Implementasi Algoritma *Q Learning* Pada Robot *Line Follower*

Irvan Pramana[1], dan Asrizal Deri Futra[1*]

[1]Politeknik Negeri Batam, Batam, Indonesia

[*]Email: deri@polibatam.ac.id

***Abstract*—Penggunaan metode algoritma *Q learning* pada robot mampu melakukan perbaikan tanpa harus memperbaharui aturan dari luar karena sifatnya *off policy* (dapat mengikuti aturan apapun untuk menghasilkan solusi optimal). Dalam sistem kerjanya, robot melakukan proses pembelajaran terhadap garis lintasan yang dilaluinya sehingga didapatkan suatu nilai untuk aksi yang telah dilakukan pada setiap *state* yang terdeteksi. Tujuan penelitian ini adalah membuat robot bergerak berdasarkan nilai *Q function* tertinggi yang dihasilkan oleh algoritma *Q learning*. Berdasarkan hasil pada pengujian penerapan algoritma *Q learning* pada robot *line follower*, persentase keberhasilan yang didapatkan adalah sebesar 100% untuk percobaan pertama, 66,67% untuk percobaan kedua, 100% untuk percobaan ketiga, 66,67% untuk percobaan keempat, dan 100% untuk percobaan kelima sehingga rata – rata keberhasilan sebesar 86,67%.**

Kata kunci: ***Robot line follower, Q function, Q learning***

I. PENDAHULUAN

PERKEMBANGAN teknologi telah banyak mengalami kemajuan dari masa ke masa terutama dalam dunia robotika. Kecerdasaan buatan atau AI (*Artificial Intelligence*) memegang peran penting dalam perkembangan dunia robotika karena memungkinkan sebuah robot dapat bergerak secara otomatis hanya dengan memasukkan perintah sederhana. Salah satu jenis robot yang umum dijumpai adalah robot *line follower*. Robot *line follower* merupakan sebuah robot yang dapat mengikuti garis secara otomatis dengan program yang telah didefinisikan terlebih dahulu.

A.S. Romadhon dan M. Fuad melakukan penelitian mengenai robot *line follower*. Pada penelitian tersebut, peneliti membuat perancangan sistem kontrol gerak menggunakan kontrol PID untuk menyelesaikan masalah bagaimana sebuah robot *line tracer* dapat melakukan kontrol gerak pada bidang datar berwarna putih dengan garis berwarna hitam dalam lingkungan statis, sehingga robot dapat mengikuti garis hitam tersebut. Robot dapat berjalan dengan baik pada lintasan lurus dan lengkung saat kondisi cahaya terang [1]. Selanjutnya, David melakukan penelitian dengan mengimplementasikan kendali logika *fuzzy* untuk pengaturan pergerakan motor supaya robot mengikuti jalur yang telah ditentukan [2]. A. Wajiansyah, dkk melakukan penelitian robot *line follower* dengan menggunakan *fuzzy logic*. Hail penelitian menunjukkan bahwa robot dapat mengikuti garis [3]. Made Santo G, dkk melakukan penelitian pada sistem navigasi robot otonom beroda dalam medan yang tidak terstruktur dengan sistem kendali *Behavior-Based* Robot [4]. V. R. Cruz-Álvarez, dkk melakukan penelitian mengenai robot *follower* menggunakan Lego *Mindstorms* Kit dan Q *Learning* [5]. S. Arifin dkk melakukan penelitian mengenai peggunaan metode Q *learning* untuk pencarian *rute line follower mobile robot* pada *maze* [6].

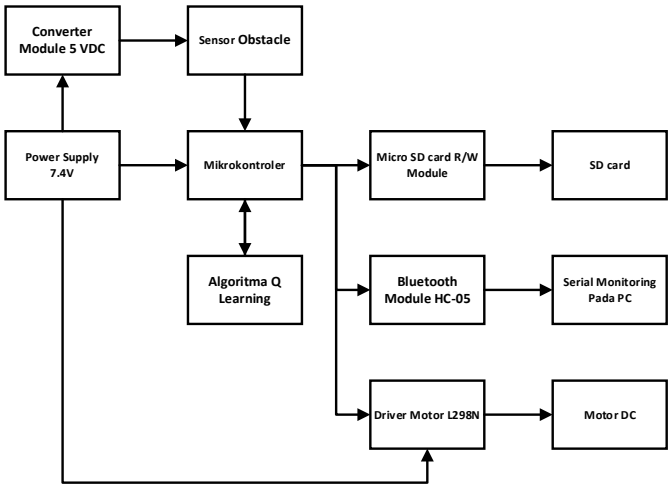
Pada penelitian ini, algoritma *Q learning* diimplementasikan pada robot *line follower* di mana robot dapat melakukan pemilihan aksi dengan nilai yang optimal berdasarkan nilai *Q function* yang dihasilkan oleh algoritma *Q learning* tanpa diperlukannya pembaharuan aturan dari luar.

II. METODE

A. Perancangan dan Perakitan Perangkat Keras

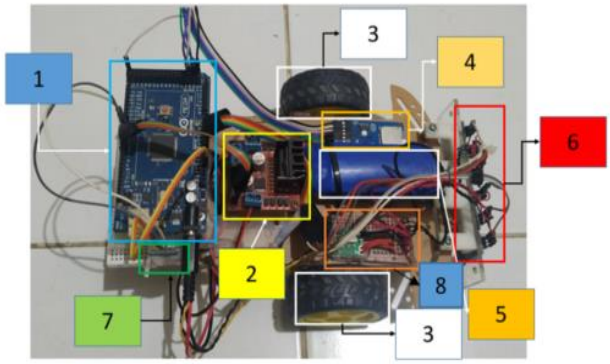
Pada perancangan perangkat keras, diperlukan beberapa komponen yaitu 6 buah sensor *obstacle* sebagai pendeteksi garis lintasan, 1 buah Arduino ATMega 2560 sebagai mikrokontroler, 1 buah *driver* motor L298N sebagai pengatur kecepatan dan arah putaran motor, 2 buah roda plastik sebagai roda robot, 2 buah motor DC sebagai penggerak roda plastik, 1 buah *Micro SD card module* sebagai penghubung antara Arduino dengan *SD card*, 1 buah *SD Card* untuk menyimpan data penelitian, 1 buah *Converter Module 5V DC* sebagai pengubah tegangan menjadi 5V DC, 1 buah *Bluetooth Module* HC-05 sebagai serial komunikasi antara mikrokontroler dengan aplikasi Arduino, 1 buah Baterai Lipo 2 *Cell* sebagai sumber tegangan, dan kabel *jumper* secukupnya. Selanjutnya komponen tersebut dirakit mengikuti blok diagram pada gambar 1. Dari gambar 1, *power supply* akan menyuplai tegangan sebesar ± 7,4 VDC pada 3 buah komponen yaitu *converter module* 5VDC, Arduino ATMega 2560, dan *driver motor L298N*. Pada *converter module* 5 VDC terjadi penurunan tegangan dari tegangan sumber ± 7,4 VDC menjadi 5 VDC. Tegangan ini digunakan sebagai tegangan input dari sensor *IR obstacle*. Sedangkan pada Arduino ATMega 2560 dan *driver motor* L298N, tegangan dari *power supply* digunakan sebagai tegangan kerja untuk menghidupkan kedua komponen tersebut. Setelah komponen ini aktif dan bekerja, selanjutnya akan terjadi sebuah proses sistem kerja dari perangkat keras. Pertama kali

sensor *IR obstacle* akan mendeteksi garis, selanjutnya hasil pendeteksian akan diproses di dalam Arduino ATMega 2560 dengan algoritma *Q learning*. Hasil dari proses akan dilanjutkan ke 3 buah komponen yaitu *Micro* SD *card module*, *Bluetooth Module* HC-05, dan *driver* motor L298N. Pada bagian *Micro* SD *card module*, data hasil pemrosesan akan disimpan pada sebuah SD *card*. Pada bagian *Bluetooth Module* HC-05, data akan diteruskan pada serial monitor di PC. Sedangkan pada *driver* motor L298N, hasil proses data akan diubah menjadi sebuah gerakan pada motor DC.



Gambar 1. Blok diagram sistem kerja perangkat keras

Gambar 2 merupakan tampilan robot ketika semua komponen selesai dirakit.

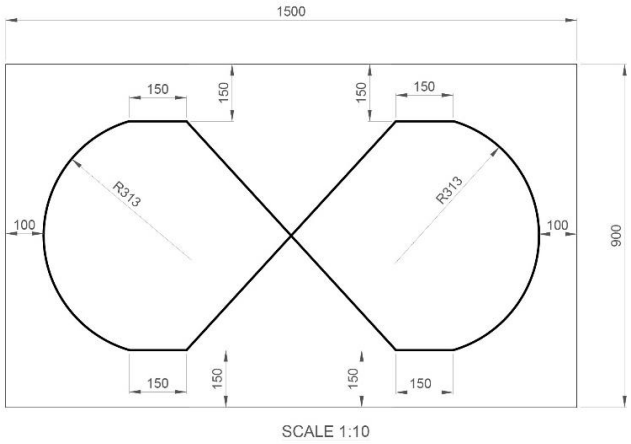


Keterangan:
1. Arduino ATMega 2560.
2. *Driver* Motor L298N.
3. Motor DC + Roda Plastik.
4. *Micro SD Card + SD Card*.
5. Baterai Lipo 2 Cell.
6. Sensor *Obstacle*.
7. *Bluetooth Module* HC-05
8. *Converter* Module 5V DC.

Gambar 2. Robot *line follower* dan keterangan komponen

B. *Perancangan Lintasan*

Lintasan dirancang menyerupai angka 8 (delapan) dengan warna hitam pada bagian garisnya dan warna putih untuk *background* lintasan. Pemilihan warna ini disebabkan kedua warna ini memiliki warna yang kontras sehingga nilai dari hasil pembacaan memiliki nilai yang kontras antara satu dengan yang lainnya sehingga memudahkan robot dalam mendeteksi garis lintasan.



Gambar 3. Desain Lintasan

Gambar 3 merupakan desain lintasan dengan dimensi dalam satuan mm dan skala yang digunakan adalah 1:10. Lintasan ini memiliki panjang 150 cm dan lebar 90 cm. Jarak antara tepi lintasan dengan tepi garis hitam untuk sisi kiri kanan sebesar 10 cm sedangkan atas dan bawah sebesar 15 cm. Untuk lebar garis warna hitam memiliki nilai 3 cm dikarenakan lebar ideal untuk 2 buah sensor yang terletak ditengah yaitu 3 cm.

C. *Perancangan Perangkat Lunak*

Dalam perancangan perangkat lunak ini dibagi menjadi 3 bagian yaitu fase *learning*, penyimpanan nilai *Q function*, dan pengambilan aksi berdasarkan nilai maksimum *Q function*.

a. Fase Learning

Fase *learning* adalah proses pembelajaran robot terhadap lingkungan yang memiliki tujuan mendapatkan nilai *Q function* pada masing –masing keadaan (*state*) yang terdeteksi di setiap aksi yang diambil. Terdapat 3 komponen yang diperlukan di dalam fase *learning* yaitu penentuan banyaknya jumlah *state*, penentuan banyaknya jumlah aksi, dan penentuan besaran *reward*. Komponen pertama adalah penentuan jumlah *state* yang didapatkan dari banyaknya sensor yang digunakan sehingga didapat persamaan (1).

$$Jumlah\ state = 2^n \quad (1)$$

Nilai 2 merupakan jumlah keadaan sensor saat mendeteksi garis yaitu ON atau OFF. Sedangkan n adalah banyaknya sensor yang digunakan. Sehingga 2^6 sama dengan 64. Dikarenakan perhitungan *state* dimulai dari angka 0 maka jumlah maksimum *state* berubah menjadi 63. Pada masing – masing sensor memiiki nilainya sendiri saat berhasil mendeteksi garis yang ditunjukan pada tabel I.

TABEL I
NILAI SETIAP SENSOR

Sensor	S1	S2	S3	S4	S5	S6
Nilai	1	2	4	8	16	32

Nilai pada tabel I terbentuk berdasarkan pendekatan nilai bit saat sensor aktif atau ON, di mana S1 merupakan bit 0, S2 adalah bit 1, S3 adalah bit 2, S4 adalah bit 3, S5 adalah bit 4, dan S6 adalah bit 5. *State* didapat dari hasil akumulasi dari nilai pada tabel 1 saat mendeteksi garis lintasan.

Komponen kedua adalah penentuan banyaknya jumlah aksi ditentukan oleh *user* di mana terdapat 5 buah aksi yang digunakan seperti pada tabel II.

TABEL II
AKSI ROBOT

Indeks Aksi	Keterangan
0	Berhenti
1	Belok kiri
2	Maju
3	Belok Kanan
4	Mundur

Dalam bergerak, robot hanya akan mengambil aksi belok kiri, maju, dan belok kanan saat berada dalam kondisi ideal. Kondisi ideal yang dimaksud adalah kondisi di mana robot masih berada atau masih mendeteksi garis lintasan. Aksi berhenti digunakan untuk jeda antara aksi dengan durasi 2700 ms. Penggunaan jeda sebesar 2700 ms bertujuan untuk meminimalisir gangguan pembacaan sensor akibat getaran yang ditimbulkan saat robot melakukan aksi. Sedangkan aksi mundur digunakan saat robot berada di luar garis lintasan.

Untuk kecepatan motor pada masing – masing aksi, digunakan sebuah cara dengan memberikan nilai PWM tertentu pada kedua buah motor seperti pada tabel III dengan durasi 100 ms.

TABEL III
NILAI PWM PADA SETIAP AKSI

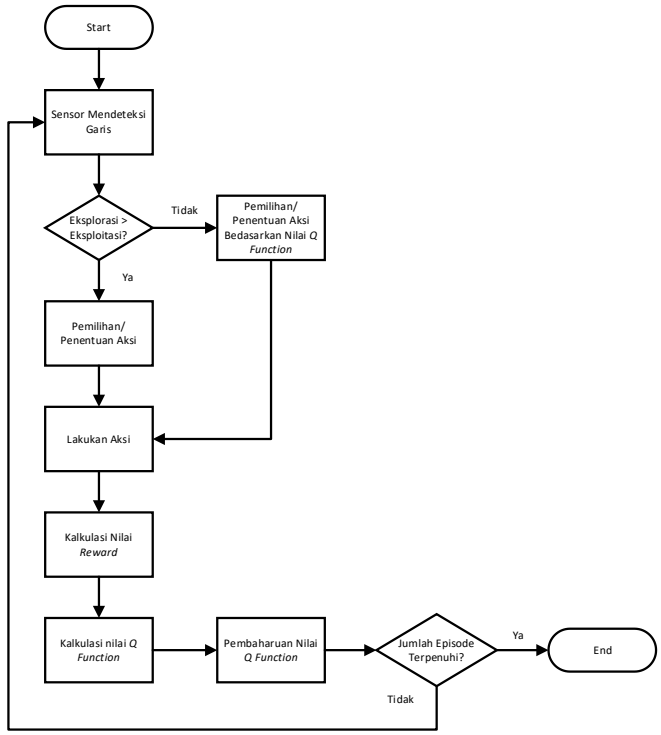
Aksi	Nilai PWM	
	Motor Kiri	Motor Kanan
Berhenti	0	0
Belok Kiri	0	150
Maju	150	150
Belok Kanan	150	0
Mundur	150	150

Komponen ketiga adalah penentuan besaran *reward*. *Reward* adalah hadiah yang didapatkan robot saat selesai melakukan aksi. Nilai *reward* bisa berupa positif atau negatif. Adapun besaran *reward* seperti yang ditunjukan oleh persamaan (2).

$$\text{Besaran } Reward = \{-3,\ 1,\ 4,\ 4,\ 1, -3\} \quad (2)$$

Nilai pada masing – masing index pada persamaan (2) merupakan representasi kondisi pada setiap sensor yang aktif. Idealnya robot akan memiliki nilai *reward* yang besar saat robot berada di tengah garis lintasan dan apabila robot berada di pinggir garis lintasan maka nilai *reward* yang didapatkan akan cenderung kecil ataupun bernilai negatif.

Setelah menentukan komponen yang diperlukan, selanjutnya dibuatlah sistem kerja dari fase *learning* seperti pada gambar 4.



Gambar 4. *Flowchart* sistem kerja fase *learning*

Gambar 4 adalah *flowchart* sistem kerja fase *learning*, di mana tahapannya seperti berikut:

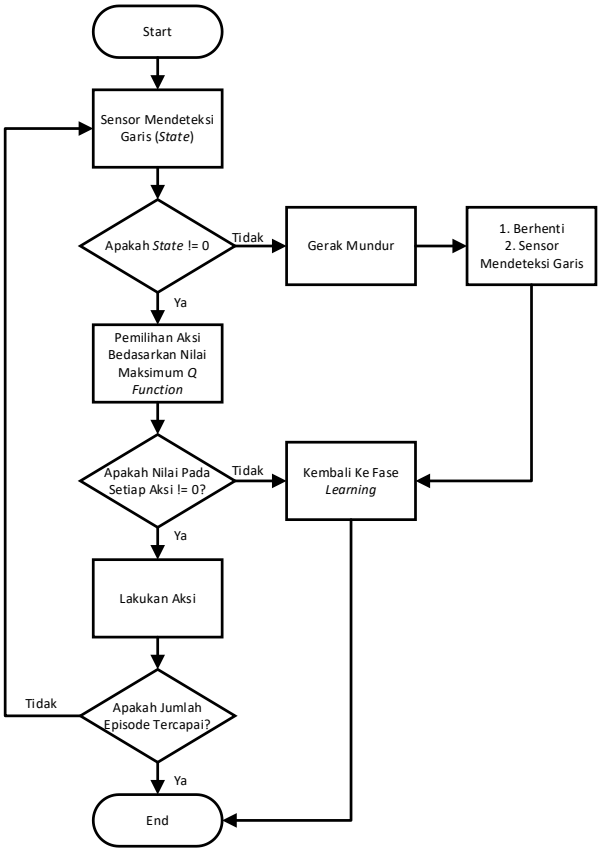
1) Pendeteksian garis lintasan untuk mendapatkan nilai *state*.
2) Melakukan perbandingan nilai antara eksplorasi dan eksploitasi.
3) Jika nilai ekploitasi lebih besar maka aksi yang diambil berdasarkan dari nilai *Q function* yang tertinggi.
4) Jika nilai ekplorasi lebih besar maka robot memilih aksi acak.
5) Melakukan aksi dari penentuan aksi.
6) Kalkulasi nilai *reward* dari aksi yang telah dilakukan.
7) Kalkulasi nilai *Q function* untuk memperbaharui nilai *Q function* pada *state* yang terdeteksi.

b. Penyimpanan Nilai Q Function

Pada bagian ini merupakan proses penyimpanan nilai *Q function* dari masing – masing *state* pada setiap aksi ke *SD card* yang memiliki tujuan untuk mempermudah dalam proses pengolahan data.

c. Pengambilan Aksi Berdasarkan Nilai Maksimum Q Function

Pada bagian ini, robot melakukan aksi berdasarkan nilai *Q function* yang tertinggi. Hal ini bertujuan untuk proses pengujian pembuktian bahwa robot benar – benar bergerak berdasarkan aksi dengan nilai yang tertinggi. Proses bagian ini tidak akan memperbaharui nilai *Q function* dan juga aksi yang dilakukan tidak akan mendapatkan *reward*.



Gambar 5. *Flowchart* Sistem Kerja Pengambilan Aksi Berdasarkan Nilai Maksimum *Q Function*

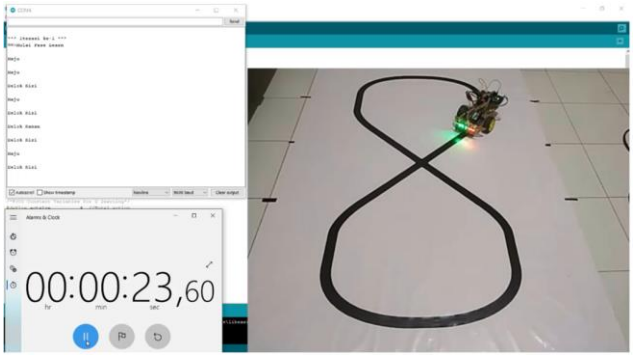
Gambar 5 adalah *flowchart* dari sistem kerja pengambilan aksi berdasarkan nilai maksimum *Q function* di mana sistem kerja ini memiliki tahapan sebagai berikut:

1) Pendeteksian garis untuk mendapatkan *state*.
2) Pemeriksaan nilai *state*. Jika bernilai 0 maka diindikasikan bahwa robot keluar jalur lalu robot akan bergerak mundur sekali.
3) Pemilihan aksi berdasarkan nilai maksimum *Q function* pada masing – masing kolom aksi. Jika semua aksi memiliki nilai 0 maka robot kembali ke fase *learning*.
4) Pengambilan aksi dari hasil pemilihan aksi pada tahap sebelumnya.
5) Pemeriksaan jumlah episode. Jika jumlah episode tercapai maka robot kembali ke fase *learning*. Banyaknya episode yang digunakan adalah 50.

III. HASIL DAN PEMBAHASAN

A. *Hasil Fase Learning*

Fase *learning* pada penelitian ini dilakukan sebanyak 50 iterasi di mana setiap iterasinya robot melakukan perhitungan nilai *Q function* dari tiap *state* yang terdeteksi sebanyak 50 kali. Alasan penggunaan 50 iterasi dikarenakan robot telah banyak mengalami perubahan nilai pada masing – masing *state* yang terdeteksi di setiap kolom aksinya.



Gambar 6. Fase *Learning* Robot

Gambar 6 adalah proses fase *learning* robot di mana robot melakukan pendeteksian garis lintasan untuk mendapatkan *state* selanjutnya melakukan aksi untuk mendapatkan *reward* dan melakukan kalkulasi nilai *Q function* dari aksi yang telah dilakukan pada *state* yang terdeteksi. Nilai *Q function* pada masing – masing *state* di setiap kolom aksi akan mengalami perubahan seiring robot melakukan aksi untuk mendapatkan *reward*. Perubahan nilai tersebut berupa nilai positif ataupun nilai negatif. Perubahaan nilai ini hanya terjadi pada *state* yang terdeteksi seperti pada tabel IV.

TABEL IV
KETERANGAN PERUBAHAN NILAI Q FUNCTION

State Terdeteksi	Aksi Belok Kiri	Aksi Maju	Aksi Belok Kanan
State 0			
State 1			
State 2			
State 3			
State 4			
State 6			
State 7			
State 8			
State 12			
State 13			
State 14			
State 15			
State 16			
State 24			
State 28			
State 29			
State 30			
State 31			
State 32			
State 33			
State 34			
State 38			
State 45			
State 46			
State 47			
State 48			
State 49			
State 56			
State 60			
State 62			
State 63			

Keterangan :

- Perubahan nilai ke negatif
- Perubahan nilai ke positif
- Perubahan nilai dari negatif menjadi positif
- Tidak mengalami perubahan nilai

Untuk nilai yang didapatkan setelah 50 kali iterasi seperti pada tabel V.

TABEL V
NILAI Q FUNCTION SETELAH 50 KALI ITERASI

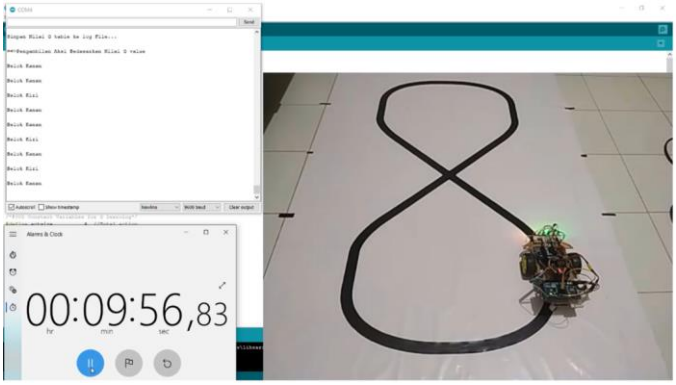
State Tedeteksi	Aksi Belok Kiri	Aksi Maju	Aksi Belok Kanan
0	-4.20	0	0
1	39.15	0	0
2	42.34	0	0
3	40.88	0	0
4	0	7.28	0
6	42.71	0	0
7	37.72	0	0
8	0	38.09	0
12	0	40.32	0
13	37.34	0	0
14	34.78	39.22	0
15	40.07	0	0
16	0	0	48.56
24	0	0	40.91
28	-4.20	32.63	0
29	-4.20	35.82	0
30	0	37.86	0
31	42.17	0	0
32	0	0	33.32
33	-4.20	-4.20	0
34	44.79	0	0
38	36.70	0	0
45	41.74	0	0
46	0	40.67	0
47	0	31.91	0
48	0	0	43.13
49	-4.20	-4.20	0
56	0	0	38.63
60	0	0	5.60
62	0	0	42.22
63	0	42.87	0

B. Pengujian Pengambilan Aksi Berdasarkan Nilai Q Function

Pengujian ini dilakukan setelah fase *learning* selesai, dimana robot mengambil aksi berdasarkan nilai *Q function*. Sama seperti fase *learning*, robot melakukan pendeteksian nilai *state* dan selanjutnya mengambil aksi. Akan tetapi dalam pengambilan aksi pada tahap pengujian berbeda dengan fase *learning* di mana pada tahapan pengujian aksi yang diambil adalah aksi dengan nilai yang paling tinggi.

Gambar 7 merupakan proses pengujian pengambilan aksi berdasarkan nilai *Q function*. Pengujian ini bertujuan untuk melihat kualitas dari aksi dengan nilai tertinggi yang diambil oleh robot. Kualitas yang dimaksud adalah robot dapat bergerak tanpa keluar garis. Untuk parameter keberhasilannya sendiri yaitu:

1) Robot tidak keluar jalur.
2) Robot kembali ke fase *learning* jika semua aksi pada *state* terdeteksi ≤ 0.



Gambar 7. Pengujian Pengambilan Aksi Berdasarkan Nilai *Q Function*

Percobaan dilakukan sebanyak 5 kali di mana pengambilan data pengujian dilakukan pada iterasi 36 sampai dengan 50 dikarenakan robot telah banyak melakukan pengambilan data dan kalkulasi nilai *Q function*. Adapun batas jumlah pengambilan aksi yang dilakukan yaitu sebanyak 50 kali. Hasil percobaan pengambilan aksi dapat dilihat pada tabel VI-X.

TABEL VI
HASIL PERCOBAAN PERTAMA

No.	Iterasi	Jumlah Pengambilan Aksi	Keterangan
1	36	50	Berhasil
2	37	50	Berhasil
3	38	9	Berhasil
4	39	50	Berhasil
5	40	50	Berhasil
6	41	50	Berhasil
7	42	50	Berhasil
8	43	45	Berhasil
9	44	50	Berhasil
10	45	50	Berhasil
11	46	50	Berhasil
12	47	50	Berhasil
13	48	50	Berhasil
14	49	29	Berhasil
15	50	50	Berhasil

TABEL VII
HASIL PERCOBAAN KEDUA

No.	Iterasi	Jumlah Pengambilan Aksi	Keterangan
1	36	10	Nilai ≤ 0
2	37	14	Keluar Jalur
3	38	50	Berhasil
4	39	22	Berhasil
5	40	16	Keluar Jalur
6	41	50	Berhasil
7	42	34	Berhasil
8	43	50	Berhasil
9	44	5	Keluar Jalur
10	45	0	Berhasil
11	46	1	Keluar Jalur
12	47	50	Berhasil
13	48	37	Nilai ≤ 0
14	49	1	Keluar Jalur
15	50	50	Berhasil

TABEL VIII
HASIL PERCOBAAN KETIGA

No.	Iterasi	Jumlah Pengambilan Aksi	Keterangan
1	36	50	Berhasil
2	37	50	Berhasil
3	38	50	Berhasil
4	39	50	Berhasil
5	40	50	Berhasil
6	41	50	Berhasil
7	42	50	Berhasil
8	43	50	Berhasil
9	44	50	Berhasil
10	45	50	Berhasil
11	46	17	Nilai ≤ 0
12	47	50	Berhasil
13	48	15	Nilai ≤ 0
14	49	50	Berhasil
15	50	50	Berhasil

TABEL IX
HASIL PERCOBAAN KEEMPAT

No.	Iterasi	Jumlah Pengambilan Aksi	Keterangan
1	36	50	Berhasil
2	37	50	Berhasil
3	38	3	Keluar Jalur
4	39	50	Berhasil
5	40	50	Berhasil
6	41	50	Berhasil
7	42	8	Keluar Jalur
8	43	50	Berhasil
9	44	50	Berhasil
10	45	50	Berhasil
11	46	1	Keluar Jalur
12	47	7	Keluar Jalur
13	48	18	Keluar Jalur
14	49	50	Berhasil
15	50	50	Berhasil

TABEL X
HASIL PERCOBAAN KELIMA

No.	Iterasi	Jumlah Pengambilan Aksi	Keterangan
1	36	50	Berhasil
2	37	50	Berhasil
3	38	5	Berhasil
4	39	50	Berhasil
5	40	50	Berhasil
6	41	50	Berhasil
7	42	50	Berhasil
8	43	50	Berhasil
9	44	50	Berhasil
10	45	50	Berhasil
11	46	50	Berhasil
12	47	50	Berhasil
13	48	50	Berhasil
14	49	50	Berhasil
15	50	50	Berhasil

Dari percobaan yang telah dilakukan, terdapat kegagalan pada percobaan 2 dan 4. Pada percobaan 2 dapat dilihat pada Tabel VII, kegagalan terjadi pada iterasi 37, 40, 44, 46, dan 49. Sedangkan pada percobaan 4 dapat dilihat pada tabel IX, kegagalan terjadi pada iterasi 38, 42, 46, 47, dan 48. Hal ini dikarenakan posisi robot yang cenderung berada di pinggir garis sehingga aksi yang dilakukan membuat robot keluar jalur.

Dalam 5 kali percobaan yang telah dilakukan maka didapatkan tingkat keberhasilan pada masing – masing percobaan ditunjukkan oleh Tabel XI.

TABEL XI
TINGKAT KEBERHASILAN

Percobaan ke-	Tingkat Keberhasilan
1	100%
2	66.67%
3	100%
4	66.67%
5	100%
Rata-rata	86.67%

IV. KESIMPULAN

Penelitian ini bertujuan agar robot dapat melakukan perbaikan kesalahan tanpa adanya campur tangan *user*. Dengan menggunakan algoritma *Q learning*, robot melakukan kalkulasi nilai untuk setiap aksi yang telah dilakukan pada setiap *state* yang terbaca berdasarkan nilai *reward* yang didapatkan. Nilai *reward* bernilai negatif jika robot keluar jalur dan bernilai positif jika robot berada di dalam jalur. Nilai kalkulasi ini disebut sebagai nilai *Q function*. Hasil pengujian pada percobaan 1, 3, dan 5 memiliki tingkat keberhasilan 100% dikarenakan robot mampu melaksanakan tugas dengan mengacu parameter keberhasilan yaitu robot tidak keluar jalur dan robot kembali ke fase *learning* apabila nilai *Q function* ≤ 0 pada setiap kolom aksi di *state* yang terdeteksi. Sedangkan pada percobaan 2 dan 4 memiliki tingkat keberhasilan 66.67% dikarenakan robot tidak mampu melaksanakan tugas dengan mengacu parameter keberhasilan. Sehingga nilai rata–rata keberhasilan sebesar 86.67%. Penelitian ini masih perlu untuk dikembangkan terutamanya pemilihan mikrokontroler agar proses kalkulasi nilai *Q function* berjalan cepat dan juga diperlukannya sebuah cara untuk meminimalisir getaran yang ditimbulkan saat robot melakukan aksi sehingga *delay* yang diperlukan untuk membuat sensor *IR obstacle* kembali stabil saat mendeteksi jalur menjadi kecil.

REFERENSI

[1] A.S. Romadhon and M. Fuad, "Perancangan Sistem Kontrol Gerakan Pada Robot Line Tracer," *Ilm. Mikrotek*, vol. 1, no. 1, pp. 53–58, 2013.

[2] David, "Kendali Logika Fuzzy Pada Robot Line Follower Line Follower Robot with Fuzzy Logic Control," *Citec J.*, vol. 3, no. 1, pp. 15–25, 2016.

[3] A. Wajiansyah, S. Supriadi, S. Nur, and A. B. Wicaksono P, "Implementasi Fuzzy Logic Pada Robot Line Follower," *J. Teknol. Inf. dan Ilmu Komput.*, vol. 5, no. 4, p. 395, 2018, doi: 10.25126/jtiik.201854747.

[4] M. Santo Gitakarma, "Perancangan Behavior-Based Robot Dengan Algoritma Fuzzy Q-Learning (Fql) Pada Sistem Navigasi Robot Otonom Beroda Dalam Medan Yang Tidak Terstruktur," *JST (Jurnal Sains dan Teknol.*, vol. 2, no. 1, pp. 139–150, 2013, doi: 10.23887/jst-undiksha.v1i1.1419.

[5] V. R. Cruz-Álvarez, E. Hidalgo-Peña, and H.-G. Acosta-Mesa, "A line follower robot implementation using Lego's Mindstorms Kit and Q-Learning," *Acta Univ.*, vol. 22, pp. 113–118, 2012, doi: 10.15174/au.2012.350.

[6] S. Arifin, A. Tandy Hermawan, and Y. Kristian, "Pencarian Rute Line Follower Mobile Robot Pada Maze Dengan Metode Q Learning," *J. Otomasi Kontrol dan Instrumentasi*, vol. 8, no. 1, p. 55, 2016, doi: 10.5614/joki.2016.8.1.5.