Sprawozdanie nr 3. „Robot omijający przeszkody”

---

1. Pierwsze kroki
2. Spostrzeżenia
3. Poprawiony model
4. Źródła

---

1. Pierwsze kroki

Robot jak się dowiedziałem aby poprawnie omijać przeszkody musi tworzyć mapę otoczenia. Wykonywane jest to przez specjalny wzór. Jednak aby go zaimplementować w pojeździe musiałem go pierwsze rozpisać. W tej czynności pomógł mi wolfram, który dla każdego wektora wyprowadził wzór, który nadawał się do implementacji w robocie. W taki sposób z takiego wzoru:

$$R(x', y', \theta') = \begin{bmatrix} \cos(\theta') & -\sin(\theta') & x' \\ \sin(\theta') & cos(\theta') & y' \\ 0 & 0 & 1 \end{bmatrix}$$

Mogłem przejść do takiego dla każdego kolejnego sensora:

```
dist = self.dist_to_obstacle()
self.position()
x_left = dist[0] * (0.5 * math.cos(self.get_fi()) - 1 / 2 * math.sqrt(3) * math.sin(
    self.get_fi())) + self.get_x() - 8 * math.sin(
    self.get_fi()) + 8 * math.cos(self.get_fi())
y_left = dist[0] * (0.5 * math.sin(self.get_fi()) + 1 / 2 * math.sqrt(3) * math.cos(
    self.get_fi())) + self.get_y() + 8 * math.sin(
    self.get_fi()) + 8 * math.cos(self.get_fi())

x_right = dist[1] * (0.5 * math.sqrt(3) * math.sin(self.get_fi()) + 0.5 * math.cos(
    self.get_fi())) + self.get_x() - 8 * math.sin(
    self.get_fi()) + 8 * math.cos(self.get_fi())
y_right = dist[1] * (0.5 * math.sin(self.get_fi()) - 0.5 * math.sqrt(3) * math.cos(
    self.get_fi())) + self.get_y() + 8 * math.sin(
    self.get_fi()) + 8 * math.cos(self.get_fi())

x_mid = dist[2] * math.cos(self.get_fi()) + self.x + 5 * math.cos(self.get_fi())
y_mid = dist[2] * math.sin(self.get_fi()) + self.y + 5 * math.sin(self.get_fi())

x_obst = x_left + x_right + x_mid - (3 * self.get_x())
y_obst = y_left + y_right + y_mid - (3 * self.get_y())
```

2. Spostrzeżenia

Jednym z nielicznych elementów do poprawy było poprawienie czułości jednego z czujnika. Przez jego gorsze odczyty maszyna nie zawsze wyłapywała przeszkodę poprawnie co sprawiało, że nie udało się jej ją ominąć. Zastosowanie skalowania poprawiło tą sytuację.

3. Poprawiony model

```
class Robot:
    def __init__(self):
        self.motor_right = LargeMotor(OUTPUT_B)
        self.motor_left = LargeMotor(OUTPUT_C)
        self.sensor_left = InfraredSensor('in1')
        self.sensor_right = InfraredSensor('in3')
        self.sensor_middle = UltrasonicSensor('in2')
        self.motor_right.position = 0
        self.motor_left.position = 0
        self.position_motor_old_right = 0
        self.position_motor_old_left = 0
        self.x = 0
        self.y = 0
        self.fi = 0
        self.R = 2.8
        self.L = 12
        self.v = 0
        self.omega = 0
        self.Vr = 0
        self.Vl = 0

    def position(self):
        n_r = self.motor_right.position
        n_l = self.motor_left.position
        delta_n_r = n_r - self.get_position_motor_old_right()
        delta_n_l = n_l - self.get_position_motor_old_left()

        D_r = 2 * math.pi * self.get_R() * (delta_n_r / 360)
        D_l = 2 * math.pi * self.get_R() * (delta_n_l / 360)

        Dc = (D_r + D_l) / 2

        self.x = self.get_x() + (Dc * math.cos(self.get_fi()))
        self.y = self.get_y() + (Dc * math.sin(self.get_fi()))
        self.fi = self.get_fi() + ((D_r - D_l) / self.L)
        self.position_motor_old_right = n_r
        self.position_motor_old_left = n_l

    def angle_to_goal(self):
        angle = self.avoid_obstacle()
        psi = math.atan2(angle[1], angle[0])
        return psi

    def go_to_goal(self):
        e = self.angle_to_goal() - self.get_fi()
        e = math.atan2(math.sin(e), math.cos(e))
        return e

    def run(self, e):
        k = 200
        self.motor_right.run_forever(speed_sp=290 + k * e)
        self.motor_left.run_forever(speed_sp=290 - k * e)

    def stop(self):
        self.motor_right.stop()
        self.motor_left.stop()
```

```
def dist_to_obstacle(self):
    return [self.sensor_left.value() * 0.35, self.sensor_right.value() * 0.7, self.sensor_middle.value() / 10]

def avoid_obstacle(self):
    dist = self.dist_to_obstacle()
    self.position()
    x_left = dist[0] * (0.5 * math.cos(self.get_fi()) - 1 / 2 * math.sqrt(3) * math.sin(
        self.get_fi())) + self.get_x() - 8 * math.sin(
        self.get_fi()) + 8 * math.cos(self.get_fi())
    y_left = dist[0] * (0.5 * math.sin(self.get_fi()) + 1 / 2 * math.sqrt(3) * math.cos(
        self.get_fi())) + self.get_y() + 8 * math.sin(
        self.get_fi()) + 8 * math.cos(self.get_fi())

    x_right = dist[1] * (0.5 * math.sqrt(3) * math.sin(self.get_fi()) + 0.5 * math.cos(
        self.get_fi())) + self.get_x() - 8 * math.sin(
        self.get_fi()) + 8 * math.cos(self.get_fi())
    y_right = dist[1] * (0.5 * math.sin(self.get_fi()) - 0.5 * math.sqrt(3) * math.cos(
        self.get_fi())) + self.get_y() + 8 * math.sin(
        self.get_fi()) + 8 * math.cos(self.get_fi())

    x_mid = dist[2] * math.cos(self.get_fi()) + self.x + 5 * math.cos(self.get_fi())
    y_mid = dist[2] * math.sin(self.get_fi()) + self.y + 5 * math.sin(self.get_fi())

    x_obst = x_left + x_right + x_mid - (3 * self.get_x())
    y_obst = y_left + y_right + y_mid - (3 * self.get_y())
    return [x_obst, y_obst]

def move_target(self):
    btn = Button()
    while True:
        self.run(self.go_to_goal())
        if btn.any():
            break
    self.stop()
```

4. Źródła

- Wykład
- http://mumin.pl/Robotyka/LEGO/Unikanieprzeszkod.html