Robotic Games WS19/20

Zwischenbericht

Enrico Kaack, Robin Fleige, Laura Nell

November 25, 2019

Chapter 1

Implementierung

1.1 Collision avoidance

```
def calculate_force(self, sonar_angles, sonar_ranges):
  sum = np.zeros(2)

  for i, sonar_range in enumerate(sonar_ranges):
      if sonar_range == 0.0:
          rospy.logerr('Catched␣Zero')
          sonar_range = 1e-12
      vec = np.array([1/sonar_range * np.cos(sonar_angles[i]), 1/
   sonar_range * np.sin(sonar_angles[i])])
      sum += vec

  sum *= (-1)
  sum_dist = np.linalg.norm(sum)

  if sum_dist < 10.0:
      return np.zeros(2)

  phi = -np.arctan2(sum[1], sum[0])
  force = np.zeros(2)
  force[0] = np.clip(-sum[0] / 20, 0, 1)
  force[1] = np.clip(phi, -np.pi/2, np.pi/2)

```

```
  if -force[1] == self.last_angle:
      force[1] = self.last_angle

  self.last_angle = force[1]
  return force
```

1.2 Homing

```
rospy.Subscriber("dead_reckoning", Pose, self.callback)
```

```
def callback(self, pos):
  self.orientation[0] = pos.orientation.z
  self.position[0] = pos.position.x
  self.position[1] = pos.position.y
```

```
def closed_loop(position,orientation,target):
  output=Twist()

  target_rel_x = target[0] - position[0]
  target_rel_y = target[1] - position[1]

  dir_x = np.cos(orientation[0]-np.pi/2)*target_rel_x + np.sin(
   orientation[0]-np.pi/2)*target_rel_y
  dir_y = np.cos(orientation[0]-np.pi/2)*target_rel_y - np.sin(
   orientation[0]-np.pi/2)*target_rel_x

  if np.abs(np.arctan2(dir_x, dir_y)) < np.pi/4 and dir_x*dir_x+
   dir_y*dir_y > 0.01:
      output.linear.x = 0.5
  else:
      output.linear.x = 0
  if np.abs(np.arctan2(dir_x,dir_y)) > np.pi/30:
      output.angular.z = np.arctan2(dir_x, dir_y)/3

  return output
```

1.3 Free Space

Das Free Space Verhalten soll dafür sorgen, dass der Roboter in Richtung der freien Fläche im Raum fährt. Die zur Verfügung stehenden Informationen beschränken sich dabei auf die vorhandenen acht Ultraschall Sensoren und deren Entfernungsmessung.
Der Algorithmus prüft, ob die größte Entfernung links oder rechts der Hauptachse gemessen wurde und dreht den Roboter entsprechend nach rechts oder links mit einer konstanten Drehrate.
Bei diesem Ansatz werden an zwei Stellen Probleme erwartet. Zum einen wird der Sensor mit der maximalen Entfernung zur Bestimmung der Drehrichtung benutzt. Wenn dieser Sensor allerdings einen Gang misst, der vom Roboter weg verläuft, würde sich der Roboter in diese Richtung orientieren. Dieses Verhalten wäre dementsprechend falsch, da der Gang nicht der freien Fläche entspricht. Um dem entgegen zu wirken, könnte man die Sensorwerte von jeweils der rechten und linken Seite mitteln und darauf basierend die Richtungsentscheidung treffen. Im späteren Schritt der Fusion wird der Free Space Teil allerdings nur eine untergeordnete Rolle spielen, sodass dieses Problem nicht in dieser Form auftritt.
Eine zweite Schwäche liegt darin, dass eine konstante Drehrate in die jeweilige Richtung gesetzt wird. Eine Auswirkung hiervon wäre, dass bei zu hoher Geschwindigkeit der Roboter gegen ein Hindernis fahren könnte, da die konstante Drehrate nicht auf die Lineargeschwindigkeit angepasst ist und der Kurvenradius demnach größer wird. Eine Lineargeschwindigkeit abhängig von der Entfernung des Sensors mit der niedrigsten Entfernung würde dies beheben. Sollte in der Fusion dies zu einem Problem führen, wird diese Idee entsprechend aufgegriffen. Da allerdings die Kollisionsvermeidung eine höhere Gewichtung in der Nähe einer Wand erhalten wird, sollte dies nicht zu einem Problem führen.

Verhaltensfusion