

ROME1 - Praktikum 4

Steuern des Roboters mit Python

Das Ziel dieses Praktikums ist es, einfache Roboterprogramme mit Bewegungssequenzen direkt in Python auf einem externen PC zu erstellen. In diesen Roboterprogrammen stehen dann nicht nur die speziellen Funktionen der Robotics Toolbox for Python zur Verfügung, sondern auch alle anderen Funktionen von Python und dessen zahlreichen Bibliotheken.

Damit Roboterprogramme in Python erstellt werden können, muss ein Kommunikationsprotokoll zwischen der Robotersteuerung und einem externen PC erstellt respektive ergänzt werden.

1 TCP/IP Kommunikation für einfache Fahrbefehle

1.1 Kommunikationsprotokoll

Für die Kommunikation zwischen dem PC, auf dem Python läuft und der ABB Robotersteuerung OmniCore wird das Ethernet Protokoll TCP/IP eingesetzt. Dieses Protokoll ermöglicht, dass zwei über Ethernet verbundene Systeme beliebige Zeichenketten senden und empfangen können.

Um nun sinnvoll Daten und Instruktionen auszutauschen muss noch ein Anwendungsprotokoll definiert werden. Für dieses Praktikum wird ein einfaches Client-Server Protokoll verwendet, bei dem der PC, auf dem Python läuft als Client funktioniert und Zeichenketten an die Robotersteuerung sendet. Die Robotersteuerung dient als Server, der diese Zeichenketten verarbeitet und eine Antwort zurücksendet.

Das einfache Protokoll, das schon im Praktikum 3 eingesetzt wurde, besteht aus Instruktionen und optionalen numerischen Parametern, die als lesbare Zeichenketten zusammengestellt werden. Abgeschlossen werden diese Zeichenketten mit Carriage Return (CR) und Line Feed (LF) Zeichen (resp. New Line). Die Zeichenketten, die schon im Praktikum 3 verwendet wurden, lauten:

```
"Home\r\n"  
"MoveAbsJ 30 30 -20 45 90 -45\r\n"  
"Close\r\n"
```

Die Robotersteuerung sendet nach dem Empfang einer Zeichenkette resp. nach dem Ausführen einer Instruktion eine Antwort zurück. Diese hat die folgende Form:

```
"OK\r\n"
```

Diese Antwort ist eine Bestätigung, dass die entsprechende Instruktion erfolgreich ausgeführt wurde.

1.2 RAPID Programm mit TCP/IP Server

Damit eine gesendete Zeichenkette von der Robotersteuerung verarbeitet werden kann, muss auf der Robotersteuerung ein TCP/IP Server gestartet werden, welcher auf Verbindungen mit einem Client wartet. Wenn eine Verbindung aufgebaut ist, müssen die empfangenen Zeichenketten interpretiert und die gewünschte RAPID Instruktion aufgerufen werden.

Im Verzeichnis dieses Praktikums liegt ein RAPID Programm namens 'MainModule.mod', welches einen TCP/IP Server implementiert. Laden Sie dieses Programm auf die Robotersteuerung, indem Sie

es im RobotStudio ins Fenster mit dem aktiven Programm der Steuerung kopieren und anschliessend übernehmen.

Wechseln Sie den Betriebsmodus der Robotersteuerung zu **Auto** und starten Sie dieses Programm. Die Robotersteuerung wartet nun auf einen Client, der eine TCP/IP Verbindung aufbauen und Instruktionen senden möchte.

1.3 Python Klasse mit TCP/IP Client

Für Python steht eine Klasse namens 'OmniCore.py' im Verzeichnis dieses Praktikums zur Verfügung. Diese Klasse implementiert verschiedene Funktionen, welche eine TCP/IP Verbindung zum Server aufbauen, die Zeichenketten gemäss dem beschriebenen Protokoll von Abschnitt 1.1 zusammenstellen und via TCP/IP Verbindung an die Robotersteuerung senden.

Die meisten Funktionen, wie zum Beispiel die Funktion **moveAbsJ** warten nach dem Senden des Protokolls auf die Antwort vom Server. Diese Funktionen blockieren damit den Programmablauf, bis die entsprechende Instruktion von der Robotersteuerung auch ausgeführt wurde.

1.4 Einfache Bewegungen mit Python ausführen

Damit nun der Roboter mit Python Kommandos gesteuert werden kann, muss von der Klasse 'OmniCore' ein Objekt erstellt werden, welches die verschiedenen Funktionen anbietet. Mit dem Kreieren des Objektes muss noch die IP-Adresse der Robotersteuerung angegeben werden, mit der via Ethernet kommuniziert werden soll. Wählen Sie dazu die IP-Adresse des Service-Ports der Steuerung wie im folgenden Beispiel:

```
>>> from OmniCore import OmniCore
>>> robot = OmniCore("192.168.125.1")
```

Damit wird eine TCP/IP Verbindung mit der Robotersteuerung aufgebaut, über die verschiedene Kommandos an den Roboter gesendet werden können. Um nun den Roboter in eine gewünschte Lage zu bewegen kann die folgende Funktion der 'OmniCore' Klasse aufgerufen werden:

```
>>> robot.moveAbsJ([30, 30, -20, 45, 90, -45])
```

Dieser Funktion muss ein Array mit den 6 gewünschten Gelenkwinkeln übergeben werden. Diese Gelenkwinkel werden in der Einheit Grad übergeben.

Der Roboter kann mit der folgenden Funktion auch in seine Null-Lage bewegt werden, also in die Lage, in der alle Gelenkwinkel 0 sind.

```
>>> robot.moveAbsJ([0, 0, 0, 0, 0, 0])
```

Um den Roboter wieder in seine Parkposition zu bewegen, kann die Funktion **home** der 'OmniCore' Klasse aufgerufen werden.

```
>>> robot.home()
```

Die Kommunikation kann wieder abgebrochen werden, indem das Objekt `robot` gelöscht wird.

```
>>> del robot
```

Dabei wird eine `delete` Funktion der 'OmniCore' Klasse aufgerufen, welche der Robotersteuerung zuerst mitteilt, dass die TCP/IP Verbindung geschlossen werden soll, und die Verbindung auf der Seite des PCs dann auch schliesst.

2 Erweiterung mit MoveL Kommando

Das bestehende RAPID Programm und die zugehörige Python Klasse ermöglichen, den Roboter in eine Lage zu bewegen, die mit Gelenkwinkeln gegeben ist. Die Bewegung des Roboters in diese Lage erfolgt durch Interpolation der Gelenkwinkel.

Das RAPID Programm und die Python Klasse sollen nun so erweitert werden, dass auch lineare Bewegungen in Kartesischen Koordinaten möglich werden.

2.1 Erweiterung des RAPID Programmes

Das RAPID Programm soll zusätzlich Fahrbefehle empfangen und ausführen können, welche den Roboter mit dem RAPID Befehl `MoveL linear` an eine Kartesische Position bewegen. Dazu muss die Funktion `processCommand` des Roboterprogramms erweitert werden, so dass sie das Kommando `MoveL` versteht.

Die folgenden Parameter sollen mit dem `MoveL` Kommando übergeben werden können:

- Kartesische Position in x, y und z
- Orientierung in Quaternionen (4 Werte)
- Konfiguration des Roboters (Quadranten von Gelenkwinkeln, 4 Werte)

Total werden somit 11 Parameter übergeben.

2.2 Erweiterung der Python Klasse

Damit die neue Funktion des RAPID Programms in Python genutzt werden kann, muss auch die `OmniCore` Klasse mit dieser neuen Funktion erweitert werden. Fügen Sie in der Klasse eine neue Funktion namens `moveL` ein. Diese Funktion soll im Unterschied zur Funktion `moveAbsJ` nicht ein Array von 6 Gelenkwinkeln als Parameter übernehmen, sondern 3 Arrays: ein Array mit 3 Werten der Kartesischen Position, ein weiteres Array mit 4 Parametern für die Orientierung (Quaternionen), sowie schliesslich ein Array mit 4 Parametern, welche die Konfiguration des Roboters beschreiben. Berücksichtigen Sie dabei, dass die numerischen Werte der einzelnen Parameter mit unterschiedlich vielen Nachkommastellen übermittelt werden sollen: Die Kartesischen Koordinaten sollen wie die Gelenkwinkel der `moveAbsJ` Funktion mit 3 Stellen abgebildet werden. Die Quaternionen hingegen benötigen 6 Nachkommastellen, so dass die Orientierung genau genug dargestellt werden kann. Und die Parameter der Konfiguration des Roboters können schliesslich als ganze Zahlen übermittelt werden, also ohne Nachkommastellen.

2.3 Bewegungen in Kartesischen Koordinaten

Testen Sie die neue Funktion `moveL` mit Bewegungen in Kartesischen Koordinaten. Wenn Sie nicht sicher sind, welche Parameter für die Position, die Orientierung und die Konfiguration gewählt werden sollen, können Sie die Angaben für die Lage in der folgenden Tabelle 2.1 verwenden, die als Beispiel dient.

Bewegen Sie dazu den Roboter zuerst mit der `moveAbsJ` Funktion aus einer beliebigen Lage, zum Beispiel aus der Parkposition in die Lage, wie sie in der Tabelle gegeben ist. Bewegen Sie dann den Roboter mit der `moveL` Funktion an verschiedene Kartesische Positionen, und verwenden Sie dabei dieselben Angaben für die Orientierung und die Konfiguration wie in der Tabelle angegeben.

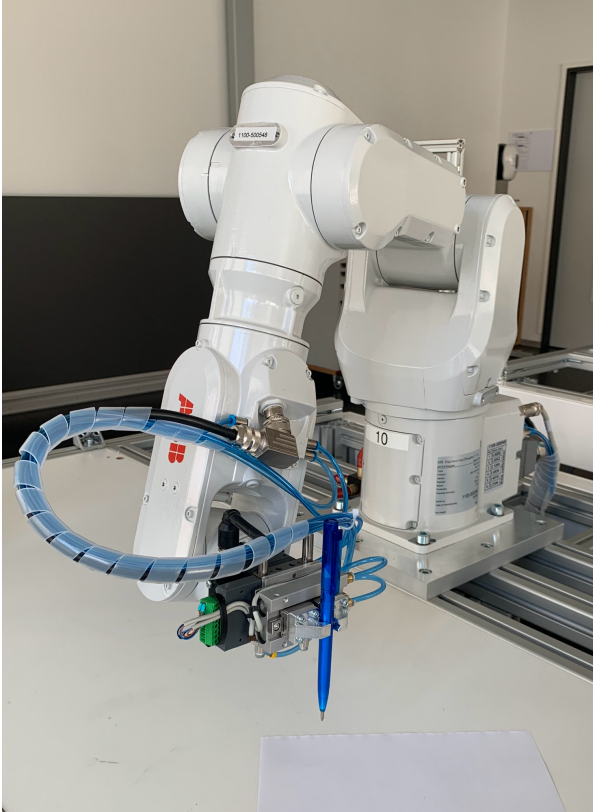
Lage	
Gelenkwinkel	[17, 79, -10, -74, -84, -72]
Position	[375, 280, 105]
Orientierung	[0.707107, -0.707107, 0.000000, 0.000000]
Konfiguration	[0, -1, -1, 1]

Tabelle 2.1: Beispiel für Gelenkwinkel sowie Parameter für Position und Orientierung

3 Plotten von Bildern

Entwickeln Sie nun ein Python Programm (z.B. ein Skript), welches ein Bild von einem File liest und dieses Bild mit dem Roboter auf ein Blatt Papier plottet.

3.1 Laden eines Bildes in Python

Das Bild sollte für diese Aufgabe als schwarz/weiss Bild (z.B. mit 8bit pro Pixel) in einem für Python kompatiblen Fileformat vorliegen, wie z.B. gif, jpeg oder png. Die Grösse des Bildes sollte zudem auf ca. 200 x 200 Pixel beschränkt sein.

Die folgenden Bilder sind mögliche Beispiele und liegen im Verzeichnis dieses Praktikums. Sie können für diese Aufgabe aber ein beliebiges Bild verwenden, welches die Kriterien oben erfüllt.



Bild 3.1: Bilder, die geplottet werden können: a) matterhorn.png b) rose.png

Ein Bild kann in Python zum Beispiel wie folgt geladen werden:

```
>>> import os
>>> import matplotlib.image as im
>>> os.chdir("/path/to/working/directory")
>>> image = im.imread("matterhorn.png")
```

Die Variable `image` ist dann eine 200 x 200 Matrix mit den Helligkeitswerten zwischen 0.0 und 1.0.

3.2 Plotten des Bildes mit dem Roboter

Spannen Sie einen Schreibstift in den Greifer des Roboters und bewegen Sie den Roboter in die Lage, die in Tabelle 2.1 vorgeschlagen wird.

Schreiben Sie nun ein Python Programm, welches aus der Matrix mit den Helligkeitswerten des Bildes Bewegungsbefehle für den Roboter generiert. Bewegen Sie den Stift dazu Pixel für Pixel und Zeile für Zeile über das Blatt Papier und verwenden Sie dazu die neue Funktion `moveL`, welche den Roboter in Kartesischen Koordinaten bewegt.

Die x-Koordinate für den Roboter kann dabei z.B. dem Index der Spalte der Bildmatrix in [mm] entsprechen. Wenn das Bild 200 x 200 Pixel gross ist, entsteht damit ein Bild mit 20 cm Seitenlänge. Ebenso kann die y-Komponente für den Roboter dem Index der Zeile der Bildmatrix entsprechen. Die z-Komponente für den Roboter sollte so gewählt werden, dass sich der Stift für kleine Helligkeitswerte an dieser Position (z.B. < 0.5) auf dem Blatt Papier befindet, für grössere Werte aber oberhalb des Papiers.