Ausgabe am **10.05.2021**
Abgabe bis zum **23.05.2021 23:59 Uhr**

UNIVERSITÄT OSNABRÜCK

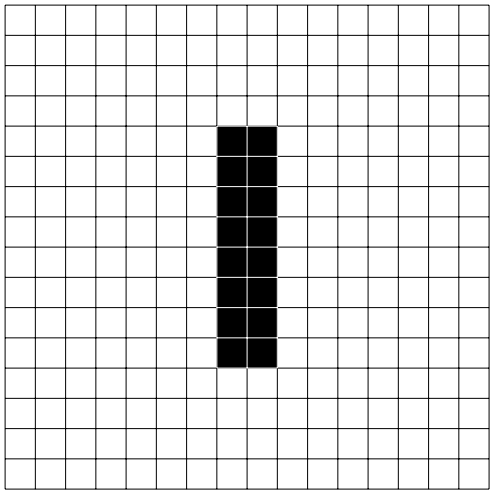
Priv. Doz. Dr. Thomas Wiemann, Alexander Mock

Robotik
Übungsblatt 3

Sommersemester 2021

Aufgabe 3.1 (Bildfaltung von Hand) **(3P)**

Gegeben sei das folgende Grauwertbild (weiß: 255, schwarz: 0) der Größe 16×16 Pixel:



(a) Schreiben Sie die Kernel auf, mit denen 1. horizontale, 2. vertikale Linien im Bild erkannt werden können.

(b) Führen Sie mit Kernel 1 und 2 eine Bildfaltung durch. *Hinweis*: Sie müssen das Bild nicht zeichnen, eine Angabe der Grauwerte im Diagramm reicht aus.

(c) Geben Sie einen passenden Kernel der Größe 5×5 für eine Gauss-Filterung an. Diskretisieren Sie hierzu eine Gaussverteilung mit $\sigma = \sqrt{1}$. Wie würde sich sich ein Kernel mit steigendem Sigma auf das Bild auswirken?

(d) Falten Sie das obige Bild mit dem Gauss-Kernel.

(e) Aus welchem Grund wird ein solcher *Gauß Filter* (bzw. seine Approximation durch einen *Center-Surround Filter*) oft als erster Schritt in einer Pipline von weiteren Filtern verwendet?

(f) Wie würden Sie vorgehen, wenn Sie einen Filter nicht auf ein Graustufenbild mit 1 Kanal sondern auf auf 3 Kanal (RGB) Farbbild anwenden möchten?

Jeweils 0.5P

Aufgabe 3.2 (Bildfaltung in ROS) **(5P)**

In dieser Aufgabe werden Sie mehrere Bildfaltungen auf einem Bild durchführen. Als Bilddaten können Sie hierfür die Bagfile aus StudIP oder sogar Ihre eigene Webcam benutzen: `http://wiki.ros.org/usb_cam`.

(a) Erstellen Sie zunächst eine Node namens `convolve_image`, die genau ein Bild falten soll. Diese Node soll sich auf ein Topic vom Typ `sensor_msgs/Image` subscriben und im Callback das ankommende Bild mit Hilfe einer Kernel-Matrix falten. Diese Kernel-Matrix soll über einen privaten ROS-Parameter namens `kernel` angegeben werden können. Publishen Sie anschließend das Ergebnis auf ein weiteres Topic vom Typ `sensor_msgs/Image`. Testen Sie Ihre Node mit Hilfe einer Launch-File namens `convolve_image_1.launch`, in der Sie eine Instanz ihrer Node starten und eine Kernel-Matrix Ihrer wahl als `kernel`-Paramter setzen. Remappen Sie auch Ihre frei gewählten Topic-Namen so, dass Sie sich auf das existierende Bild-Topic Ihrer Test-Bagfile subscriben können. (3P)

Hinweis: Sie dürfen für die Bildfaltungsoperation auch die Funktionalitäten von OpenCV benutzen.

(b) Schreiben Sie nun eine Launch-File namens `convolve_image_2.launch`, welche ein Bild über verschiedene Instanzen Ihrer Node mit unterschiedlicher Kernel-Matrix faltet. Die erste Instanz soll das Original-Bild lesen und einen Gauss-Kernel anwenden. Die zweite Instanz soll das Ergebnis lesen und einen Laplace-Kernel anwenden. Visualisieren Sie die Ergebnisse mit RViz. (2P)

Aufgabe 3.3 (Theorieaufgabe: SIFT) **(3P)**

In der Vorlesung wurde der SIFT-Algorithmus zum Auffinden von stabilen Merkmalen in Bildern vorgestellt.

(a) Welche Pixel werden als Keypoints ausgewählt und wie bestimmt man deren Orientierung?

(b) Beschreiben Sie den Aufbau eines SIFT-Diskriptors und wie dieser berechnet wird.

(c) Beschreiben Sie kurz, wie Sie damit ähnliche Objekte in zwei gegebenen Bildern auffinden können.

(d) Was passiert wenn zwei unterschiedlich rotierte Bilder miteinander verglichen werden?

(e) Funktioniert SIFT immernoch, wenn Bilder miteinander verglichen werden, die aus unterschiedlichen Perspektiven aufgenommen wurden?

Abbildung 1: Arc de Triomphe aus zwei unterschiedlichen Perspektiven.

Aufgabe 3.4 (Differenzialantrieb) **(4P)**

Gegeben sei ein Roboter mit Differenzialantrieb an Pose $(0\,\text{cm}, 0\,\text{cm}, 0°)$ zum Zeitpunkt $t = 0$.

Die Kinematik für einen Differenzialantrieb sei wie folgt gegeben:

$$\Delta d = \frac{(s_l + s_r)}{2}$$
$$\Delta\theta = \frac{(s_l - s_r)}{b}$$

Δd entspricht dabei der gefahrenden Strecke des Roboters, $\Delta\theta$ der Winkeländerung. Die Räder der linken/rechten Seite haben einen Abstand von $b = 30\,\text{cm}$.

(a) Zur Zeit $t = 1$ wird gemessen, dass sich das rechte Rad $s_r = 6\,\text{cm}$ bewegt hat, das linke $s_l = 10\,\text{cm}$. An welcher Pose befindet sich der Roboter nun? Für $t = 2$ liefert die Messung: Links 5 cm, rechts 7 cm. Wo steht der Roboter nun? (3P)

(b) Sie stellen fest, dass der Roboter nicht genau arbeitet. Bei jeder Rotation und Translation ist die resultierende Pose geringfügig ungenau. Argumentieren Sie welcher Fehler (Rotation oder Translation) sich langfristig stärker auswirkt. (1P)

Aufgabe 3.5 (Gazebo Simulation) **(5P)**

Bislang haben Sie Bagfiles benutzt, um Algorithmen auf aufgenommen Sensordaten zu entwickeln und zu testen. Alternativ stellt ROS die Simulationsumgebung *Gazebo*, welche einen Roboter und seine Umgebung simulieren kann.

(a) Um die Simulation starten zu können, benötigen Sie zunächst ein Robotermodell, eine modellierte Welt und weitere nützliche Tools. Klonen Sie sich alle Pakete von der folgenden Website in Ihren `src`-Ordner Ihres Workspaces: `https://gitlab.informatik.uni-osnabrueck.de/robotik_ss21_packages`.

- `epos2_motor_controller` + `volksbot_driver` + `ceres_robot`: Robotermodell Ceres
- `uos_tools`: Modellierte Welten und weitere nützliche Tools (z.B. Tastatursteuerung).

Für das Roboter-Modell werden Sie noch die FTDI-Bibliothek benötigen:

```
$ sudo apt-get install libftdipp1-dev
```

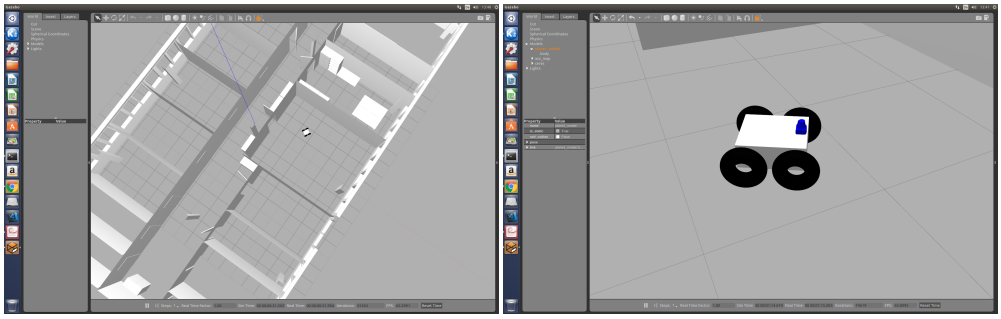
Weitere fehlende ROS-Abhängigkeiten können Sie über die allgemeinen Paketquellen installieren (oder mit `rosdep`). Zum Beispiel benötigen Sie zur Simulation des Laserscanners das Paket `sick_tim`. Dieses kann wie folgt installiert werden:

```
$ sudo apt-get install ros-noetic-sick-tim
```

Kompilieren Sie danach Ihren Workspace. Führen Sie folgenden Befehl aus, um die Simulation zu starten:

```
$ roslaunch ceres_gazebo ceres.launch
```

Es öffnet sich eine Oberfläche, in der Sie folgendes sehen sollten:



Machen Sie sich mit der Benutzeroberfläche vertraut. Verschieben und rotieren Sie den Roboter. Platzieren Sie eine weiteres Objekt, z.B. einen Würfel. Geben Sie zu dieser Aufgabe auch einen geeigneten Screenshot ab. (1P)

(b) Bewegen Sie nun den Ceres-Roboter in der Simulation. Nutzen Sie dazu die Tastaturfernsteuerung aus den `uos_tools`. Diese kann mit

```
$ roslaunch uos_diffdrive_teleop key.launch
```

gestartet werden. Achten Sie bei der Steuerung darauf, dass ihr Konsolenfenster im Vordergrund bleibt und beobachten Sie, wie sich der LaserScan in RViz verhält. Wenn Sie A,W,S oder D drücken, publisht die gestartet Node eine Message vom Typ `geometry_msgs/Twist` auf das Topic `cmd_vel`. Der (simulierte) Roboter nimmt diese Message entgegen und führt daraufhin eine Bewegung aus. (0P)

(c) Gazebo erzeugt simulierte Sensordaten für den Roboter. Geben Sie sich die erzeugten Topics auf der Kommandozeile aus. Nutzen Sie auch RViz, um diese zu visualisieren. Was fällt Ihnen bei dem Laserscan im Gegensatz zu den bisherigen Bagfiles auf? Lassen Sie sich in RViz auch das Robotermodell anzeigen.

Beschreiben Sie eine Situation, in der eine Bagfile nicht mehr ausreicht und stattdessen auf eine Simulation zurückgegriffen werden muss. Geben Sie dazu auch ein konkretes Beispiel an. (1P)

(d) Schreiben Sie nun eine Node, welche die Roboteransteuerung vollautonom übernimmt. Nutzen Sie den LaserScan, um Hindernissen auszuweichen. Denken Sie sich eine geeignete Stategie aus, um aus den LaserScan-Daten auf eine geeignete Ansteuerung zu schließen. Die Ansteuerung des Roboters geschieht genau wie bei der Tastaturfernsteuerung über das Topic `cmd_vel` vom Typ `geometry_msgs/Twist`. Und keine Angst: In der Simulation kostet es nichts, wenn Sie einen Unfall verursachen. (3P)