auf der Konsole aus.

LB-S 03.

a) Erweitern Sie Ihr Programm aus LB-S 02. c) derart, dass es keine Punkte auf der Konsole ausgibt, sondern diese visualisiert. Erstellen Sie dazu eine Instanz der Klasse `cv::viz::Viz3d` und verwenden Sie deren Methoden `spin` und `showWidget`. Übergeben Sie letzterer eine passend initialisierte Instanz der Klasse `cv::viz::WCloud`. Setzen Sie zuvor alle Punkte, deren z-Koordinate größer als 10 ist, auf eine Instanz der `cv::Vec3f`-Klasse, deren Komponenten den Wert `std::numeric_limits<float>::quiet_NaN()` haben, um die Visualisierung der dazugehörigen Punkte zu verhindern.
Hinweis: Das bei der 3-D-Visualisierung dargestellte Fenster lässt sich mit der Taste Q beenden. Mit dem Mausrad kann gezoomt werden; mit Mausbewegungen bei gleichzeitig gedrückter linker Maustaste kann die Ansicht gedreht werden.

Zusätzlich benötigte Header: `opencv2/viz.hpp`

b) Modifizieren Sie Ihr Programm aus a) derart, dass es eine andere, Ihrer Meinung nach besser geeignete z-Koordinaten-Filtergrenze verwendet.

c) Modifizieren Sie Ihr Programm aus b) derart, dass es die Punkte standardmäßig in der richtigen Orientierung anzeigt, d.h. so, dass die Bildrepräsentation aufrecht anstatt auf dem Kopf steht. Setzen Sie dazu mittels `cv::viz::Viz3d::setViewerPose` und `cv::viz::Viz3d::getViewerPose` eine passende Kameratransformationsmatrix.