Artomat Simulation

Nils Egger

13.08.2019

1 Grundidee

Als wir auf die Idee kamen einen Roboter zu bauen und dazu eine Steuerungssoftware zu schreiben stellte ich mir die Frage wie wir dies am besten Parallel machen. Meiner Meinung nach kann man Software besser entwickeln wenn man sie auch gleich ausprobieren kann. In unserem Fall wäre dies die Kamera auf den aufgehängten Roboter mit den Erkennungspunkten zu richten. Mit diesen Gedanken im Kopf kam ich auf die Idee eine Simulation zu schreiben während David an unserem Roboter arbeitet.

2 Bibliotheken

Für die Simulation gebrauche ich Tkinter für die Darstellung. Im Nachhinein war dies vermutlich ein fehlentschluss, dies nicht weil die Bibliothek die Simulation schlecht darstellt, sondern weil sie die Darstellung nicht einfach in eine Pixel Matrix kopieren kann. Müsste ich nochmals eine auswahl treffen, so würde diese bei OpenCV liegen, denn mit OpenCV hätte ich nur kleine Performance Verluste beim Transport des Bildes zum Navigationsalgorithmus gehabt. Dies wusste ich im voraus nicht da ich weder mit OpenCV oder Tkinter bekannt war, in Zukunft wird dies aber ein No-Brainer. Damit ich eine Kamera auf die Simulation simulieren konnte, musste ich den Tkinter Canvas in das einzig Unterstüzte Format exportieren, nähmlich Postscript ein Vektor Format. Postscript wird jedoch natürlich nicht von OpenCV unterstützt und anstatt eine Kamera einfach auf den Bildschirm zu richten, musste ich eine weitere Bibliothek benutzen. PIL (Python Imaging Library) kam hierbei zur Lösung, diese Bibliothek kann Postscript in ein normales Bild

verwandeln welches von Numpy (Matrix Bibliothek, welche von OpenCV für die Pixel Matrix von Bildern benutzt wird.) dann als Multidimensionaler Array eingelesen wird und von OpenCV bereit ist. Dies entspricht folgendem Code:

```
numpy.array(Image.open(io.BytesIO(self.ps_frame.encode('utf-8'))))
```

3 Aufbau

Die Simulation besteht aus drei Klassen:

1. **Window**
   Die Window Klasse ist unter anderem dafür zuständig einen Canvas für die Simulation bereitzustellen und fängt zudem noch Tastatur Schläge ab und steuert mit diesen die Motoren der Simulation an.

2. Simulation

3. Object