LAPORAN TUGAS SIMULASI NAVIGASI ROBOT E-PUCK DI LABIRIN HORIZONTAL

Dibuat Oleh:

Augryes Farha Slistya Negara

(1103210115)

1. Pendahuluan

1.1 Latar Belakang

Robot mobile semakin sering digunakan dalam berbagai aplikasi, termasuk dalam bidang eksplorasi, pengawasan, dan navigasi otomatis. Salah satu tantangan dalam pengembangan robot mobile adalah kemampuan untuk melakukan navigasi secara mandiri di lingkungan yang kompleks seperti labirin. Dalam tugas ini, kami menggunakan robot e-puck yang disimulasikan menggunakan perangkat lunak Webots untuk mengikuti dinding kiri di dalam labirin horizontal.

1.2 Tujuan

Tujuan dari proyek ini adalah untuk mengembangkan algoritma sederhana menggunakan Python yang memungkinkan robot e-puck untuk bergerak dari titik awal di kiri bawah hingga mencapai titik tujuan di kanan atas dengan mengikuti dinding kiri pada labirin horizontal.

2. Metodologi

2.1 Deskripsi Lingkungan

Lingkungan simulasi yang digunakan adalah labirin horizontal yang terdiri dari beberapa lorong dengan dinding pembatas. Labirin memiliki beberapa rintangan dan jalur yang memerlukan robot untuk berbelok mengikuti dinding kiri. Robot e-puck ditempatkan di sudut kiri bawah dengan tujuan mencapai sudut kanan atas.

2.2 Deskripsi Robot E-Puck

E-puck adalah robot kecil dengan beberapa sensor yang dapat digunakan untuk navigasi, yaitu:

- **8 Sensor Inframerah (IR)**: Digunakan untuk mendeteksi jarak terhadap dinding atau objek.
- **2 Motor Roda**: Mengendalikan pergerakan robot dengan mengubah kecepatan motor kiri dan kanan.

2.3 Algoritma Navigasi

Algoritma yang digunakan adalah navigasi dengan mengikuti dinding kiri (left-wall following). Metode ini menggunakan sensor jarak inframerah untuk mendeteksi jarak antara robot dan dinding di sekitarnya, serta melakukan penyesuaian arah gerak berdasarkan input sensor.

Langkah-langkah algoritma:

1. **Inisialisasi Sensor dan Motor**: Mengaktifkan sensor jarak dan mengatur motor untuk kecepatan bebas.
2. **Navigasi Berdasarkan Sensor**:
    - Jika ada dinding di depan, robot berbelok ke kanan.
    - Jika tidak ada dinding di samping kiri, robot sedikit berbelok ke kiri.
    - Jika berada di ujung lorong, robot berbelok ke atas untuk melanjutkan navigasi.
    - Jika tidak ada rintangan di depan atau di samping kiri, robot akan terus berjalan lurus.
3. **Mengatur Kecepatan Motor**: Mengubah kecepatan motor kiri dan kanan sesuai dengan kondisi sensor untuk menghindari tabrakan dan mengikuti jalur yang tersedia.

2.4 Kode Program

Kode Python yang digunakan untuk simulasi adalah sebagai berikut:

```
from controller import Robot

TIME_STEP = 64

MAX_SPEED = 6.28

robot = Robot()

prox_sensor = []

sensor_names = ["ps0", "ps1", "ps2", "ps3", "ps4", "ps5", "ps6", "ps7"]

for name in sensor_names:

    sensor = robot.getDevice(name)

    sensor.enable(TIME_STEP)

    prox_sensor.append(sensor)

left_motor = robot.getDevice("left wheel motor")

right_motor = robot.getDevice("right wheel motor")

left_motor.setPosition(float('inf'))

right_motor.setPosition(float('inf'))

left_motor.setVelocity(0.0)

right_motor.setVelocity(0.0)

FRONT_THRESHOLD = 150.0

SIDE_THRESHOLD = 80.0

TURN_THRESHOLD = 100.0

while robot.step(TIME_STEP) != -1:

    front = prox_sensor[0].getValue()

    front_right = prox_sensor[1].getValue()
```

```
front_left = prox_sensor[6].getValue()
left_side = prox_sensor[7].getValue()
left_speed = 0.5 * MAX_SPEED
right_speed = 0.5 * MAX_SPEED
if front > FRONT_THRESHOLD or front_left > FRONT_THRESHOLD:
    left_speed = 0.5 * MAX_SPEED
    right_speed = -0.2 * MAX_SPEED
elif left_side < SIDE_THRESHOLD:
    left_speed = 0.3 * MAX_SPEED
    right_speed = 0.5 * MAX_SPEED
elif front_right > TURN_THRESHOLD:
    left_speed = -0.2 * MAX_SPEED
    right_speed = 0.5 * MAX_SPEED
else:
    left_speed = 0.5 * MAX_SPEED
    right_speed = 0.5 * MAX_SPEED
left_motor.setVelocity(left_speed)
right_motor.setVelocity(right_speed)
```

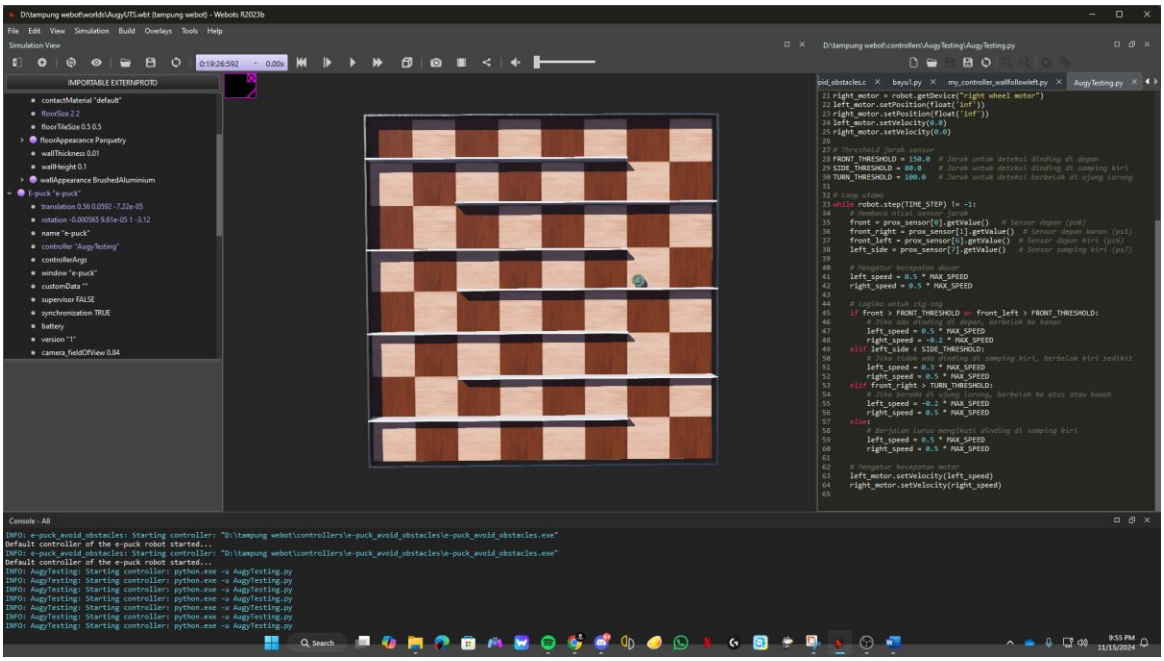
3. Hasil dan Analisis

3.1 Hasil Simulasi

Dari simulasi yang dilakukan, robot e-puck berhasil melakukan navigasi dari sudut kiri bawah hingga mencapai sudut kanan atas pada labirin. Robot mengikuti dinding kiri dengan baik dan mampu berbelok di ujung lorong saat diperlukan. Robot juga berhasil menghindari tabrakan dengan dinding.

3.2 Analisis Hasil

Algoritma left-wall following terbukti efektif untuk lingkungan labirin horizontal ini. Robot dapat menjaga jarak aman dari dinding menggunakan sensor samping kiri (ps7). Namun, beberapa penyesuaian diperlukan untuk meningkatkan keakuratan navigasi, terutama pada tikungan tajam di mana robot mungkin memerlukan waktu lebih lama untuk berbelok.



Kelebihan:

- Algoritma sederhana dan efisien untuk labirin terstruktur.
- Menggunakan input sensor jarak yang sudah tersedia pada e-puck.

Kekurangan:

- Algoritma ini bergantung pada pola labirin yang terstruktur. Jika terdapat bagian labirin yang tidak memiliki dinding kiri, robot mungkin tidak dapat melanjutkan dengan baik.
- Kecepatan dasar yang konstan menyebabkan waktu tempuh yang lebih lama pada lorong panjang tanpa rintangan.

4. Kesimpulan

Dari hasil simulasi yang dilakukan, dapat disimpulkan bahwa:

1. Algoritma left-wall following efektif untuk navigasi di labirin horizontal yang memiliki dinding pembatas terstruktur.
2. Penggunaan sensor inframerah e-puck memungkinkan robot untuk mendeteksi jarak dinding dan menyesuaikan arah gerak secara real-time.
3. Perbaikan dan optimisasi lebih lanjut, seperti penggunaan kontroler PID, dapat diterapkan untuk meningkatkan performa robot terutama pada bagian tikungan tajam dan lorong panjang.

5. Rekomendasi

Untuk meningkatkan performa navigasi, beberapa rekomendasi yang dapat diberikan adalah:

- Mengimplementasikan kontrol PID untuk penyesuaian jarak ke dinding secara lebih halus.
- Menggunakan algoritma lain seperti kombinasi wall-following dengan line-following jika tersedia garis panduan di lantai.
- Mengatur kecepatan adaptif berdasarkan jarak ke dinding untuk mengurangi waktu tempuh.