TESIS – EE185401

DESAIN PENGOLAHAN DATA MULTISENSOR MENGGUNAKAN FUZZY Q-LEARNING UNTUK SISTEM NAVIGASI MOBILE ROBOT

ARGA DWI PAMBUDI
07111650020003

DOSEN PEMBIMBING
Dr. Trihastuti Agustinah, ST., MT.
Ir. Rusdhianto Effendi AK., MT.

PROGRAM MAGISTER
BIDANG KEAHLIAN TEKNIK SISTEM PENGATURAN
DEPARTEMEN TEKNIK ELEKTRO
FAKULTAS TEKNOLOGI ELEKTRO
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA
2019

TESIS – EE185401

DESAIN PENGOLAHAN DATA MULTISENSOR MENGGUNAKAN FUZZY Q-LEARNING UNTUK SISTEM NAVIGASI MOBILE ROBOT

ARGA DWI PAMBUDI
07111650020003

DOSEN PEMBIMBING
Dr. Trihastuti Agustinah, ST., MT.
Ir. Rusdhianto Effendi AK., MT.

PROGRAM MAGISTER
BIDANG KEAHLIAN TEKNIK SISTEM PENGATURAN
DEPARTEMEN TEKNIK ELEKTRO
FAKULTAS TEKNOLOGI ELEKTRO
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA
2019

LEMBAR PENGESAHAN

Tesis disusun untuk memenuhi salah satu syarat memperoleh gelar
Magister Teknik (M.T)
di
Institut Teknologi Sepuluh Nopember

oleh:

Arga Dwi Pambudi
NRP. 07111650020003

Tanggal Ujian : 28 Desember 2018
Periode Wisuda: Maret 2019

Disetujui oleh:

1. Dr. Trihastuti Agustinah, ST., MT. (Pembimbing I)
   NIP: 196808121994032001
2. Ir. Rusdhianto Effendi AK., MT. (Pembimbing II)
   NIP: 195704241985021001
3. Prof. Ir. H. Abdullah Alkaff, M.Sc., Ph.D (Penguji)
   NIP: 195501231980031002
4. Prof. Dr. Ir. Achmad Jazidie, M.Eng. (Penguji)
   NIP: 195902191986101001

Dekan Fakultas Teknologi Elektro

Dr. Tri Arief Sardjono, S.T., M.T.
NIP. 197002121995121001

Halaman ini sengaja dikosongkan

PERNYATAAN KEASLIAN TESIS

Dengan ini saya menyatakan bahwa isi keseluruhan Tesis saya dengan judul "**DESAIN PENGOLAHAN DATA MULTISENSOR MENGGUNAKAN FUZZY Q-LEARNING UNTUK SISTEM NAVIGASI MOBILE ROBOT**" adalah benar-benar hasil karya intelektual mandiri, diselesaikan tanpa menggunakan bahan-bahan yang tidak diijinkan dan bukan merupakan karya pihak lain yang saya akui sebagai karya sendiri.

Semua referensi yang dikutip maupun dirujuk telah ditulis secara lengkap pada daftar pustaka. Apabila ternyata pernyataan ini tidak benar, saya bersedia menerima sanksi sesuai peraturan yang berlaku.

Surabaya, 10 Desember 2018

Arga Dwi Pambudi

NRP. 07111650020003

Halaman ini sengaja dikosongkan

DESAIN PENGOLAHAN DATA MULTISENSOR MENGGUNAKAN FUZZY Q-LEARNING UNTUK SISTEM NAVIGASI MOBILE ROBOT

Nama mahasiswa : Arga Dwi Pambudi
NRP : 07111650020003
Pembimbing : 1. Dr. Trihastuti Agustinah, ST., MT.
2. Ir. Rusdhianto Effendi AK., MT.

ABSTRAK

Mobile robot merupakan alat yang sering dipakai di dalam banyak bidang kehidupan manusia. Untuk menunjang kebutuhan manusia, penelitian tentang *mobile robot* dikembangkan dalam berbagai bidang dengan permasalahan yang sering terjadi adalah pada sistem navigasi, dimana robot bisa menuju ke titik tujuan tanpa menabrak halangan. Salah satu metode yang digunakan adalah Fuzzy Q-Learning menggunakan *reinforcement learning* sehingga robot bisa belajar secara mandiri menggunakan sensor pada robot dengan melihat kondisi yang terjadi.

Pada beberapa penelitian terdapat perbedaan dalam pemilihan desain dari penggunaan FQL yang digunakan, sehingga perlu dilakukan penelitian untuk membandingkan desain yang dibuat. Pada Tesis ini membandingkan hasil antara penggunaan penalaan reinforcement poin menggunakan perubahan state ($S_{t-1} \rightarrow S_t$) dan dengan menggunakan kondisi state terakhir (S_t). Selain itu, perbandingan dilakukan dengan menggunakan sudut region 2,3,4 dan 5 sensor untuk menghindari halangan. Untuk menunjang penggunaan FQL pada sistem navigasi akan dilakukan desain pengolahan data sensor pada proses *fuzzifikasi* untuk mengurangi jumlah state yang terjadi.

Hasil simulasi menunjukkan bahwa dengan *reinforcement* poin menggunakan perubahan state dan 5 sudut region sensor menghasilkan performa yang paling baik untuk halangan dinamis, yaitu waktu tercepat menuju target sedangkan untuk halangan statis cukup dengan *reinforcement point* state terakhir dan 3 sudut region.

Kata kunci: *Reinforcement Learning, Q-Learning, Fuzzy.*

Halaman ini sengaja dikosongkan

MULTISENSOR DATA PROCESSING DESIGN USING FUZZY Q-LEARNING FOR MOBILE ROBOT NAVIGATION SYSTEM

By : *Arga Dwi Pambudi*
Student Identity Number : *07111650020003*
Supervisory : *1. Dr. Trihastuti Agustinah, ST., MT.*
2. Ir. Rusdhianto Effendi AK., MT.

ABSTRACT

Mobile robots are device that are often used in many areas of human life. In order to support human needs, research on mobile robots developed in various fields with problems that often occur is in the navigation system, where the robot can go to the destination without colliding obstacles. One of the methods is Fuzzy Q-Learning using reinforcement learning so that robots can learn independently using sensors on robots by looking at the conditions that occur.

In some research there are differences in the selection of designs from the use of FQL, so it is necessary to do research to compare the designs. In this thesis, the comparative of the results of using tuning reinforcement points using transition changes $(S_{t-1} \rightarrow S_t)$ *and current state* (S_t) *condition will be conducted. In addition, it will also compare the use of angles of regions 2,3,4 and 5 sensors used to avoid obstacles. To support the use of FQL on the navigation system a sensor data processing design will be carried out in the fuzzification process to reduce the number of states that occur.*

The simulation results show that reinforcement points using transition state and 5 angles of the sensor region produces the best performance for dynamic obstacles, which is the fastest time to the target while for static obstacles enough with the last reinforcement point current state and 3 angels of the regions.

Keywords— Reinforcement Learning, Q-Learning, Fuzzy.

Halaman ini sengaja dikosongkan

KATA PENGANTAR

Alhamdulillah, Segala puji bagi Allah SWT, Tuhan semesta alam.

Tesis ini disusun untuk memenuhi persyaratan guna menyelesaikan pendidikan Magister pada Bidang Studi Teknik Sistem Pengaturan, Fakultas Teknologi Elektro, Institut Teknologi Sepuluh Nopember (ITS) Surabaya dengan judul:

"DESAIN PENGOLAHAN DATA MULTISENSOR MENGGUNAKAN FUZZY Q-LEARNING UNTUK SISTEM NAVIGASI MOBILE ROBOT"

Terima kasih pada semua pihak yang telah memberikan dukungan dalam pengerjaan Tesis ini. Terutama untuk Dr. Trihastuti Agustinah, S.T., M.T dan Ir. Rusdhianto Effendi AK., MT, selaku pembimbing atas segala bimbingan, dukungan, dan motivasi hingga terselesaikannya Tesis ini. Penulis juga berterima kasih kepada teman-teman Teknik Sistem Pengaturan yang selalu memberikan kritik demi terselesaikannya Tesis ini. Penulis juga ingin menyampaikan permintaan maaf kepada orang tua karena belum bisa membalas pengorbanan beliau berdua. Semoga buku Tesis ini dapat memberikan manfaat untuk pembaca yang tertarik pada pengembangan tekonologi *mobile robot*.

Surabaya, 10 Desember 2018

Penulis

Halaman ini sengaja dikosongkan

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

Halaman ini sengaja dikosongkan

BAB 1
PENDAHULUAN

1.1 Latar Belakang

Seiring berjalannya waktu, peranan robot untuk membantu tugas manusia semakin banyak dan salah satu robot yang cukup populer dikembangkan saat ini adalah *mobile robot.* Permasalahan utama terjadi pada sistem navigasi untuk membuat robot bergerak secara benar dari satu titik awal menuju ke target tanpa menabrak *obstacle* [1].

Perkembangan robot saat ini sudah sampai pada tahap interaksi dengan manusia oleh sebab itu dibutuhkan kemampuan belajar bagi robot dengan mengamati keadaan sekitarnya dan berbagai kondisi yang mungkin tidak diperkirakan oleh pembuat robot bisa terjadi. Proses pembelajaran robot secara mandiri berdasarkan lingkungan disekitarnya disebut *reinforcement learning*. Berbagai penelitian terkait *reinforcement learning* telah dilakukan sebelumnya [1-3] [6-8].

Penggunaan *reinforcement learning* jarang diterapkan untuk *obstacle* yang dinamis, dikarenakan akan lebih banyak masalah yang akan muncul dan memiliki *state* tak terbatas [2]. Pada penelitian ini akan menerapkan *Fuzzy Q-Learning* (FQL) yang digunakan untuk mengeneralisasi *state*, sehingga *state* yang ada tidak terlalu banyak untuk mengurangi komsumsi memori pada prosesor.

Pada penelitian sebelumnya sudah dilakukan penggunaan *metode Fuzzy Q-Learning* (FQL) untuk keperluan navigasi dengan obstacle statis [1], akan tetapi pengujian dilakukan untuk halangan yang statis saja dan hanya menggunakan dua buah sensor. Pada penelitian ini digunakan robot SCITOS G5 yang dilengkapi 24 sensor sonar di sekeliling robot sehingga pemilihan path yang dilalui lebih terarah dan diharapkan bisa menghasilkan hasil yang lebih memuaskan.

Pada penelitian ini akan mencoba untuk meningkatkan performa dari metode FQL yang digunakan pada paper yang berjudul *"Compact Fuzzy Q-Learning For Autonomous Mobile Robot Navigation"* dengan cara manambahkan input fuzzifikasi dan mengubah sistematika pada reinforcement learning yang

digunakan untuk menghindari *obstacle* pada saat robot berjalan menuju target, baik *obstacle* yang statis maupun dinamis yang nantinya akan dibandingkan dengan penggunaan yang dimplementasikan pada robot dan lingkungan yang sama. Pengujian juga akan dilakukan menggunakan 3, 4 dan 5 input fuzzifikasi.

1.2 Rumusan Masalah

Rumusan masalah pada tesis ini adalah bagaimana cara melakukan pengolahan data multisensor sebagai masukan kontroller, desain penalaan fungsi *reward point.*

1.3 Tujuan

Tujuan dari tesis ini dihasilkan kontroller Fuzzy Q-Learning (FQL) yang mampu untuk menuju target tanpa menabrak halangan dengan waktu yang tercepat dari beberapa desain yang diuji.

1.4 Batasan Masalah

Dalam penelitian ini terdapat batasan masalah untuk membatasi permasalahan yang muncul diantaranya,

1. Obstacle dipertimbangkan jika terdeteksi oleh sensor sonar.
2. Kecepatan *obstacle* tidak melebihi dari kecepatan robot.
3. Jumlah *obstacle* dibatasi maksimal 2 buah.

1.5 Kontribusi

Kontribusi dari penelitian ini adalah menghasilkan desain penalaan FQL berdasarkan masukan multisensor dan penalaan fungsi reward point dengan melihat kondisi sebelumnya untuk tujuan *obstacle avoidance* pada *mobile robot*.

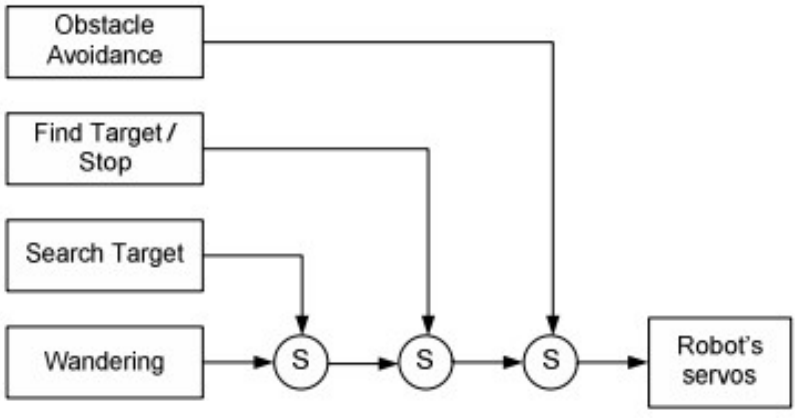
BAB 2
KAJIAN PUSTAKA

Pada bab ini penulis akan membahas tentang materi yang berhubungan dengan thesis yang akan dikerjakan berisi tentang kajian penelitian terkait dan teori dasar.

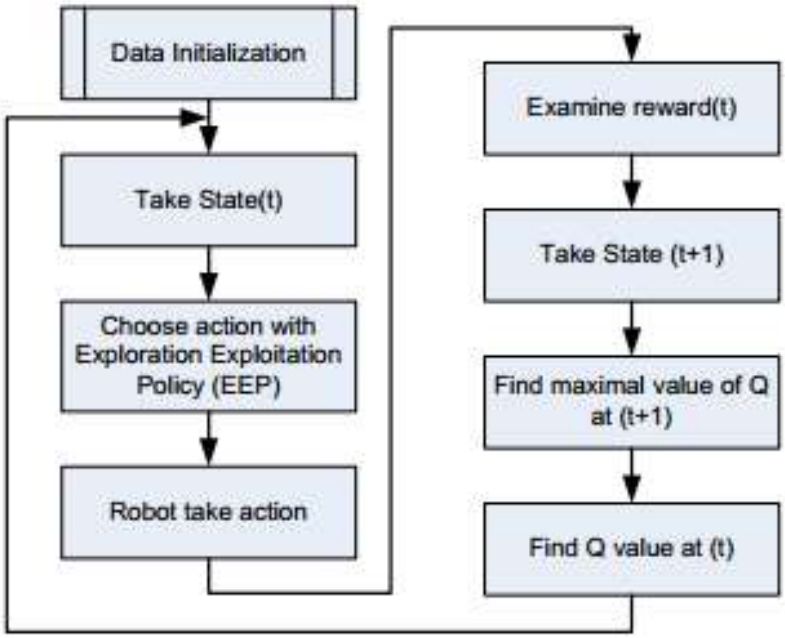
2.1 Kajian Penelitian Terkait

2.1.1 Compact Fuzzy Q Learning for Autonomous Mobile Robot Navigation [1]

Penelitian ini dibahas tentang penggunaan *metode Compact Fuzzy Q-Learning* (*CFQL*) untuk navigasi robot dari posisi awal ke posisi tujuan dengan adanya *obstacle* statis. Pada Gambar 2.1 ditunjukkan arsitektur navigasi robot otomatis, dimana robot harus memiliki beberapa tingkah laku untuk memenuhi navigasi yang otomatis. Q-learning dipilih pada penelitian ini karena algoritma ini cocok untuk di implementasikan ke robot. Gambar 2.2 menunjukkan flowchart dari metode Q-Learning. Penggunaan algoritma Q-learning akan membuat tabelQ meningkat, sehingga proses *learning* membutuhkan waktu yang lama dan kapasitas memori yang besar, maka digunakan Fuzzy logic sebagai generalisasi, sehingga robot bisa bekerja pada *state* dan aksi yang kontinyu.



Gambar 2.1 Arsitektur Navigasi Robot Otomatis [1]



Gambar 2.2 Flowchart Q-Learning [1]

Persamaan nilai Q yang digunakan pada algoritma adalah,

$$Q(s,a) \leftarrow Q(s,a) + \alpha[r + \gamma max_a, Q(s',a') - Q(s,a)] \qquad (2.1)$$

dimana,

$Q(s,a)$ = komponen dari tabel Q (*state, action*)

r = reward

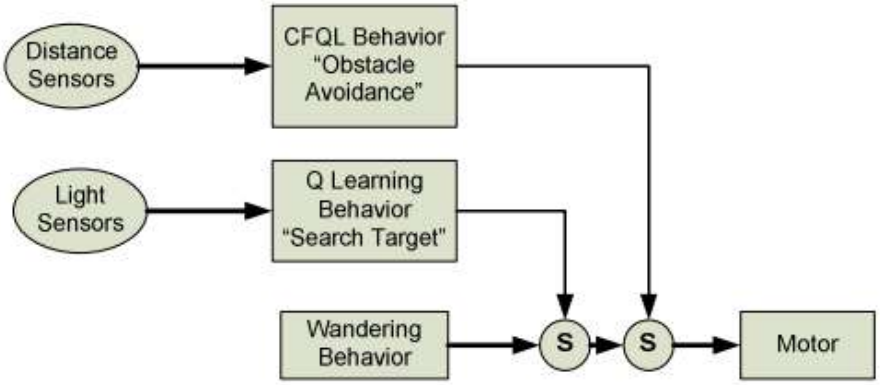
s'= next state

α = *learning rate*

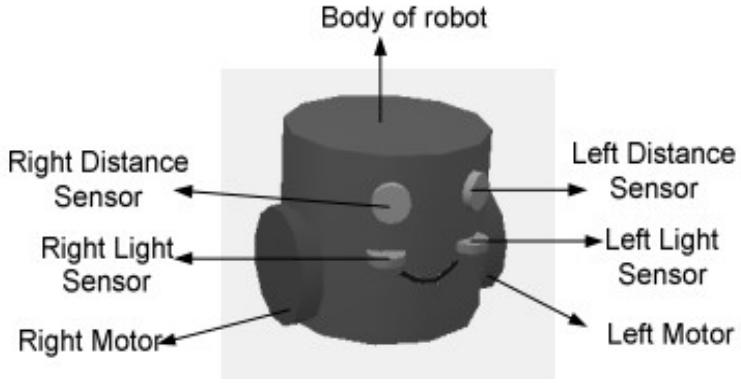
γ = *discount factor*

a'= next action

Pada penelitian ini CFQL hanya digunakan untuk tujuan menghindari *obstacle* karena mencari target memiliki karakteristik yang random. Arsitektur robot menggunakan CFQL ditunjukkan pada Gambar 2.3. Penelitian ini menggunakan dua *premis* Fuzzy yaitu data jarak *obstacle* kanan dan kiri yang terdeteksi oleh sensor dan satu keluaran dengan tiga aksi belok kiri, lurus, belok kanan. Posisi sensor dan mekanik robot ditunjukkan pada Gambar 2.4,

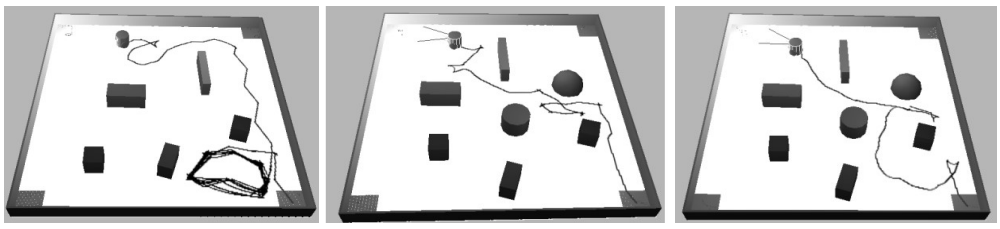


Gambar 2.3 Arsitektur Robot Menggunakan CFQL [1]



Gambar 2.4 Mekanik Robot [1]

Simulasi dilakukan menggunakan tiga metode yang berbeda untuk membandingkan hasil menggunakan Q-Learning, FQL dan CFQL. Menurut paper ini menggunakan metode FQL memiliki performa yang lebih baik dari keduanya, karena memiliki jalur terpendek dan pergerakan yang smooth. Hasil simulasi ditunjukkan pada Gambar 2.5,



(a) (b) (c)

Gambar 2.5 Hasil Simulasi (a) QL, (b) FQL, (c) CFQL [1]

Jika dilihat dari hasil simulasi, aksi pergerakan dari robot masih terlihat kurang, hal ini dikarenakan jumlah sensor yang hanya berjumlah dua buah. Keterbatasan ini yang membuat robot kurang pandai dalam menentukan path yang seharusnya dilalui. Berdasarkan beberapa literature tentang reinforcement learning untuk navigasi mobile robot [2,3] penentuan nilai reward dengan melihat perubahan kondisi dari $S_t \rightarrow S_{t-1}$, tetapi pada panelitian ini pemberian nilai reward hanya berdasarkan S_t.

Pada Tesis ini saya akan mencoba menggunakan 2,3,4 dan 5 sudut region dengan tujuan mencari jumlah sudut region paling baik untuk menjalankan tugas menghindari *obstacle.* Jika ditinjau dari jumlah sudut region, menggunakan sudut region yang banyak akan menghasilkan hasil lebih baik, akan tetapi membutuhkan waktu komputasi yang lebih lama, maka perlu dilakukan penelitian untuk mengetahui berapa jumlah sensor yang paling baik untuk digunakan.

2.1.2 Reinforcement Based Mobile Robot Navigation in Dynamic Environment [2]

Penelitian ini membahas tetang *reinforcement learning* untuk navigasi robot menuju target bola yang bergerak dengan menghindari musuh/*obstacle* yang bergerak. Diasumsikan semua data posisi dan kecepatan *obstacle* maupun robot diketahui secara instant oleh sistem. Lingkungan dari robot terdiri dari *obstacle* dan targetnya, dengan rumusan sebagai berikut,

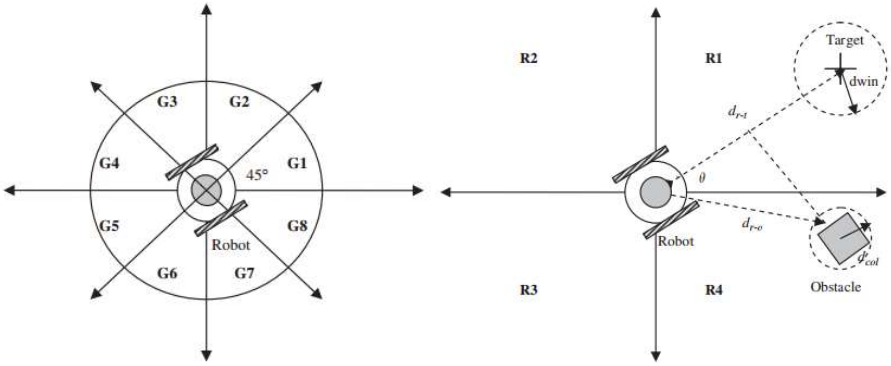
$$S_t = (R_g, R_o, G_n) \tag{2.2}$$

Dimana S_t adalah *state* dari lingkungan saat *t*, R_g adalah region dari target yang digambarkan pada Gambar 2.6(a) dan G_n adalah sudut region *obstacle* yang ditunjukkan pada Gambar 2.6(b). Selanjutnya adalah fungsi *reward* sebagai indikator terhadap aksi robot terhadap *state*. *State* yang terjadi diantaranya, *Safe State* (*SS*)*, Non-Safe State* (*NS*)*, Winning State* (*WS*)*, Failure State* (*FS*). Fungsi *reward* didefinisikan sebagai berikut,

- Bergerak dari *NS* ke *SS*, $r = 1$.
- Bergerak dari *SS* ke *NS*, $r = -1$.
- Bergerak dari *NS* ke *NS* tetapi semakin mendekat ke *obstacle*, $r = -1$.

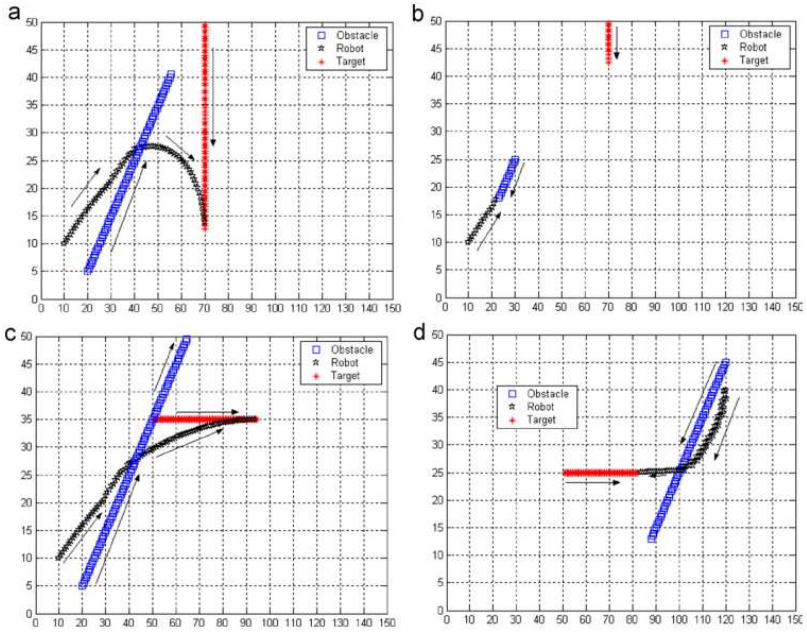
- Bergerak dari *NS* ke *NS* dan semakin menjauh dari *obstacle*, $r = 0$.
- Berhasil mencapai target *WS*, $r = 2$.
- Bertabrakan dengan *obstacle FS*, $r = -2$.

Setelah dilakukan *training,* hasil keputusan yang didapatkan digunakan oleh robot untuk navigasi ke titik target dengan menghindari *obstacle*. Simulasi dibagi menjadi dua fase, fase *training* dan fase *testing*. Fase training dilakukan 4 dengan 4 skenario yang ditunjuukan pada Gambar 2.7,



(a) (b)

Gambar 2.6 (a) Sudut Region Target (b) Sudut Region *Obstacle* [2]



Gambar 2.7 Skenario Fase *Training* [2]

Fungsi *reward* memberikan kecenderungan robot untuk tetap berada pada region yang aman. Pada algoritma Q-Learning fungi nilai dimasukan ke satu table yang disebut Q-Table, dapat dituliskan sebagai berikut,

$$Q(s,a) \leftarrow Q(s,a) + \alpha[r + \gamma max_a, Q(s',a') - Q(s,a)] \qquad (2.3)$$

dimana,

S_t : *State* pada waktu *t*

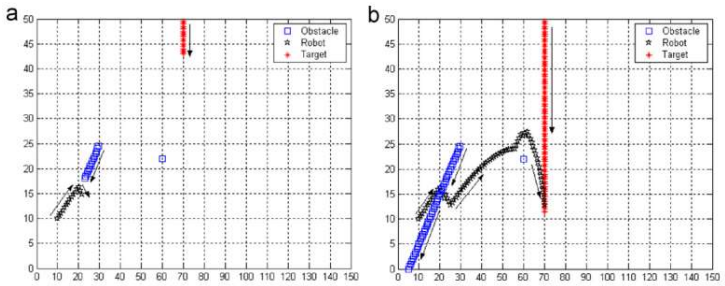
$Max(Q, S_{t-1})$: nilai maksimal Q-value

a_t : aksi robot pada waktu *t*

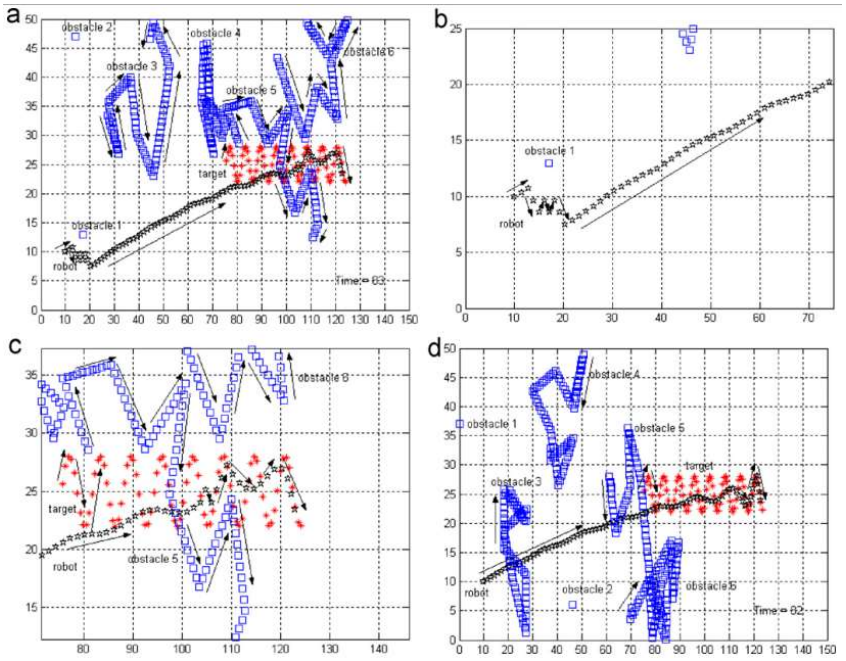
γ : *discount factor*

Jika dilihat pada Gambar 2.7, objek merah adalah target bola, objek biru adalah *obstacle* dan objek hitam adalah robot. Robot tidak berhasil mencapai target untuk skenario 2. Fase *training* ini dilakukan untuk membentuk Q-Table seperti yang dijelaskan sebelumnya. Selanjutnya fase *testing* dilakukan dengan kondisi *obstacle* yang banyak baik yang diam maupun bergerak. Fase testing disimulasikan dua kali, hasil simulasi fase *testing* ditunjukkan pada Gambar 2.8 dan Gambar 2.9.

Skenario pertama hampir sama dengan skenario fase *training* pertama dengan menambahkan satu *obstacle* yang diam. Terlihat bahwa robot bisa mencapai target, hal ini menunjukkan bahwa fase training yang dilakukan bisa dibilang berhasil. Robot tidak melakukan hal yang sama dilakukan pada saat *training*. Skenario kedua menggunakan 2 *obstacle* yang diam dan 4 *obstacle* yang bergerak secara acak. Robot terlihat berhasil mengejar target yang bergerak sinusoidal. Pada paper ini dijelaskan secara detail proses Q-Learning dilakukan yang selanjutnya akan dipakai untuk menunjang penelitian thesis dengan menambahkan konsep Fuzzy.



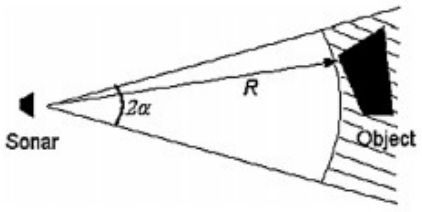
Gambar 2.8 Hasil Simulasi Fase Testing Skenario Pertama [2]



Gambar 2.9 Hasil Simulasi Fase Testing Skenario Kedua [2]

2.1.3 Microcontroller-based Mobile Robot Positioning and Obstacle Avoidance [3]

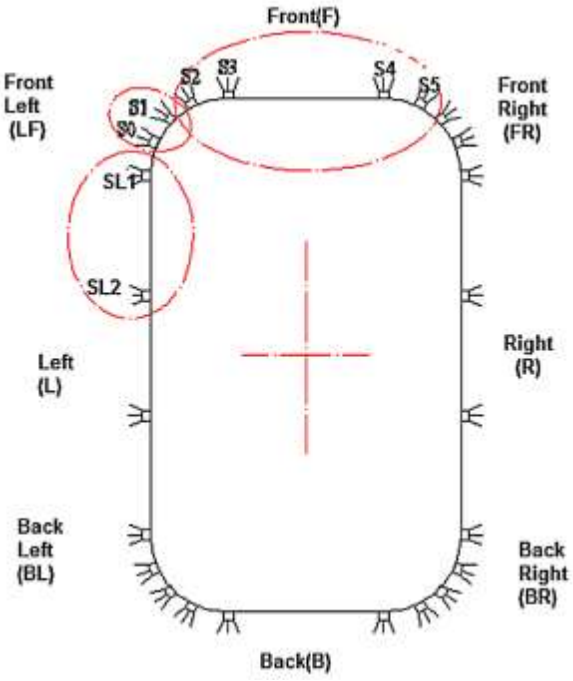
Tujuan dari paper ini adalah implementasi robot untuk bernavigasi di lingkungan pabrik untuk melakukan tugas pengantar barang dari gudang ke ruang-ruang produksi di wilayah pabrik. Navigasi robot memiliki makna kemampuan untuk memahami lingkungan tanpa bertabrakan dengan halangan, kemampuan untuk mengetahui posisi sendiri, dan kemampuan untuk mencapai suatu lokasi tujuan. Pada paper ini sistem navigasi di buat pada mobile robot yang beroperasi di suatu pabrik didalam dan luar gudang. Odometry digunakan untuk mengetahui posisi robot dengan masukan sensor rotary encoder pada robot. Odometry diketahui bisa menghasilkan akurasi yang cukup baik akan tetapi memiliki beberapa kelemahan diantaranya terjadinya slip roda dan membutuhkan waktu sampling yang cepat. Untuk mengkoreksi posisi robot dari hasil odometry dan menghindari halangan digunakan sensor ultrasonic pada robot. Gambar 2.10 merupakan sudut beam dari sensor yang digunakan.



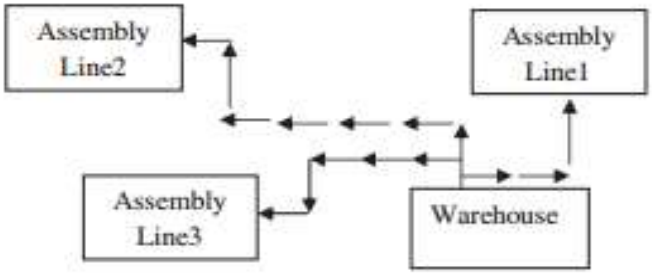
Gambar 2.10 Sudur dari sensor ultrasonic (α), respon sensor (R) [3]

Metode odometry digunakan untuk mendeteksi putaran dari masing2 roda untuk memperkirakan posisi dari robot menggunakan kinematika dari robot. Penggunaan metode ini bisa menyebabkan robot mengalami kesalahan yang semakin lama semakin besar dalam memprediksi posisinya, karena beberapa hal yang telah disebutkan sebelumnya, oleh karena itu pada paper ini digunakan suatu titik referensi yang digunakan secara periodik untuk mereduksi kesalahan possisi.

Pada saat mendesain sistem kontrol untuk robot otomatis yang menjadi permasalahan utama adalah menghindari halangan yang tidak bisa diprediksi sebelumnya. Gambar 2.11 menunjukkan penempatan sensor pada robot dan pengelompokannya.



Gambar 2.11 Penempatan Sensor [3]



Gambar 2.12 Path Robot ketika Bernavigasi [3]

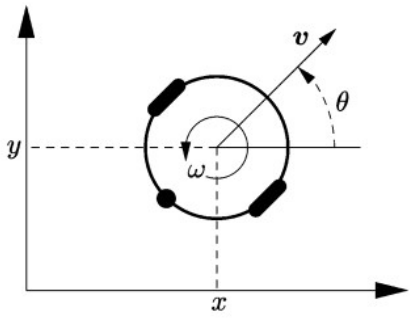
Sensor yang sebenarnya berjumlah 24 sensor dikelompokkan menjadi 8 sensor. Satu kelompok dari sensor digunakan sebagai satu masukan sensor dengan menganggap pembacaan sensor paling dekat sebagai acuan dan yang lain tidak diperhitungkan. Pada paper ini robot bernavigasi pada tiga lingkungan diantaranya, didalam gudang, diluar gudang dan bermanuver di pintu gudang. Robot memiliki kemampuan untuk mendeteksi di lingkungan mana dia berada dengan memperhatikan data sensor. Gambar 2.12 adalah path yang dilalui robot diluar gudang.

Pada paper ini menggunakan jarak terdekat dari satu sensor di dalam satu grup sensor, hal ini membuat robot kurang baik dalam menentukan posisi obstacle. Pada penelitian saya akan menggunakan nilai rata2 dari masing-masing sensor dalam grup dan menggunakannya sebagai masukan ke sistem fuzzy.

2.2 Dasar Teori

2.2.1 Model Kinematika Robot

Model kinematika adalah pembelajaran dari pergerakan sistem mekanik tanpa memperhitungkan gaya yang yang berefek pada pergerakan robot[4]. Secara umum, kita bisa memetakan posisi robot nonholonomic seperti yang terlihat pada Gambar 2.13. Dimana x dan y adalah titik koordinat robot secara horizontal dan vertikal, θ adalah sudut hadap robot. Kita bisa mengetahui dimana posisi robot saat t dengan masukan kecepatan translasi robot(v) dan kecepatan sudut robot(ω).



Gambar 2.13 Sketsa robot tampak atas dengan varibelnya [3]

Persamaan kinematika dari robot bisa diasumsikan sebagai berikut[4]:

$$\dot{x} = v.cos\theta \tag{2.3}$$

$$\dot{y} = v.sin\theta \tag{2.4}$$

$$\dot{\theta} = \omega \tag{2.5}$$

Kecepatan dari robot didapatkan dari kecepatan translasi dari roda kiri(v_l) dan roda kanan(v_r) dengan rumusan berikut[4],

$$v_r = R\dot{\varphi}_r \tag{2.6}$$

$$v_l = R\dot{\varphi}_r \tag{2.7}$$

$$v = \frac{v_r + v_l}{2} = R\frac{(\dot{\varphi}_r + \dot{\varphi}_l)}{2} \tag{2.8}$$

Dimana $\dot{\varphi}_r$ adalah kecepatan putar motor kanan dan $\dot{\varphi}_l$ adalah kecepatan putar motor kiri dalam rad/s. Selanjutnya untuk mencari kecepatan sudut digunakan jarak antara roda penggerak dengan rumusan sebagai berikut[4],

$$\omega = \frac{v_r - v_l}{h} = R\frac{(\dot{\varphi}_r - \dot{\varphi}_l)}{h} \tag{2.9}$$

Dari rumusan diatas kita bisa mengetahui posisi robot dari perputaran masing-masing roda. Jika dibentuk dalam matrik sebagai berikut[4],

$$\begin{bmatrix}\dot{x}\\ \dot{y}\\ \dot{\theta}\end{bmatrix} = \begin{bmatrix}\frac{R}{2}cos\theta & \frac{R}{2}cos\theta\\ \frac{R}{2}sin\theta & \frac{R}{2}sin\theta\\ \frac{R}{h} & -\frac{R}{h}\end{bmatrix}\begin{bmatrix}\dot{\varphi}_r\\ \dot{\varphi}_l\end{bmatrix} \tag{2.10}$$

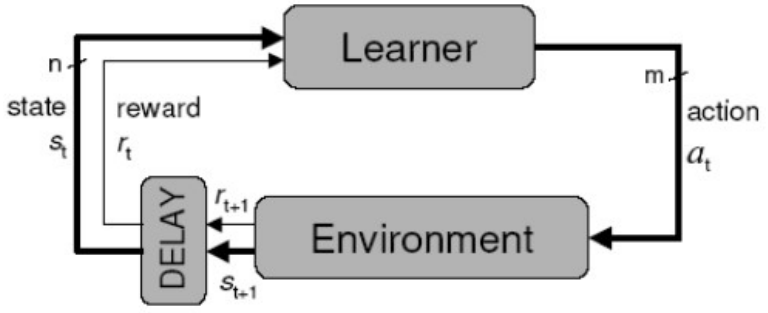
2.2.2 Reinforcement Learning

Reinforcement learning adalah pendekatan belajar dengan *trial and error* untuk mencapai tujuan. Reinforcement Learning termasuk dalam *unsupervisory learning,* sehingga tidak perlu ada bimbingan untuk melakukan tugasnya. Metode *learning* ini lebih cocok untuk diterapkan ke suatu sistem yang *online* dimana robot akan melakukan *learning* dan aksi secara bersamaan [6]. Skema dasar Reinforcement Learning ditunjukkan pada Gambar 2.14.

Jika dilihat dari skema, *Learner/Agent* mendapatkan informasi *state* lingkungan, selanjutnya akan melakukan aksi untuk mencapai suatu target posisi atau menghindari *obstacle*. Dari keadaan lingkungan yang dilewatinya, *agent* akan mendapatkan *reward* positif atau negative bergantung pada state yang diperoleh setelah melakukan aksi.

2.2.3 Q Learning

Algoritma Q-Learning adalah salah satu metode yang popular diterapkan untuk melakukan *reinforcement learning* [1]. Untuk menggunakan algoritma Q-Learning empat hal yang harus dilakukan terlebih dahulu ditentukan diataranya: lingkup kerja lingkungan, fungsi *reward*, nilai fungsi, keputusan yang diadaptasi ke robot [2]. Semua data disimpan didalam tabel dengan baris perbedaan *state* yang dilalui oleh *agent* pada saat *learning* dan kolom yang berisi aksi yang dilakukan oleh robot. Tabel tempat menyimpan data disebut dengan *Q-Table*, setelah menjalani fase training tabel akan terisi oleh *Q-Value* untuk masing-masing *state* yang diberikan. *Q-Value* berisi bobot atas aksi *agent* terhadap suatu *state*. Algoritma



Gambar 2.14 Skema Dasar Reinforcement Learning [1]

dari *Q-Learning* dinyatakan dalam uraian berikut,

1. Inisialisasi $Q(s, a)$.
2. Mencari kondisi state saat ini.
3. Lakukan aksi berdasarkan state berdasarkan nilai *Q-Table*.
4. Mendapatkan fungsi reinforcement (reward/punishment) dari hasil aksi yang dilakukan.
5. Perbarui fungsi $Q(s, a)$ dengan rumus,

   $$\hat{Q}(s_t, a_t) = Q(s_t, a_t) + \alpha[r_{t+1} + \gamma V(S_{t+1}) - Q(s_t, a_t)] \qquad (2.11)$$

   Dengan $V(S_{t+1}) = \max Q(s_{t+1}, a_{t+1})$

6. Ulangi langkah 2 sampai didapatkan *Q-Table* yang optimal.

2.2.4 Fuzzy

Ada beberapa kaidah yang harus dipenuhi untuk sistem Fuzzy agar hasil desain dapat memenuhi objektif kontrol yang digunakan diantaranya [9],

- **Variabel linguistic**, Untuk menggambarkan input (x_i) dan output (u_i) sistem fuzzy, Contoh: x_1 = "posisi", x_2 = "kecepatan" dan u_1 = "tegangan"
- **Aturan Linguistik**, Pemetaan input ke output dalam sistem fuzzy, aturan yang terdiri dari kondisi → aksi. ***IF*** premis ***Then*** konsekuen ***If*** x_1 is A_1 and x_2 is B_1 ***Then*** u is C_1.
- **Himpunan Fuzzy, Operator Fuzzy, Inferensi Fuzzy**.

Arsitektur dari kontroler fuzzy memiliki 4 komponen diantaranya [9],

- **Rule Base**, berisi pengetahuan dalam bentuk beberapa aturan, untuk tujuan mendapatkan sistem kontrol yang baik.
- **Mekanisme Inferensi Fuzzy**, Mengevaluasi rule mana yang sesuai dengan kondisi saat *t,* dan menentukan seperti apa keluaran ke plant.
- **Fuzzification**, Memodifikasi masukan sehingga bisa mudah digunakan pada *rule base*.

- **Defuzzifikasi**, mengubah hasil keputusan yang dicapai oleh mekanisme inferensi menjadi masukan untuk ke *plant.*

2.2.5 Fuzzy Q-Learning

Dari beberapa literature [1] [7-8] dalam penerapan Fuzzy Q-Learning (FQL) untuk suatu sistem berbeda-beda jika dilihat dari prosesnya, tetapi pada intinya FQL adalah ekstensi dari metode Q-Learning yang dibawah ke lingkungan Fuzzy [7]. Hal ini dilakukan karena ketika Q learning bekerja dengan state dan aksi yang kontinyu *Q-Table* akan membesar, sehingga memerlukan waktu yang sangat lama dalam proses pembelajarannya. Untuk mempresentasikan aksi pada *Fuzzy Inference System* diperlukan modifikasi pada *rule base,* bentuk rulenya adalah,

Jika S_i maka **aksi** = $a_i, a_i \in A_i$.

Jika S_i maka **aksi** = a_i, maka nilai $q = q(S_i, a_i)$

Dengan State S_i didefinisikan dengan "x_1 adalah $S_{i,1}$ **dan** x_2 adalah $\boldsymbol{S_{i,2}}$ **dan** … x_n adalah $S_{n,1}$", $(S_{i,j})_{j=1}^{n}$ adalah label *fuzzy* dan a_i adalah himpunan aksi yang mungkin yang dapat dipilih pada *state* S_i. Dengan menggunakan FIS Takagi-Sugeno dengan *N* Rule, aksi $a(x)$ untuk vector masukan $x(x_1, \dots, x_n)$ dan konsekuen rule $(a_i)_{i=1}^{N}$ adalah:

$$a(x) = \frac{\sum_{i=1}^{N} a_i(x) \times a_i}{\sum_{i=1}^{N} a_i(x)} \tag{2.12}$$

Dan nilaiQ yang bersesuaian adalah:

$$Q(x, a) = \frac{\sum_{i=1}^{N} a_i(x) \times q(S_i, a_i)}{\sum_{i=1}^{N} a_i(x)} \tag{2.13}$$

Fungsi $x \rightarrow a_i(x)$ mewakili nilai sebenarnya dari *rule i* yang diberikan vector masukan x. Jika $a[i, j]$ adalah aksi yang mungkin ke-*j* pada rule *i* dan $q[i, j]$ adalah nilai yang bersesuaian, maka *rule* untuk FIS dapat disederhanakan menjadi,

Jika x adalah S_i **maka** $a[i, 1]$ **dengan** $q[i, 1]$ **atau** $a[i, 2]$ **dengan** $q[i, 2]$… **atau** $a[i, j]$ **dengan** $q[i, j]$.

Robot diharuskan mampu menemukan kesimpulan terbaik untuk tiap *rule*, yaitu aksi yang memberikan nilai q terbaik.

Untuk menjelajahi semua himpunan aksi yang mungkin dan mendapatkan pengalaman melalui sinyal *reinforcement*, aksi local dipilih menggunakan strategi *Exploration/Exploitation Policy* (EEP) yang didasarkan pada kualitas pasangan state-aksi yaitu nilai Q. Pada thesis ini digunakan metode $\varepsilon - greedy$ sederhana untuk memilih aksi, yaitu aksi *greedy* dengan probabilitas $1 - \varepsilon$ dan aksi acak dipilih dengan probabilitas ε. Probabilitas eksplorasi ditujukan untu mengendalikan keperluan antara eksplorasi dan eksploitasi. Untuk nilai i^* sehingga $q[i, i^*] = max_{j \leq i} q[i, j]$, aksi globalnya adalah:

$$a(x) = \frac{\sum_{i=1}^{N} a_i(x) \times q(i, i^0)}{\sum_{i=1}^{N} a_i(x)} \tag{2.14}$$

Dengan i^* adalah *i* pada saat *q* maksimal dan i^o adalah *i* pada saat *q* optimal, dengan nilai Q actual dari aksi yang disimpulkan *a* adalah:

$$Q(x, a) = \frac{\sum_{i=1}^{N} a_i(x) \times q(i, i^*)}{\sum_{i=1}^{N} a_i(x)} \tag{2.15}$$

Dan nilai *state x*:

$$V(x) = \frac{\sum_{i=1}^{N} a_i(x) \times q(i, i^*)}{\sum_{i=1}^{N} a_i(x)} \tag{2.16}$$

Berdasarkan penjelasan sebelumnya, proses algoritma Fuzzy Q-Learning adalah sebagai berikut,

1. Dapatkan state x
2. Untuk tiap *rule,* pilih konsekuen actual menggunakan seleksi $\varepsilon - greedy$.
3. Hitung konsekuen global dan q global.
4. Terapkan aksi global a dan dapatkan state baru
5. Dapatkan fungsi *reinforcement r*
6. Perbarui nilai *q*

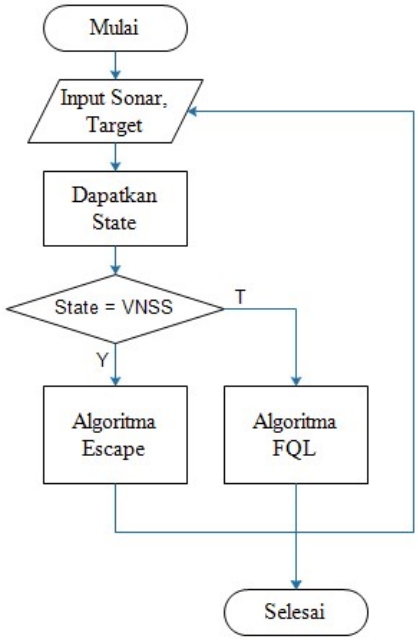
BAB 3
METODOLOGI PENELITIAN

Bab ini membahas tentang tahapan penelitian yang akan dibuat. Pada Gambar 3.1 adalah robot Scitos G5 yang digunakan pada penelitian ini. Robot Scitos G5 memiliki dua roda pengerak dengan jari-jari roda (R) = 20 cm dan jarak antara kedua roda (h) = 40 cm [6]. Robot Scitos G5 memiliki 24 sensor sonar yang mengelilingi robot dan 2 sensor rotary encoder.

Penggunaan metode FQL pada paper (1) digunakan untuk tujuan menghindari obstacle saja sehingga robot tidak bisa bergerak menuju target bersamaan dengan menghindari obstacle dikarenakan masalah rule yang terlalu banyak untuk dilakukan proses *learning*. Sesuai dengan tujuan dari penelitian untuk meningkatkan performa dikakukan dengan cara berikut,

1. Menambahkan sensor pada robot dengan tujuan,
    a. Pada proses *learning* robot bisa mengetahui posisi dari *obstacle* dengan sudut region yang lebih banyak. Hal ini memungkinkan robot untuk lebih baik dalam menentukan aksi.
    b. Pada proses *fuzzyfikasi* akan dilakukan modifikasi premis fuzzi yang bertujuan agar robot bisa menghindari halangan dan bergerak menuju posisi target secara bersamaan.

Gambar 3.1 Robot Scitos G5 [6]



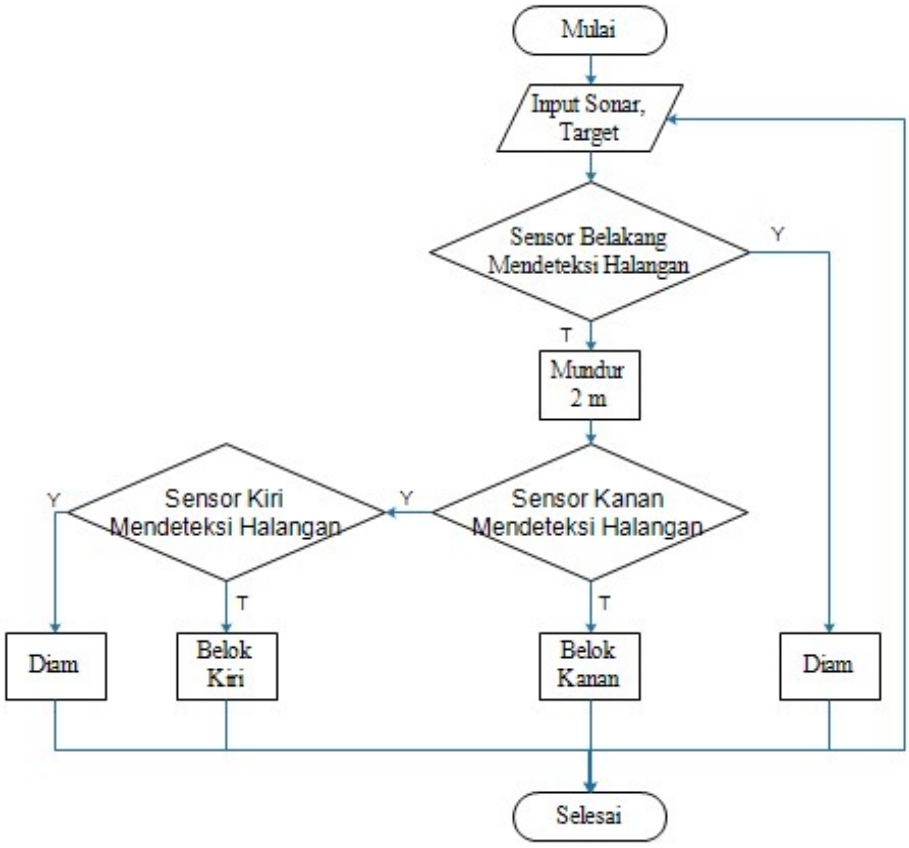
Gambar 3.2 Flowchart Program Utama

2. Mengubah penalaan fungsi reward dengan melihat perubahan state. Dari beberapa literature (2,11) pada penerapan reinforcement learning untuk tujuan obstacle avoidance fungsi reward didefinisikan dengan melihat perubahan state yang terjadi.

Pada thesis ini menggunakan dua algoritma yaitu algoritma untuk melarikan diri dari kondisi ketika semua sensor mendeteksi adanya obstacle yang terlalu dekat yang selanjutnya akan disebut sebagai *Very Non-Safe State* (*VNSS*) dimana robot akan melakukan program mengamankan state (*escape)*. Proses *switch* algoritma dilakukan dengan melihat kondisi lingkungan robot. Pada Gambar 3.2 ditunjukkan gambar algoritma pemrograman proses *switch* algoritma.

3.1 Perancangan Algoritma Escape

Algoritma escape digunakan pada saat *state* robot VNSS yang menyebabkan untuk bergerak maju, belok kanan dan belok kiri tidak bisa dilakukan, sehingga robot akan melakukan aksi tanpa memperhitungkan dimana target berada. Dengan menggunakan acuan dari sensor sonar, robot akan melakukan aksi khusus untuk bergerak mundur, belok kanan, belok kiri atau diam. Flowchart program ditunjukkan pada Gambar 3.3.



Gambar 3.3 Flowchart Program *Escape*

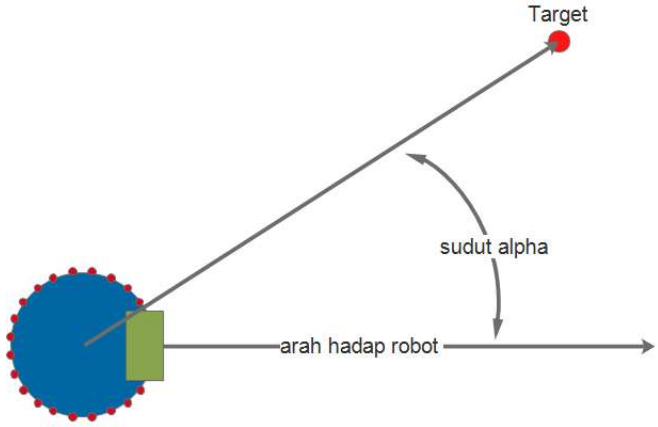
3.2 Perancangan Premis dan Keluaran Fuzzy

Algoritma FQL digunakan untuk menghindari halangan dan menuju target secara bersamaan. Dengan 2 masukan dan satu keluaran fuzzy.

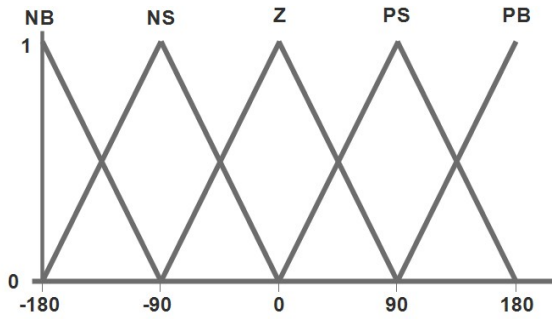
3.2.1 Premis Fuzzy

3.2.1.1 Posisi Target

Posisi target ditentukan mengunakan perbedaan sudut antara arah hadap robot dengan target yang selanjutnya akan disebut dengan sudut alpha (α). Posisi dari robot (X_r, Y_r, θ_r) dan posisi target (X_g, Y_g) diketahui oleh robot. Sudut α diilustrasikan pada Gambar 3.4 dengan *membership function* yang ditunjukkan pada Gambar 3.5



Gambar 3.4 Sudut Arah Hadap Robot Terhadap Target



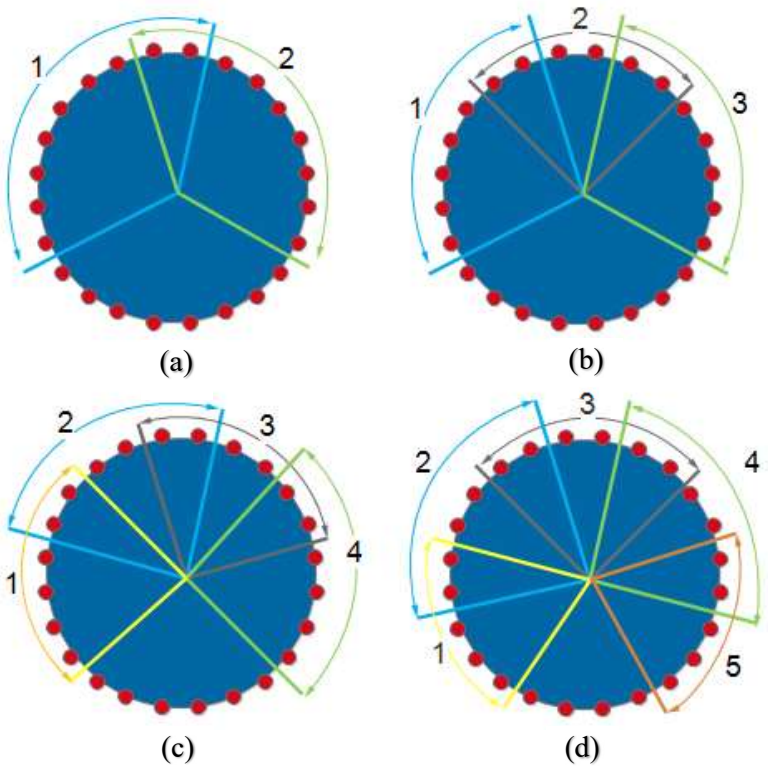
Gambar 3.5 Membership Function Sudut Target Terhadap Robot

nilai α didapatkan dengan rumusan sebagai berikut [2],

$$\alpha = arctan \frac{(Y_g - Y_r)}{(X_g - X_r)} - \theta_r \quad (3.1)$$

3.2.1.2 Posisi dan Jarak Halangan

Pada penelitian ini akan mmenggabungkan data posisi dan jarak menjadi satu himpunan *premis fuzzy* dengan menjadikan jarak sebagai acuan besaran bobot dan sudut sebagai membership functionnya. Robot Scitos G5 memiliki 24 sensor yang mengelilingi robot, maka dilakukan pengelompokan sensor menjadi 2,3,4 dan 5 region yang selanjutnya akan diuji. Pengelompokan region dari sensor ditunjukkan pada Gambar 3.6.



Gambar 3.6 Ilustrasi Pengelompokan Sensor Sonar pada Robot (a) 2 Region, (b) 3 Region, (c) 4 Region, (d) 5 Region

untuk menghitung posisi sensor pada keliling robot digunakan rumus,

$$X_i = X_r + (d_r cos\alpha_i) \quad (3.2)$$

$$Y_i = Y_r + (d_r sin\alpha_i) \quad (3.3)$$

dimana,

X_i = posisi sensor ke-*i* pada sumbu *x*

Y_r = posisi sensor ke-*i* pada sumbu *y*

Y_i = posisi sensor ke-*i* pada sumbu

d_r = jari-jari robot

X_r = posisi robot pada sumbu *x*

α_i = sudut sensor ke-*i* terhadap titik pusat robot

Area yang terdeteksi oleh sensor berupa segitiga dengan titik $((X_A, Y_A), (X_B, Y_B), (X_C, Y_C))$. Untuk menghitung area segitiga yang terdeteksi dari sensor digunakan rumus,

$$X_A^i = X_i \tag{3.4}$$

$$Y_A^i = Y_i \tag{3.5}$$

$$X_B^i = X_i + (S_{max} cos(\alpha_i - 7.5)) \tag{3.6}$$

$$Y_B^i = Y_i + (S_{max} sin(\alpha_i - 7.5)) \tag{3.7}$$

$$X_C^i = X_i + (S_{max} cos(\alpha_i + 7.5)) \tag{3.8}$$

$$Y_C^i = Y_i + (S_{max} sin(\alpha_i + 7.5)) \tag{3.9}$$

dimana,

X_P^i = posisi titik P sensor ke-*i* pada sumbu *x*

S_{max} = maksimum jarak sensor (*2 m*)

Y_P^i = posisi titik P sensor ke-*i* pada sumbu *y*

Sensor memiliki sudut beam angle 30° yang diilustrasikan pada Gambar 3.7. Sensor akan mendeteksi halangan jika halangan berada pada area beam sensor dengan jarak kurang dari 2 meter, dengan rumusan sebagai berikut,

$$S_i^m = \frac{d_i^m}{2} \tag{3.10}$$

Untuk menghitung jarak dari masing- masing region akan diuji menggunakan dua cara yaitu, jarak terdekat salah satu sensor dalam region tersebut dan rata-rata dari semua sensor di dalam region tersebut. Untuk menghitung rata-rata dengan rumusan sebagai berikut,

$$\mu_s(m) = \begin{cases} 1 - \dfrac{\sum_{i=1}^{n} S_i^m}{n} \ \forall \ \dfrac{\sum_{i=1}^{n} S_i^m}{n} \leq 1 \\ 0 \ \forall \dfrac{\sum_{i=1}^{n} S_i^m}{n} > 1 \end{cases} \tag{3.11}$$

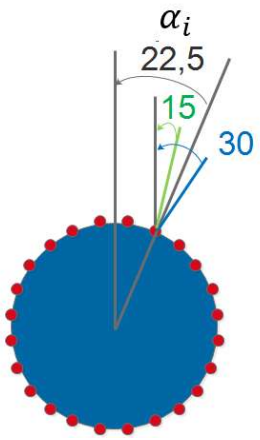
dimana,

$\mu_s(m)$ = premis sensor region ke-*m*

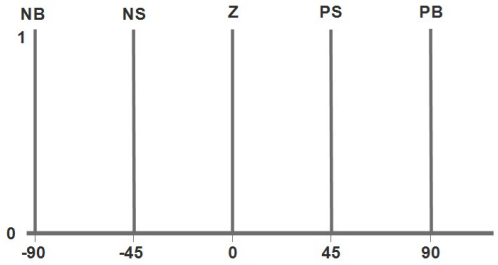
m = region ke-*m*

n = jumlah sensor dalam region ke-*m*

S_i^m = jarak halangan ke sensor ke-*i* dalam region ke-*m*



Gambar 3.7 Sudut sensor beam angle 30°



Gambar 3.8 Membership Function Output

3.2.2 Keluaran Fuzzy

Pada penelitian ini point tracking digunakan untuk menuju target berdasarkan nilai α. Keluaran dari *fuzzy* (u^t) memberikan bobot tambahan untuk nilai α, sehingga robot bisa menghindari halangan dan menuju ke target. *Membership function* untuk keluaran fuzzy digambarkan pada Gambar 3.8 berupa *singleton*.

3.3 Perancangan Kontroller Untuk Mencapai Target

Point target sudah ditentukan dari awal dan tidak berubah. Selanjutnya robot akan menggunakan data sensor sonar untuk menentukan state robot saat *t.* Robot akan bergerak menuju target menggunakan heading error seperti yang telah dijelaskan pada subbab 3.2. Robot akan berhenti apabila telah mencapai titik target.

Untuk mencapai target digunakan sensor rotary encoder dengan menghitung pulsa yang didapatkan dalam satu waktu *t* (*P*). Diagram blok proses

untuk mencapai target ditunjukkan pada Gambar 3.3. Masukan ke sistem berupa target (X_g, Y_g), masukan tersebut akan dihitung dengan umpan balik posisi robot (X_r, Y_r, θ_r) untuk mendapapatkan nilai heading error (α) dan jarak robot ke titik target (D_g). Persamaan Untuk aktuator berupa motor DC diberikan sebagai berikut [4]:

$$v_a = R_a i_a + L_a \frac{di_a}{dt} + e_a \tag{3.12}$$

$$e_a = K_b \omega_m \tag{3.13}$$

$$\tau_m = K_t i_a \tag{3.14}$$

$$\tau = N t_m \tag{3.15}$$

dimana,

i_a = arus jangkar

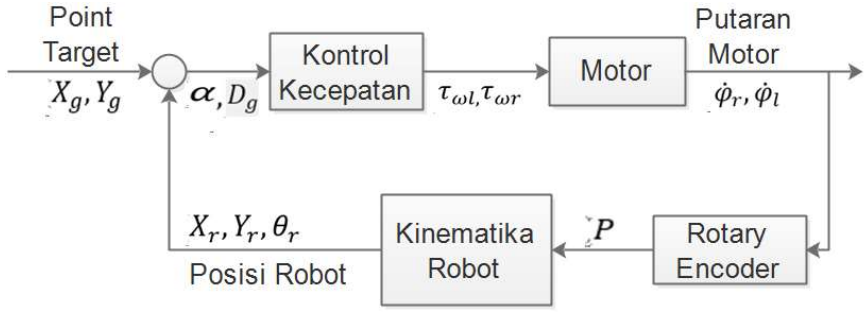
(R_a, L_a) = resistansi dan induktansi kumparan jangkar

(K_t, K_b) = konstanta torsi dan gaya gerak listrik

ω_m = kecepatan angular rotor

N = gear ratio

Sensor rotary encoder digunakan untuk mendeteksi berapa kecepatan sudut motor dengan menghitung banyak pulsa yang dihasilkan pada saat *t*. Dengan diketahui jumlah pulsa, robot bisa mengetahui posisi nya sekarang dengan memasukkan persamaan kinematika robot. Diagram blok pengaturan kecepatan motor tanpa fuzzy ditunjukkan pada Gambar 3.9.



Gambar 3.9 Diagram Blok Pengaturan Motor Tanpa Fuzzy

3.3.1 Pengaturan Kecepatan Motor Untuk Menuju Target

Untuk mencapai target (X_g, Y_g) robot menghitung α dan jarak robot ke titik target (D_g) yang diilustrasikan pada Gambar 3.4. Nilai D_g didapatkan dengan rumusan sebagai berikut [2],

$$D_g = \sqrt{(X_g - X_r)^2 + (Y_g - Y_r)^2} \tag{3.16}$$

Keluaran dari *fuzzy* akan menentukan nilai kecepatan *angular* (ω) robot sedangkan v dibuat tetap sebesar 1.1 m/s. Jika keluaran *fuzzy* (u^t) kurang dari 0 maka robot akan bergerak kearah kiri dari jalur *point tracking*, demikian juga sebaliknya. Jika $u^t = 0$ robot akan bergerak menuju target berdasarkan nilai point trackingnya. Diagram blok pengaturan kecepatan motor yang ditambahkan keluaran dari *fuzzy* digambarkan pada Gambar 3.10,

Dengan mengetahui nilai α dan u^t kemudian ditentukan kecepatan *angular* (ω) dari robot, dengan rumusan sebagai berikut,

$$\omega = \alpha u^t \tag{3.17}$$

dengan nilai *v* yang konstan bernilai 1.1 m/s, menggunakan rumus 2.9 dan 2.10 didapatkan,

$$\begin{bmatrix} v \\ \omega \end{bmatrix} = \begin{bmatrix} \mathrm{R}/2 & R/2 \\ R/h & -R/h \end{bmatrix} \begin{bmatrix} \dot{\varphi}_r \\ \dot{\varphi}_l \end{bmatrix} \tag{3.17}$$

$$\begin{bmatrix} \dot{\varphi}_r \\ \dot{\varphi}_l \end{bmatrix} = \begin{bmatrix} \mathrm{R}/2 & R/2 \\ R/h & -R/h \end{bmatrix}^{-1} \begin{bmatrix} \mathrm{v} \\ \omega \end{bmatrix} = \begin{bmatrix} \mathrm{R}/2 & R/2 \\ R/h & -R/h \end{bmatrix}^{-1} \begin{bmatrix} 1.1 \\ \alpha u^t \end{bmatrix} \tag{3.18}$$

dengan nilai *R = 0.2 m* dan h = *0.4 m* didapatkan,

$$\begin{bmatrix} \dot{\varphi}_r \\ \dot{\varphi}_l \end{bmatrix} = \begin{bmatrix} 0.1 & 0.1 \\ 0.5 & -0.5 \end{bmatrix}^{-1} \begin{bmatrix} 1.1 \\ \alpha u^t \end{bmatrix} \tag{3.19}$$

dimana,

R = jari-jari roda

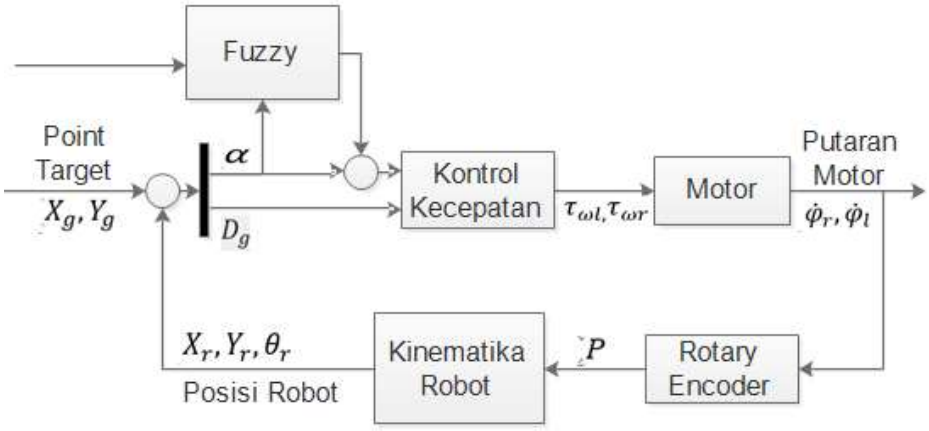
$\dot{\varphi}_l$ = kecepatan putar motor kiri

h = jarak antara roda

v = kecepatan translasi robot

$\dot{\varphi}_r$ = kecepatan putar motor kanan

ω = kecepatan *angular* robot



Gambar 3.10 Diagram Blok Pengaturan Kecepatan Motor Dengan Penambahan *Fuzzy*

3.3.2 Pemetaan Posisi Robot

Dengan menggunakan kinematika robot pada rumus (2.11) peneliti bisa mendapatkan nilai *v* dan *ω* robot SCITOS G5 dari kecepatan angular masing-masing motor, dengan rumusan berikut [4],

$$\begin{bmatrix}\dot{x}\\ \dot{y}\\ \dot{\theta}\end{bmatrix} = \begin{bmatrix}\frac{R}{2}cos\theta & \frac{R}{2}cos\theta\\ \frac{R}{2}sin\theta & \frac{R}{2}sin\theta\\ \frac{R}{h} & -\frac{R}{h}\end{bmatrix}\begin{bmatrix}\dot{\varphi}_r\\ \dot{\varphi}_l\end{bmatrix} = \begin{bmatrix}0.1cos\theta & 0.1cos\theta\\ 0.1sin\theta & 0.1sin\theta\\ 0.5 & -0.5\end{bmatrix}\begin{bmatrix}\dot{\varphi}_r\\ \dot{\varphi}_l\end{bmatrix} \quad (3.20)$$

Robot Scitos G5 telah dilengkapi dengan sensor rotary encoder dengan resolusi 460 *tick/rad* [5] untuk mendeteksi banyaknya putaran disetiap sampling program. Keluaran dari sensor rotary berupa counter pulsa (P) dengan nilai $R = 0.2$ dan $h = 0.4$, didapatkan,

$$\begin{bmatrix}\dot{x}\\ \dot{y}\\ \dot{\theta}\end{bmatrix} = \begin{bmatrix}0.1cos\theta & 0.1cos\theta\\ 0.1sin\theta & 0.1sin\theta\\ 0.5 & -0.5\end{bmatrix}\begin{bmatrix}P_r/460\\ P_l/460\end{bmatrix} \quad (3.21)$$

Diasumsikan robot sebelum bergerak berada pada posisi $X_r(0) = 0, Y_r(0) = 0$ dan $\theta_r(0) = 0$. Posisi dari robot saat t didapatkan dengan rumusan sebagai berikut,

$$X_r(t) = X_r(t-1) + \dot{x}(t) \quad (3.22)$$

$$Y_r(t) = Y_r(t-1) + \dot{y}(t) \quad (3.23)$$

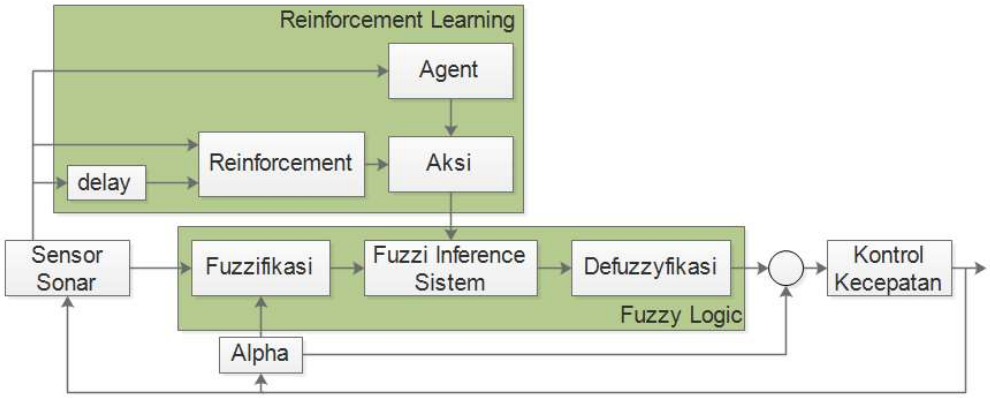
$$\theta_r(t) = \theta_r(t-1) + \dot{\theta}(t) \quad (3.24)$$

Dari rumusan diatas kita bisa mengetahui posisi robot dari perputaran masing-masing roda. Jadi rumusan diatas dipakai pada penelitian ini dengan masukan pulsa rotary kiri dan kanan(P_l, P_r), keluaran posisi dari robot (X_r, Y_r, θ_r).

3.4 Perancangan Algoritma Q Learning Untuk Menghindari Obstacle

Konsep yang diajukan pada penelitian adalah penggunaan metode Fuzzy Q-Learning untuk tujuan menghindari obstacle yang ditunjukkan pada Gambar 3.10, dimana learning dilakukan untuk mengubah bobot pada Fuzzy Inference System. Keluaran dari *Fuzzy* adalah sudut hindar yang kemudian akan diterjemahkan oleh *velocities module* menjadi kecepatan masing-masing motor.

Untuk menggunakan Fuzzy Q-Learning perlu didefinisikan state dan reward poin, pada penelitian ini Fuzzy Q-Learning digunakan ketika robot mendeteksi adanya obstacle yang beresiko bertabrakan dengan robot ketika bergerak menuju target. Untuk menuju ke target menggunakan heading error seperti yang dijelaskan sebelumnya. Sensor sonar pada robot SCITOS G5 memiliki range 0mm-3000mm, dengan nilai u_i adalah masukan nilai sensor ke i yang ditunjukkan pada Gambar 3.6 dengan lingkaran bewarna merah. *State* yang terjadi diantaranya, *Safe State (SS), Non-Safe State (NS), Very Non-Safe State (VNSS), Failure State (FS)*.



Gambar 3.11 Diagram Blok FQL untuk tujuan Obstacle Avoidance

Berikut pengelompokan state berdasarkan data sensor,

1. State = *Safe State* (SS) → semua data region sensor > 2 m.
2. State = *Non-Safe State* (NS) → salah satu data region sensor < 30 cm.
3. State = *Very Non-Safe State* (VNSS) → semua data region sensor < 30 cm.
4. State = *Failure State* (FS) → menabrak halangan.

Dengan menggunakan *Fuzzy Inference System* yang telah dimodifikasi pada subbab 2.2.5 dan dua masukan *fuzzy* dimana μ_1 adalah posisi halangan dan μ_2 adalah posisi taget, maka pada penelitian ini aturan yang digunakan adalah sebagai berikut,

Jika $\mu_1 = NS$ **dan** $\mu_2 = NB$ **maka** nilai $q = \{q_1, q_2, q_3, q_4, q_5\}$, yang bersesuaian dengan aksi {NB, NS, Z, PS, PB}.

Jika $\mu_1 = NS$ **dan** $\mu_2 = NS$ **maka** nilai $q = \{q_1, q_2, q_3, q_4, q_5\}$, yang bersesuaian dengan aksi {NB, NS, Z, PS, PB}.

Jika $\mu_1 = NS$ **dan** $\mu_2 = Z$ **maka** nilai $q = \{q_1, q_2, q_3, q_4, q_5\}$, yang bersesuaian dengan aksi {NB, NS, Z, PS, PB}.

Dan seterusnya.

Jumlah aturan yang digunakan berdasarkan pada jumlah region sensor yang digunakan yang ditunjukkan pada Tabel 3.1. Selanjutnya akan dilakukan proses learning pada masing-masing nilai q untuk setiap aturan yang ada. Nilai q merupakan bobot yang akan dilearning dengan cara yang di jelaskan pada subbab 2.2.5.

Tabel 3.1 Jumlah Aturan Setiap Perubahan Jumlah Region

Jumlah Region	Jumlah Aturan	Jumlah Q-Table
2	10	50
3	15	75
4	20	100
5	25	150

3.5 Hipotesa Penelitian

Berdasarkan kajian pustaka, beberapa hipotesa penelitian diantaranya,

1. Dengan memodifikasi fuzzyfikasi dengan penggabungan antara posisi halangan dan posisi robot, diharapkan robot bisa lebih cepat untuk menuju target dibandingkan dengan desain FQL pada paper (1).
2. Dengan modifikasi pengubahan penalaan *reinforcement* point dengan melihat dua state terakhir, diharapkan robot akan mendapatkan total reinforcement point yang lebih tinggi dibandingkan dengan desain FQL pada paper (1).
3. Dengan meperbanyak jumlah sudut region sensor dan mengubah pengolahan data sensor, diharapkan robot akan memiliki sudut hindar yang lebih minimum dibandingkan dengan desain FQL pada paper (1).

3.6 Kriteria Pengujian

Untuk melihat respon dari robot diukur berdasarkan dua fase, yaitu fase training dan fase testing. Kriteria yang akan dilihat adalah,

1. Pada fase training akan dilihat pada iterasi ke berapa robot mulai pandai dalam menentukan aksi ketika diketahui adanya *obstacle* dengan cara melihat reward poin yang didapatkan oleh robot.
2. Pada fase testing akan dilihat seberapa cepat robot akan mencapai target yang telah ditentukan dengan cara melihat banyak iterasi yang diperlukan untuk mencapai target.

Halaman ini sengaja dikosongkan

BAB 4
HASIL DAN PEMBAHASAN

Pada bab ini akan membahas perihal pengujian yang dilakukan. Terdapat tiga tahapan pengujian sistem diantaranya sebagai berikut,

1. Pengujian desain proses fuzzifikasi
2. Pengujian penalaan *reinforcement* point.
3. Pengujian jumlah sudut region sensor.
4. Pengujian pengolahan data region sensor.

Keempat pengujian tersebut dilakukan dengan spesifikasi robot SCITOS G5 yangditunjukkan pada tabel 4.1. Untuk menggunakan FQL diperlukan tuning beberapa konstanta yang telah dilakukan pada paper (1). Dengan rumus (2.12) menggunakan nilai $\alpha = 0.0001,\ \varepsilon = 0.5,\ \gamma = 0.9, \lambda = 0.3$[1].

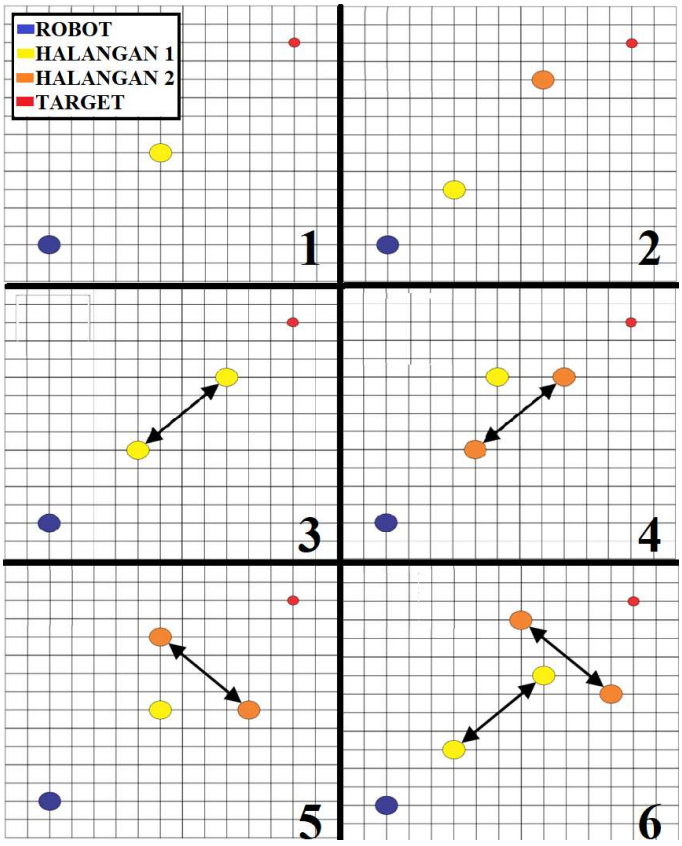
Tabel 4.1 Spesifikasi Robot SCITOS-G5

Spesifikasi	Keterangan
Panjang	737 mm
Lebar	617 mm
Berat	60 kg
Tipe Mobilitas	*Differential Drive Robot*
Roda Penggerak	2 roda, diameter 200mm, lebar 40mm
Rasio Gear	i = 39
Kecepatan Maksimal	1.1 m/s
Bumper	Mengelilingi Robot
Sonar	24 Ultrasonic Range Finder
	Beam angle: 360°
	Beam angle satu sensor: 15°
	Jarak Deteksi: 200mm - 3000mm
Encoder	Tipe: incremental encoder
	Resolusi: 460 ticks/rad

Tabel 4.2 Jumlah Halangan Pada Skenario Pengujian

Skenario	Jumlah Halangan	
	Statis	Dinamis
1	1	0
2	2	0
3	0	1
4	1	1
5	1	1
6	0	2

Kombinasi halangan dari skenario lingkungan yang dibuat ditunjukkan pada Tabel 4.2. Pengujian akan dilakukan sebanyak 10 kali untuk masing-masing skenario lingkungan yang ditunjujjan pada Gambar 4.1.



Gambar 4.1 Skenario Lingkungan Pengujian

Tabel 4.3 Desain Pengujian Pertama

Model	Desain	
	1	2
Fuzzifikasi	Switch	Tanpa Switch
Reinforcement Point	S_t	S_t
Sudut Region	2	2
Pengolahan Data	Rata-Rata	Rata-Rata

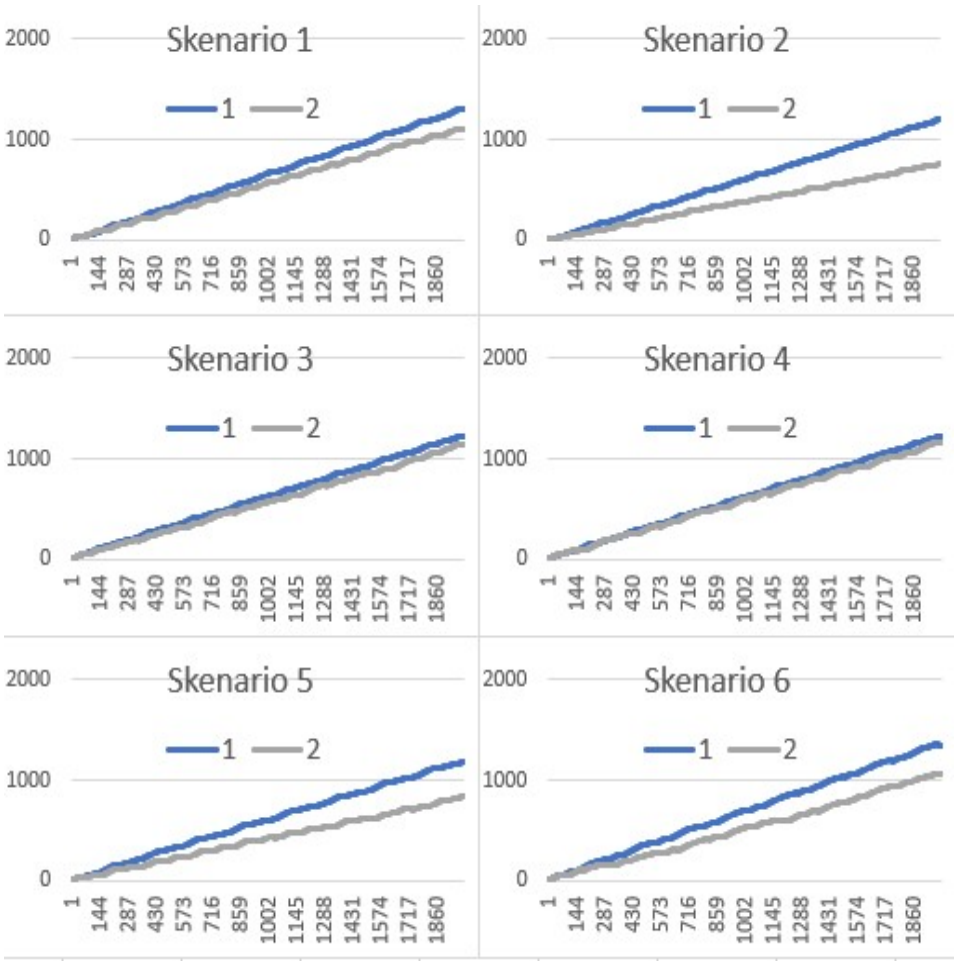
4.1 Pengujian Desain Proses Fuzzifikasi

Pengujian ini dilakukan dengan dua desain proses data premis fuzzy. Desain pertama dilakukan sesuai dengan paper (1), dengan memisahkan algoritma antara menghindari halangan dengan menuju target sehingga membutuhkan proses *switch* algoritma. Desain kedua adalah dengan menggabungkan algoritma menghindari halangan dan menuju ke target menjadi dengan menghilangkan proses *switch*. Untuk menggunakan desain kedua diperlukan penyesuaian data premis fuzzy untuk mengurangi rule pada fuzzy, yang telah dijelaskan pada subbab 3.2.1.2.

Tujuan dari pengujian pertama ini dilakukan untuk membandingkan performa desain FQL pada paper (1) dengan desain FQL pada Thesis yang telah dibuat. Desain yang digunakan untuk pengujian pertama ditunjukkan pada Tabel 4.3.

4.1.1 Pengujian Pada Fase Training

Pada fase training akan dilihat berapa total reinforcement point yang didapatkan, semakin besar total nilai *reinforcement* poin maka semakin baik dalam proses belajarnya karena poin bernilai positif jika robot melakukan aksi yang benar berdasarkan umpan balik dari data sensor sonar. Grafik akumulasi *reinforcement* poin ditunjukkan pada Gambar 4.2. Sumbu y menunjukkan total dari nilai *reinforcement* poin yang didapatkan dan sumbu x menunjukkan waktu iterasi. Warna biru adalah grafik untuk desain pertama sedangkan abu-abu adalah desain kedua.



Gambar 4.2 Hasil Pengujian Total Reinforcement Point Pada Pengujian Desain Proses Fuzzifikasi

Dengan melihat hasil percobaan menggunakan desain pertama mendapatkan nilai total *reinforcement* poin lebih tinggi dari pada menggunakan desain kedua untuk semua skenario, hal ini membuktikan pada fase training menggunakan desain pertama lebih cepat dalam menentukan aksi pada fase training, karena pada desain pertama ketika robot menghindari halangan menghiraukan targetnya, sehingga robot bisa bergerak bebas untuk menghindari halangan yang terdeteksi oleh sensor sonar. Total akumulasi reinforcement poin ditunjukkan pada Tabel 4.4.

Tabel 4.4 Jumlah Nilai Reinforcement Dari 2000 Iterasi Pada Pengujian Proses Fuzzifikasi

Skenario	Total Nilai Reinforcement	
	Desain 1	**Desain 2**
1	1590	1108
2	1631	753
3	1742	1148
4	1675	1163
5	1304	883
6	1166	1062

4.1.2 Pengujian Pada Fase Testing

Dari hasil yang didapatkan pada Tabel 4.5, menunjukkan semakin kecilnya nilai angka iterasi mengartikan semakin cepat robot bergerak menuju target. Untuk semua skenario lebih cepat menggunakan desain kedua untuk mencapai target, karena untuk desain kedua ketika robot bergerak menghindari halangan melakukan sudut hindar berdasarkan arah hadap robot ke target. Hal ini membuktikan dengan menghilangkan proses *switch* algoritma dapat meningkatkan performa dari penggunaan FQL.

Tabel 4.5 Jumlah Iterasi Yang Dibutuhkan Untuk Mencapai Target Pada Pengujian Proses Fuzzifikasi

Skenario	Jumlah Iterasi Untuk Mencapai Target	
	Desain 1	**Desain 2**
1	240	197
2	310	263
3	275	221
4	220	173
5	263	219

6	297	240

4.2 Pengujian Penalaan Reinforcement Point

Pengujian ini dilakukan untuk melihat respon robot dari penggunaan *reinforcement* poin yang diubah dari paper pertama yang menggunakan *state* terakhir untuk dibandingkan dengan penalaan *reinforcement* poin perubahan *state*. Berdasarkan data pengujian dari pengujian pertama dengan menghilangkan proses *switch* algoritma lebih cepat untuk mencapai target maka, pengujian dilakukan dengan proses fuzzifikasi tanpa *switch*, 3 sudut region sensor dan pengolahan data menggunakan rata-rata.

Hasil yang diperoleh adalah penalaan *reinforcement* poin yang terbaik menurut kriteria pengujian. Untuk melihat respon pada fase training dilakukan dengan melihat total akumulasi nilai reinforcement yang didapatkan. Semakin besar nilai reinforcement maka dapat dikatakan robot semakin baik dalam proses belajarnya. Desain yang akan dibandingkan ditunjukkan pada Tabel 4.6.

Tabel 4.6 Desain Pengujian Kedua

Model	**Desain**	
	1	**2**
Fuzzifikasi	Tanpa Switch	Tanpa Switch
Reinforcement Point	S_t	$S_{t-1} \rightarrow S_t$
Sudut Region	3	3
Pengolahan Data	Rata-Rata	Rata-Rata

Penalaan *reinforcement* poin menggunakan state terakhir (S_t) didefinisikan sebagai berikut,

- Bertabrakan dengan *obstacle FS*, $r = -2$.
- State V*NSS*, $r = -1$.
- State NS, $r = \dot{0}$.
- State SS, $r = 1$.

Penalaan *reinforcement* poin menggunakan perubahan state $(S_{t-1} \rightarrow S_t)$ didefinisikan sebagai berikut,

- Bertabrakan dengan *obstacle FS*, $r = -2$.
- Bergerak dari *NS* ke V*NSS*, $r = -1$.
- Bergerak tetapi tidak ada perubahan state, $r = \dot{0}$.
- Bergerak dari V*NSS* ke *NS*, $r = 1$.
- Bergerak dari *NS* ke *SS*, $r = 2$

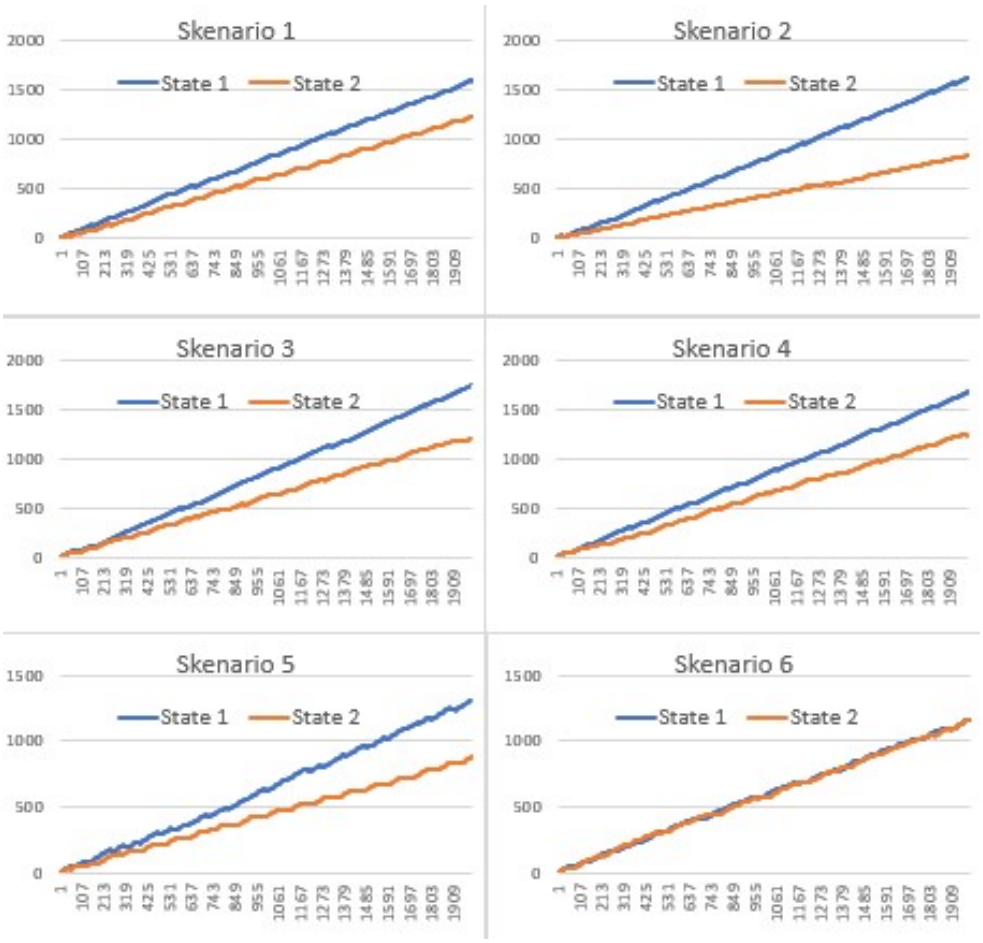
4.2.1 Pengujian Pada Fase Training

Dengan melihat hasil percobaan menggunakan state terakhir (S_t) sebagai acuan menentukan nilai *reinforcement* mendapatkan nilai total lebih tinggi dari pada menggunakan perubahan state $(S_{t-1} \rightarrow S_t)$, hal ini membuktikan pada fase training menggunakan state terakhir (S_t) lebih cepat dalam menentukan aksi.

Kelemahan menggunakan kondisi state terakhir (S_t) ketika ada halangan yang bergerak akan terjadi kesalahan pada saat menentukan nilai poin *reinforcement* yaitu ketika halangan bergerak menjauh dari robot, poin *reinforcement* memberikan *reward* atas keputusan yang diambil meskipun keputusan tersebut belum pasti kebenarannya. Total akumulasi reinforcement poin ditunjukkan pada Tabel 4.7.

Tabel 4.7 Jumlah Nilai Reinforcement Dari 2000 Iterasi Pada Pengujian State

Skenario	**Total Nilai Reinforcement**	
	(S_t)	$(S_{t-1} \rightarrow S_t)$
1	1590	1239
2	1631	846
3	1742	1201
4	1675	1243
5	1304	881
6	1166	1165



Gambar 4.3 Total *Reinforcement* Poin Pengujian *Reinforcement* Poin

Pada Gambar 4.3, grafik dengan warna biru adalah total *reinforcement* poin menggunakan state terakhir (S_t), sedangkan warna oranye adalah total *reinforcement* poin menggunakan perubahan state ($S_{t-1} \rightarrow S_t$). Sumbu y merupakan nilai total *reinforcement* poin dan sumbu x merupakan waktu iterasi.

4.2.2 Pengujian Pada Fase Testing

Pengujian pada fase testing dilakukan berdasarkan data bobot yang didapatkan pada fase *training*. Pada pengujian 5 dan 6 menggunakan penalaan reinforcement poin state terakhir (S_t) masih terjadi tabrakan 2 kali dari 10 kali pengujian, hal ini tidak memenuhi syarat untuk digunakan sebagai acuan pengujian yang lain maka, dari pengujian pertama ini menggunakan perubahan state ($S_{t-1} \rightarrow S_t$) lebih baik digunakan untuk pengujian sudut region.

Tabel 4.8 Jumlah Iterasi Yang Dibutuhkan Untuk Mencapai Target Pada Pengujian State

Skenario	Jumlah Iterasi Untuk Mencapai Target	
	(S_t)	$(S_{t-1} \rightarrow S_t)$
1	94	109
2	101	118
3	114.6	121
4	109.8	118.2
5	150	128
6	171.7	145

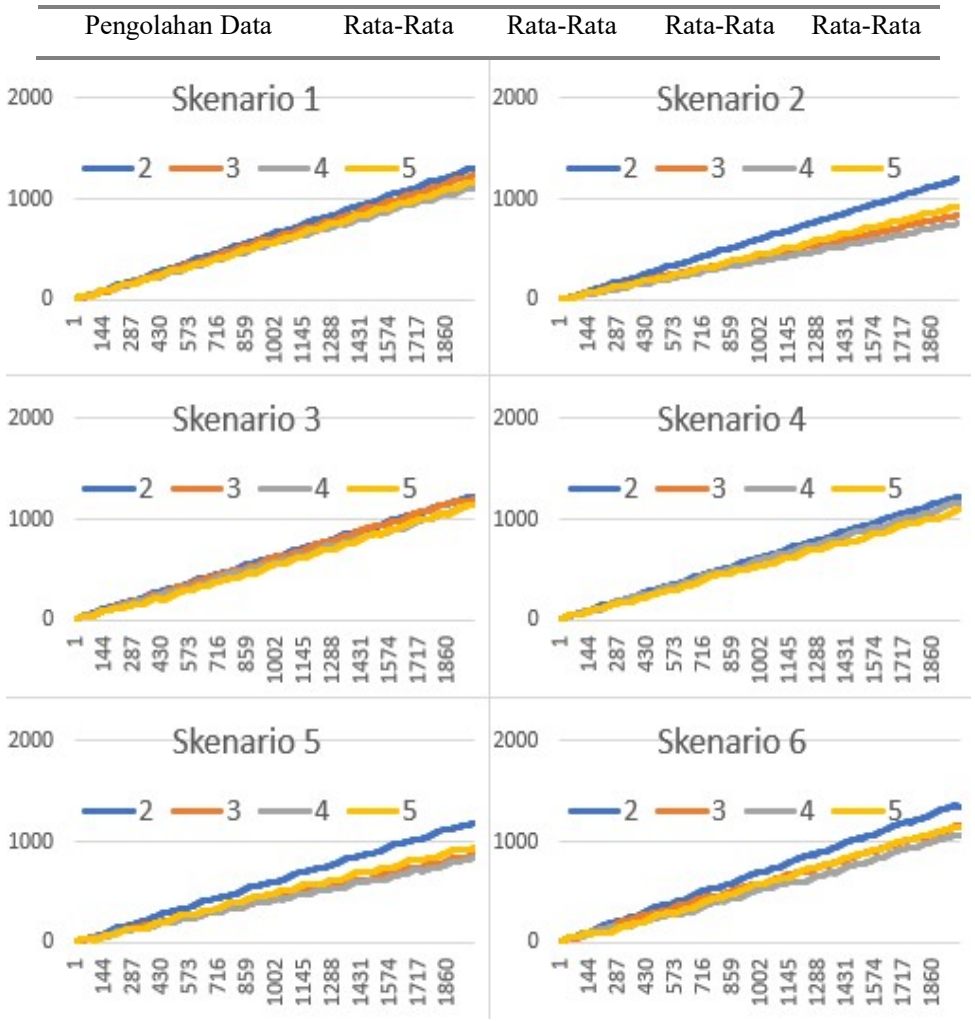
Dari hasil yang ditunjukkan pada Tabel 4.8, untuk skenario 1-4 lebih cepat menggunakan state terakhir (S_t) untuk menentukan nilai reinforcement. Sedangkan untuk skenario 5 dan 6 lebih cepat menggunakan $(S_{t-1} \rightarrow S_t)$. Hal ini dikarenakan pada skenario 5 dan 6 terdapat halangan yang bergerak menyilang dari arah robot menuju ke target.

4.3 Pengujian Jumlah Sudut Region Sensor

Setelah didapatkan penalaan reward poin yang terbaik dari pengujian kedua, yaitu dengan menggunakan perubahan state, penalaan reward poin tersebut digunakan untuk pengujian yang ketiga.

Tabel 4.9 Desain Pengujian Ketiga

Model	Desain			
	1	**2**	**3**	**4**
Fuzzifikasi	Tanpa Switch	Tanpa Switch	Tanpa Switch	Tanpa Switch
Reinforcement Point	$S_{t-1} \rightarrow S_t$	$S_{t-1} \rightarrow S_t$	$S_{t-1} \rightarrow S_t$	$S_{t-1} \rightarrow S_t$
Sudut Region	2	3	4	5



Gambar 4.4 Grafik Total Reinforcement Poin Pengujian Ketiga

Pengujian dilakukan menggunakan pengolahan data nilai rata-rata sudut region. Pengujian ini dilakukan untuk melihat respon robot menggunakan 2,3,4 dan 5 sudut region sensor dengan menggunakan kinematika dari robot untuk melihat hasil mana yang terbaik berdasarkan kriteria pengujian. Karena dengan menggunakan sudut region yang semakin banyak maka semakin banyak pula aturan yang harus dilatih pada robot, maka ada kemungkinan pengujian menggunakan 5 sensor belum tentu memiliki hasil yang terbaik dari yang lain. Hasil yang diperoleh adalah banyak sudut region sensor yang terbaik menurut kriteria pengujian. Penjelasan tentang pembagian sudut region sensor pada subbab 3.2.1.2.

Tabel 4.10 Jumlah Nilai Reinforcement Dari 2000 Iterasi Pada Pengujian Sudut Region Sensor

Skenario	Total Nilai Reinforcement			
	2 Region	3Region	4 Region	5 Region
1	1310	1239	1208	1154
2	1208	846	753	619
3	1228	1201	1148	1142
4	1224	1133	1123	1091
5	1180	881	873	841
6	1352	1165	1072	1035

4.3.1 Pengujian Pada Fase Training

Pengujian dilakukan sebanyak 2000 iterasi untuk mengetahui performa yang didapatkan pada fase training. Dari hasil pengujian yang dilakukan dengan menggunakan 2 region sensor memiliki akumulasi nilai rata-rata dari total reinforcement poin yang paling tinggi untuk semua skenario lingkungan. Hal ini dikarenakan rule yang digunakan lebih sedikit, sehingga proses *learning* lebih cepat. Semakin banyak jumlah sudut region yang digunakan akan memperlambat proses *learning* pada algoritma FQL. Hasil dari pengujian ditunjukkan pada Tabel 4.10.

4.3.2 Pengujian Pada Fase Testing

Dengan menggunakan data bobot pada fase training dilakukan pengujian fase testing. Pada pengujian fase testing dengan menggunakan 2 sudut region sensor terlalu lama untuk mencapai target dibandingkan dengan region lainnya. Dari Tabel 4.11 ditunjukkan dengan menggunakan 5 sudut region sensor robot bisa mencapai target lebih cepat untuk skenario lima dan enam yang terdapat halangan dinamis, sedangkan dengan menggunakan 3 sudut region sensor robot bisa mencapai target lebih cepat untuk halangan statis. Penggunaan 5 sudut region sensor direkomendasikan untuk halangan dinamis sedangkan 3 sudut region sensor untuk halangan yang statis.

Tabel 4.11 Jumlah Iterasi yang Dibutuhkan Untuk Mencapai Target Pada Pengujian Sudut Region

Skenario	Jumlah Iterasi Untuk Mencapai Target			
	2 Region	3Region	4 Region	5 Region
1	197	100	103	115
2	263	103	117	125
3	221	121	136	131
4	173	131	133	141
5	219	153	148	128
6	240	235	236	202

4.4 Pengujian Pengolahan Data Sensor

Setelah didapatkan desain yang terbaik dari pengujian pertama dan kedua, yaitu dengan menggunakan perubahan state dan 5 sudut region sensor, penalaan reward poin dan jumlah sudut region tersebut digunakan untuk pengujian yang keempat. Pengujian ini dilakukan untuk melihat respon robot menggunakan pengolahan data dengan cara menghitung rata-rata dari sensor didalam satu region dan dengan cara menghitung jarak terdekat dari salah satu sensor didalam satu region. Hasil pengujian ditunjukkan pada Tabel 4.12.

Tabel 4.12 Jumlah Iterasi yang Dibutuhkan Untuk Mencapai Target Pada Pengujian Pengolahan Data Sensor

Skenario	Jumlah Iterasi Untuk Mencapai Target	
	Jarak Terdekat	Rata - Rata
1	123	110
2	160	135
3	167	158
4	180	175
5	160	143
6	212	202

Pada pengujian keempat menggunakan nilai rata-rata memiliki waktu lebih cepat untuk mencapai target. Hal ini karena, ketika menggunakan jarak terdekat bobot akan sama untuk dua region yang berbeda ketika halangan berada diantara perpotongan 2 region.

4.5 Kesimpulan Pengujian

Setelah dilakukan pengujian didapatkan desain untuk penggunaan metode FQL yang ditunjukkan pada Tabel 4.13. Untuk halangan statis menggunakan 3 sudut region sensor dan kondisi *state* terakhir sedangkan untuk halangan dinamis menggunakan 5 sudut region sensor dan kondisi perubahan *state.*

Tabel 4.13 Desain yang Didapatkan Dari Pengujian

Model	**Halangan**	
	Statis	**Dinamis**
Fuzzifikasi	Tanpa Switch	Tanpa Switch
Reinforcement Point	S_t	$S_{t-1} \rightarrow S_t$
Sudut Region	3	5
Pengolahan Data	Rata-Rata	Rata-Rata

Halaman ini sengaja dikosongkan

BAB 5
KESIMPULAN

Pada Tesis ini telah dideskripsikan bagaimana mendesain suatu algoritma FQL untuk tujuan menghindari halangan dan menuju ke target. Pada pengujian desain proses fuzzifikasi dengan menghilangkan proses switch membuat fase training lebih lama tetapi waktu tempuh menuju target lebih cepat. Pada pengujian reinforcement poin dengan referensi state sekarang (S_t) memang menghasilkan sudut hindar yang minimum daripada menggunakan perubahan state ($S_{t-1} \rightarrow S_t$) akan tetapi masih ada kemungkinan terjadi tabrakan ketika diuji dengan halangan yang bergerak.

Pada pengujian jumlah sudut region menggunakan 5 sudut region menghasilkan waktu tercepat untuk mencapai target untuk halangan yang dinamis sedangkan untuk halangan yang statis menggunakan 3 sudut region. Untuk semua skenario menggunakan rata-rata pada pengolahan data menghasilkan waktu tempuh tercepat. Dari hasil yang telah didapatkan, diketahui bahwa FQL yang telah didesain mampu meningkatakan performa algoritma terhadap kondisi lingkungan, sehingga robot bisa menghindari halangan dan mencapai target.

Halaman ini sengaja dikosongkan

DAFTAR PUSTAKA

[1] Wicaksono H, Anam K, Prihastono, Sulistijono IA, Kuswadi S*, "Compact Fuzzy Q-Learning For Autonomous Mobile Robot Navigation", Project for Reasearch and Education Development on Information and Communication Technology in Sepuluh Nopember Institute of Technology(PREDICT – ITS)*, 2014.

[2] Jaradat MAK, Al-Rousan M, Quadan L, *"Reinforcement Based Mobile Robot Navigation in Dynamic Environtment"*, *Robotics and Computer-Integrated Manufacturing,* vol. 27, pp. 135-149, 2010.

[3] Zaki AM, Arafa O, Amer SI, *"Microcontroller-based mobile robot positioning and obstacle avoidance"*, *Power Electronic and Energy Conversition Department, Electronics Research Institute, El-Tahrir Street, Dokki, Cairo, Egypt*, 2014.

[4] Dhaouadi R, Hatab AA *, "Dynamic Modelling of Differential-Drive Mobile Robots Using Lgrange And Newton-Euler Methodologies: A Unified Framework", Advance in Robotics & Automation*, 2013.

[5] Jaradat MAK, Al-Rousan M, Quadan L, *"Reinforcement Based Mobile Robot Navigation in Dynamic Environtment"*, *Robotics and Computer-Integrated Manufacturing,* vol. 27, pp. 135-149, 2010.

[6] Anonymous ,*"User Manual Scitos G5 Version 1.2"*.

[7] Deng C, Er MJ, Xu J, *"Dynamic Fuzzy Q-Learning and Control of Mobile Robots"*, *International Conference on Control, Automation, Robotics and Vision,* vol. 8, , 2004.

[8] Choi YC, Son J, Ahn H, *"Fault Detection and Isolation for Small CMG-based Satellite: AFuzzy Q-Learning Approach"*, *Aerospace Science and Technology,* vol. 47, pp. 340-355, 2015.

[9] Passino KM, Yurkovich S, *"Fuzzy Control"*, Addison-Wesley, 1998.

Halaman ini sengaja dikosongkan

LAMPIRAN

Rule Fuzzy

2 Region Sensor:

1. **Jika** $\mu_1 = NS$ **dan** $\mu_2 = NB$
2. **Jika** $\mu_1 = NS$ **dan** $\mu_2 = NS$
3. **Jika** $\mu_1 = NS$ **dan** $\mu_2 = Z$
4. **Jika** $\mu_1 = NS$ **dan** $\mu_2 = PS$
5. **Jika** $\mu_1 = NS$ **dan** $\mu_2 = PB$
6. **Jika** $\mu_1 = PS$ **dan** $\mu_2 = NB$
7. **Jika** $\mu_1 = PS$ **dan** $\mu_2 = NS$
8. **Jika** $\mu_1 = PS$ **dan** $\mu_2 = Z$
9. **Jika** $\mu_1 = PS$ **dan** $\mu_2 = PS$
10. **Jika** $\mu_1 = PS$ **dan** $\mu_2 = PB$

3 Region Sensor:

1. **Jika** $\mu_1 = NS$ **dan** $\mu_2 = NB$
2. **Jika** $\mu_1 = NS$ **dan** $\mu_2 = NS$
3. **Jika** $\mu_1 = NS$ **dan** $\mu_2 = Z$
4. **Jika** $\mu_1 = NS$ **dan** $\mu_2 = PS$
5. **Jika** $\mu_1 = NS$ **dan** $\mu_2 = PB$
6. **Jika** $\mu_1 = Z$ **dan** $\mu_2 = NB$
7. **Jika** $\mu_1 = Z$ **dan** $\mu_2 = NS$
8. **Jika** $\mu_1 = Z$ **dan** $\mu_2 = Z$
9. **Jika** $\mu_1 = Z$ **dan** $\mu_2 = PS$
10. **Jika** $\mu_1 = Z$ **dan** $\mu_2 = PB$
11. **Jika** $\mu_1 = PS$ **dan** $\mu_2 = NB$
12. **Jika** $\mu_1 = PS$ **dan** $\mu_2 = NS$
13. **Jika** $\mu_1 = PS$ **dan** $\mu_2 = Z$
14. **Jika** $\mu_1 = PS$ **dan** $\mu_2 = PS$
15. **Jika** $\mu_1 = PS$ **dan** $\mu_2 = PB$

4 Region Sensor:

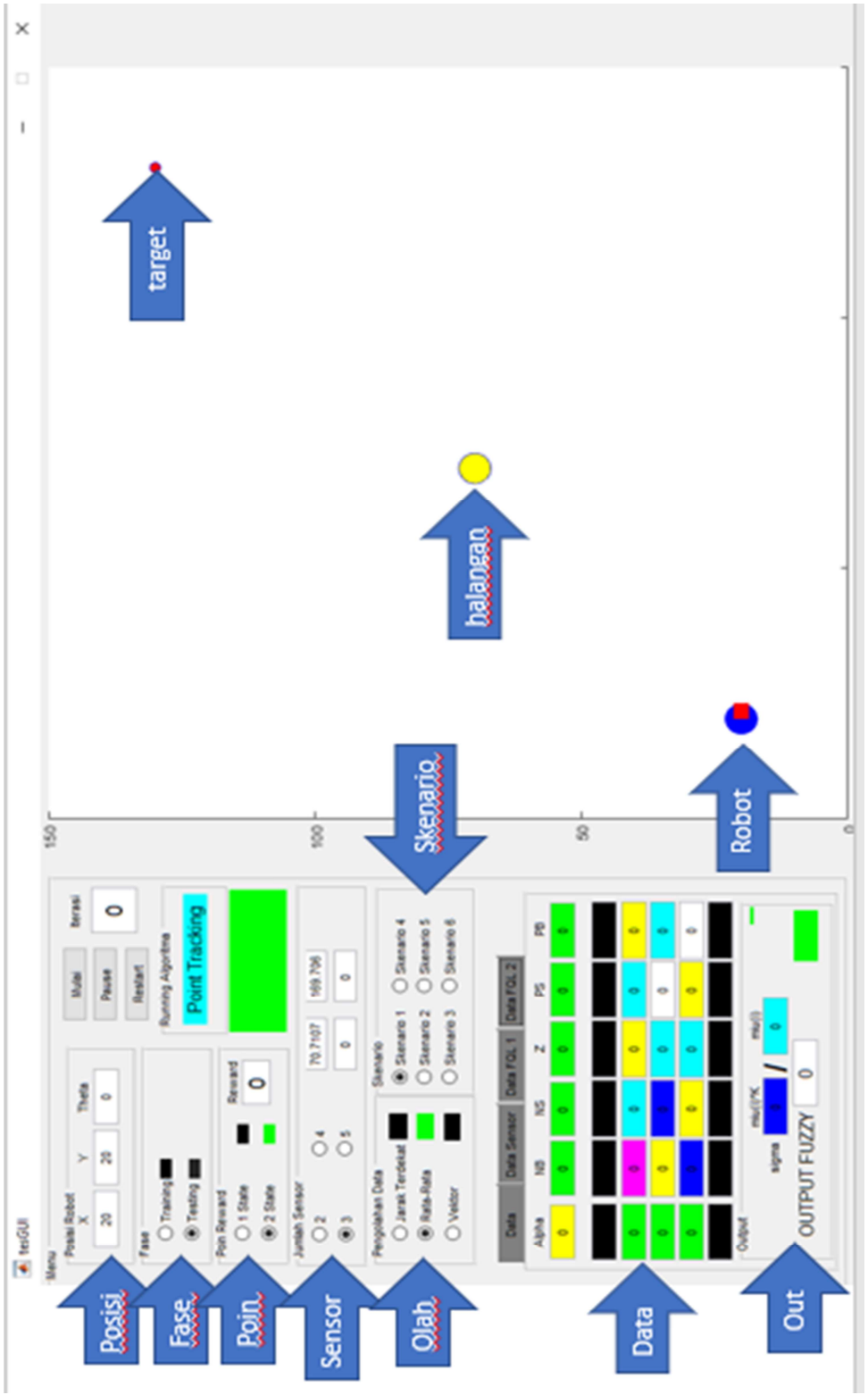
1. **Jika** $\mu_1 = NB$ **dan** $\mu_2 = NB$
2. **Jika** $\mu_1 = NB$ **dan** $\mu_2 = NS$
3. **Jika** $\mu_1 = NB$ **dan** $\mu_2 = Z$
4. **Jika** $\mu_1 = NB$ **dan** $\mu_2 = PS$
5. **Jika** $\mu_1 = NB$ **dan** $\mu_2 = PB$
6. **Jika** $\mu_1 = NS$ **dan** $\mu_2 = NB$
7. **Jika** $\mu_1 = NS$ **dan** $\mu_2 = NS$
8. **Jika** $\mu_1 = NS$ **dan** $\mu_2 = Z$
9. **Jika** $\mu_1 = NS$ **dan** $\mu_2 = PS$
10. **Jika** $\mu_1 = NS$ **dan** $\mu_2 = PB$
11. **Jika** $\mu_1 = PS$ **dan** $\mu_2 = NB$
12. **Jika** $\mu_1 = PS$ **dan** $\mu_2 = NS$
13. **Jika** $\mu_1 = PS$ **dan** $\mu_2 = Z$
14. **Jika** $\mu_1 = PS$ **dan** $\mu_2 = PS$
15. **Jika** $\mu_1 = PS$ **dan** $\mu_2 = PB$
16. **Jika** $\mu_1 = PB$ **dan** $\mu_2 = NB$
17. **Jika** $\mu_1 = PB$ **dan** $\mu_2 = NS$
18. **Jika** $\mu_1 = PB$ **dan** $\mu_2 = Z$
19. **Jika** $\mu_1 = PB$ **dan** $\mu_2 = PS$
20. **Jika** $\mu_1 = PB$ **dan** $\mu_2 = PB$

5 Region Sensor:

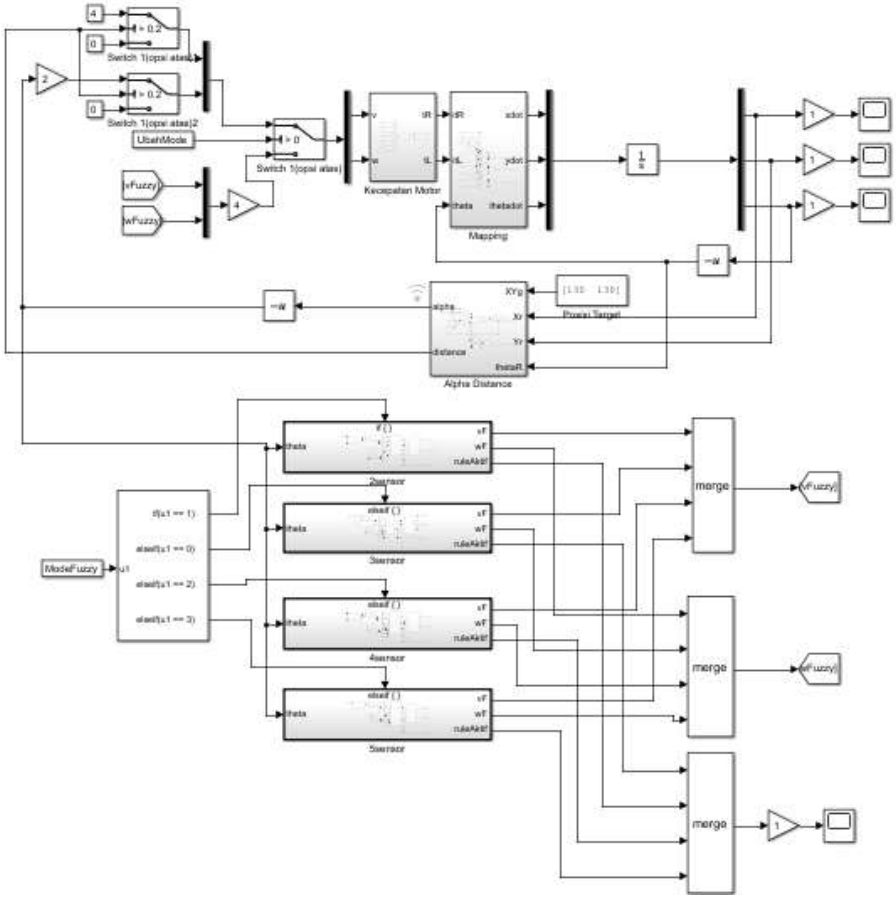
1. **Jika** $\mu_1 = NB$ **dan** $\mu_2 = NB$
2. **Jika** $\mu_1 = NB$ **dan** $\mu_2 = NS$
3. **Jika** $\mu_1 = NB$ **dan** $\mu_2 = Z$
4. **Jika** $\mu_1 = NB$ **dan** $\mu_2 = PS$
5. **Jika** $\mu_1 = NB$ **dan** $\mu_2 = PB$
6. **Jika** $\mu_1 = NS$ **dan** $\mu_2 = NB$
7. **Jika** $\mu_1 = NS$ **dan** $\mu_2 = NS$
8. **Jika** $\mu_1 = NS$ **dan** $\mu_2 = Z$

9. **Jika** $\mu_1 = NS$ **dan** $\mu_2 = PS$
10. **Jika** $\mu_1 = NS$ **dan** $\mu_2 = PB$
11. **Jika** $\mu_1 = Z$ **dan** $\mu_2 = NB$
12. **Jika** $\mu_1 = Z$ **dan** $\mu_2 = NS$
13. **Jika** $\mu_1 = Z$ **dan** $\mu_2 = Z$
14. **Jika** $\mu_1 = Z$ **dan** $\mu_2 = PS$
15. **Jika** $\mu_1 = Z$ **dan** $\mu_2 = PB$
16. **Jika** $\mu_1 = PS$ **dan** $\mu_2 = NB$
17. **Jika** $\mu_1 = PS$ **dan** $\mu_2 = NS$
18. **Jika** $\mu_1 = PS$ **dan** $\mu_2 = Z$
19. **Jika** $\mu_1 = PS$ **dan** $\mu_2 = PS$
20. **Jika** $\mu_1 = PS$ **dan** $\mu_2 = PB$
21. **Jika** $\mu_1 = PB$ **dan** $\mu_2 = NB$
22. **Jika** $\mu_1 = PB$ **dan** $\mu_2 = NS$
23. **Jika** $\mu_1 = PB$ **dan** $\mu_2 = Z$
24. **Jika** $\mu_1 = PB$ **dan** $\mu_2 = PS$
25. **Jika** $\mu_1 = PB$ **dan** $\mu_2 = PB$

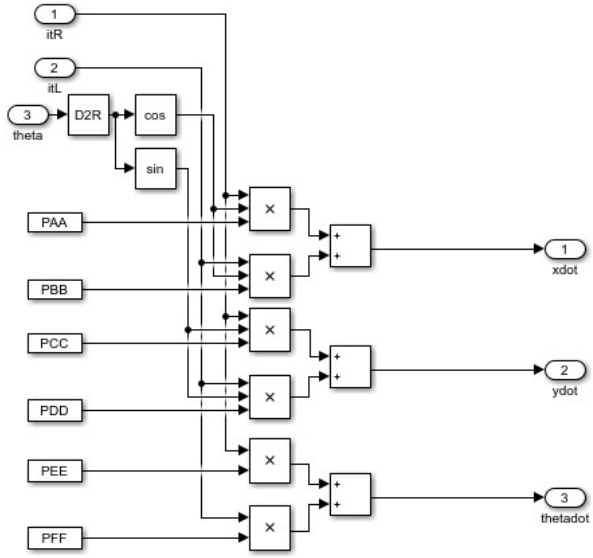
GUI PROGRAM


target
halangan
Skenario
Robot
Posisi
Fase
Poin
Sensor
Olah
Data
Out
OUTPUT FUZZY
Point Tracking
Mulai
Pause
Restart
Data
Data Sensor
Data FGL 1
Data FGL 2

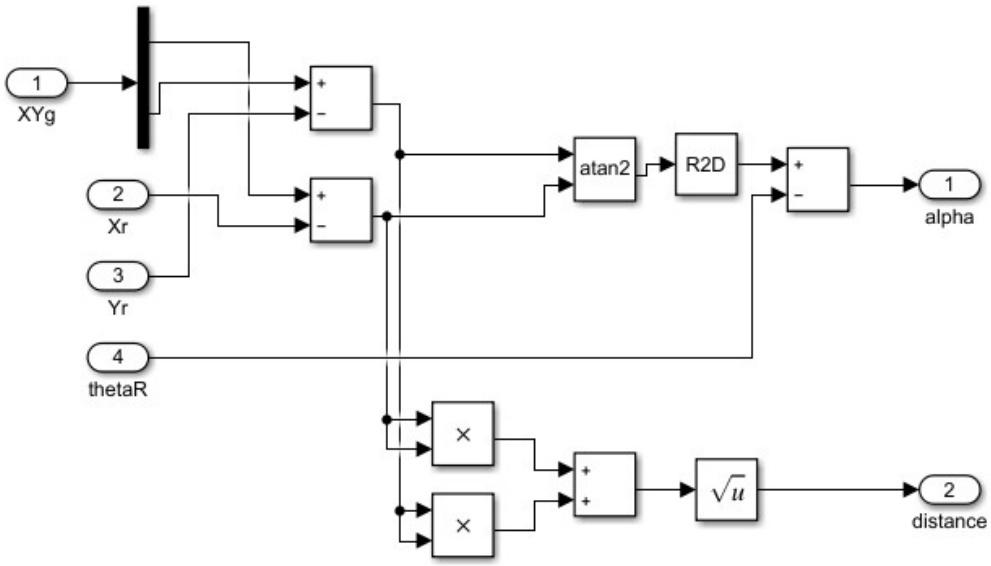
SIMULINK MAIN



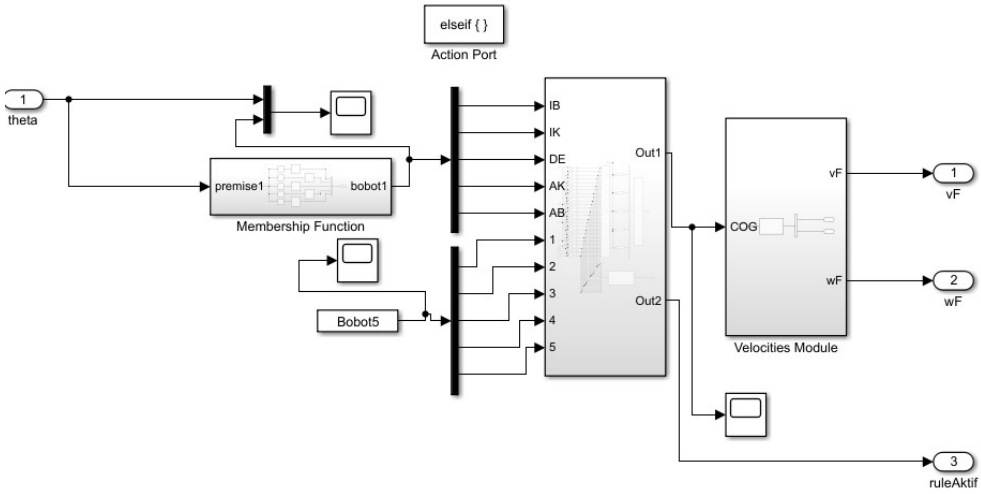
SIMULINIK MAPPING



SIMULINK ALPHA DISTANCE



SIMULINK FUZZY



INISIALISASI PROGRAM

```
JRoda=2; %2dm=200mm
hRoda=4;
%data matrix kecepatan motor
AA=0.5; BB=1;
CC=0.5; DD=-1;
%data kecepatan maksimum
VRobot=1.1;
%data matrix posisi
PAA=JRoda/2;      PBB=JRoda/2;
PCC=JRoda/2;      PDD=JRoda/2;
```

```
PEE=JRoda/hRoda; PFF=-JRoda/hRoda;

OlahData=[];
UbahMode=1;
ModeFuzzy=3;
val=0;
Bobot5=[0 0 0 0 0];
Bobot4=[0 0 0 0];
Bobot3=[0 0 0];
Bobot2=[0 0];
Metode=1;
GainFuzzy=6;
SpeedObs=3; %semakin besar semakin pelan
eps=0.5;
gamma=0.9;
lamda=0.3;
alfa=0.1;
lamaSampling=10;
%Qtable
QTablePaper=[0.007283, 0.005914, 0.309990, 0.003963, 0.004107;...1
             0.000653, 5.006245, 0.000706, 0.001446, 0.001406;...2
             4.997247,-0.000052, 0.000000, 0.000000, 0.000006;...3
             0.000091, 0.000193, 0.000119, 0.008982, 0.000035;...4
             0.000183, 4.997324, 0.000068,-0.000014, 0.000112;...5
             4.997324, 0.000000, 0.000005,-0.000027,-0.000006;...6
            -0.000016,-0.000019,-0.000012,-0.000013, 0.000000;...7
            -0.000047,-0.000037,-0.000037,-0.000038,-0.000033;...8
             0.007283, 0.000000, 0.000000,-0.000010, 0.000068;...9
            0.000513, 0.000087, 0.000349, 0.000709, 0.019596;...10
            0.000014, 0.000065, 0.000045, 0.001756, 0.000042;...11
           -0.000035,-0.000044,-0.000035,-0.000026,-0.000045;...12
           -0.000020, 0.000117, 0.000001, 0.000094, 0.003945;...13
           -0.000107,-0.000094,-0.000125,-0.000094,-0.000091;...14
           -0.000264,-0.000221,-0.000223,-0.000254,-0.000230;...15
           -0.000019,-0.000126,-0.000025,-0.000008,-0.000137;...16
           -0.000162,-0.000162,-0.000176,-0.000166,-0.000162;...17
           -0.000846,-0.000820,-0.000819,-0.000817,-0.000900;...18
            0.000000, 0.000001, 0.000119, 0.000007, 4.999763;...19
           -0.000181,-0.000139,-0.000032,-0.000022,-0.000123;...20
           -0.000043,-0.000001, 0.000000,-0.000021, 0.000000;...21
            0.000000,-0.000002, 0.000001, 0.000000, 4.999727;...22
           -0.000008,-0.000010, 0.000011,-0.000019,-0.000013;...23
           -0.000089,-0.000074,-0.000077,-0.000204,-0.000088;...24
            0.000000, 0.000000, 0.000000, 0.000000, 4.999187;...25
           -0.000230,-0.000227,-0.000243,-0.000232,-0.000236;...26
            -0.004685,-0.004662,-0.004707,-0.004710,-0.004672];
```

PROGRAM REINFORCEMENT POINT

```
function CekHadiah(hObject)
handles=guidata(hObject);
  if get(handles.radiobutton2,'Value')==0 &&
strcmp(get(handles.TrackingText,'String'),'FQL')
      DataObs1=str2double(get(handles.DH1,'String'));
      DataObs2=str2double(get(handles.DH2,'String'));
      DataObs1Aksen=str2double(get(handles.DA1,'String'));
      DataObs2Aksen=str2double(get(handles.DA2,'String'));
      %current state
```

```
        if get(handles.RadState1,'Value')==1
            if DataObs1 >= 14 && DataObs2 >= 14
                handles.re=1;
            elseif DataObs1 <= 8 || DataObs2 <= 8
                handles.re=-2;
            elseif DataObs1 <= 12 || DataObs2 <= 12
                handles.re=-1;
            else
                handles.re=0;
            end
            set(handles.HadiahBox,'String',handles.re);
        %transition state
        else
            StateCur=0;
            StateB4=0;
            if DataObs1Aksen >= 13 && DataObs2Aksen >= 13
                StateCur=1; %SAFE STATE
            elseif DataObs1Aksen <= 8 || DataObs2Aksen <= 8
                StateCur=4; %FAILSTATE
            elseif DataObs1 <= 11 || DataObs2 <= 11
                StateCur=3; %VERY NON-SAFE STATE
            else
                StateCur=2; %NON-SAFE STATE
            end
            if DataObs1 >= 13 && DataObs2 >= 13
                StateB4=1; %SAFE STATE
            elseif DataObs1 <= 8 || DataObs2 <= 8
                StateB4=4; %FAILSTATE
            elseif DataObs1 <= 11 || DataObs2 <= 11
                StateB4=3; %VERY NON-SAFE STATE
            else
                StateB4=2; %NON-SAFE STATE
            end

            if StateCur == 4
               handles.re=-2;
            elseif StateCur == StateB4
               handles.re=0;
            elseif StateCur > StateB4
               handles.re=1;
            else
               handles.re=-1;
            end
            set(handles.HadiahBox,'String',handles.re);
        end
    else
        set(handles.HadiahBox,'String','X');
    end

guidata(hObject, handles);
```

PROGRAM UTAMA

```
function Play(XH1,YH1,XH2,YH2,XT,YT,hObject)
handles=guidata(hObject);
```

```
      Xr=str2double(get(handles.XRobot,'String'));
      Yr=str2double(get(handles.YRobot,'String'));
      Tr=str2double(get(handles.TRobot,'String'));
      handles.Alpha=-(rad2deg(atan2((YT-Yr),(XT-Xr)))-(-Tr));
%hitung heading error
      handles.Distance=sqrt(((XT-Xr)*(XT-Xr))+((YT-Yr)*(YT-Yr)));
%hitung jarak target
      guidata(hObject, handles);
      if(handles.Distance > 2) %jika target blom tercapai
        set(handles.cek,'String',handles.count);
        JRoda=evalin('base','JRoda'); %2dm=200mm
        hRoda=evalin('base','hRoda');
        %data matrix kecepatan motor
        AA=evalin('base','AA'); BB=evalin('base','BB');
        CC=evalin('base','CC'); DD=evalin('base','DD');
        %inisialisasi pertama
        VRobot=evalin('base','VRobot');
        maxV=evalin('base','maxV');
        ytR= 0;
        ytL= 0;
        HasilFuzzy=0;
        %%%%%%%%%%%%%%%%%%%%%%%%%2SENSOR%%%%%%%%%%%%%%%%%%%
        if get(handles.sensor2,'Value') ==
1InputSensorBe19=str2double(get(handles.Sonar19,'String'));
InputSensorBe20=str2double(get(handles.Sonar20,'String'));
InputSensorBe21=str2double(get(handles.Sonar21,'String'));
InputSensorBe22=str2double(get(handles.Sonar22,'String'));
InputSensorBe23=str2double(get(handles.Sonar23,'String'));
InputSensorBe24=str2double(get(handles.Sonar24,'String'));
          InputSensorBe1=str2double(get(handles.Sonar1,'String'));
          InputSensorBe2=str2double(get(handles.Sonar2,'String'));
          InputSensorBe3=str2double(get(handles.Sonar3,'String'));
          InputSensorBe4=str2double(get(handles.Sonar4,'String'));
          InputSensorBe5=str2double(get(handles.Sonar5,'String'));
          InputSensorBe6=str2double(get(handles.Sonar6,'String'));
SensorDetect2=[InputSensorBe19,InputSensorBe20,InputSensorBe21,Inp
utSensorBe22,InputSensorBe23,InputSensorBe24,InputSensorBe1];
          minSens2=min(SensorDetect2);
SensorDetect4=[InputSensorBe24,InputSensorBe1,InputSensorBe2,Input
SensorBe3,InputSensorBe4,InputSensorBe5,InputSensorBe6];
          minSens4=min(SensorDetect4);
          if (minSens2 < 15 && minSens4 < 15 &&
get(handles.radiobutton2,'Value') == 1) ||
str2double(get(handles.kondisiEscape,'String')) > 0 %rad2 testing
%kondisi khusus perlu aksi yang unik ketika semua region
mendeteksi obstacle escape program guys
              set(handles.khususBox,'BackgroundColor','red');
              %%%EscapeAlgorithm(hObject);
              TungguT =
str2double(get(handles.delayEscape,'String'));
              KondisiT =
str2double(get(handles.kondisiEscape,'String'));
              if KondisiT == 0
                KondisiT=1;
                set(handles.kondisiEscape,'String',1);
              elseif KondisiT ==
1InputSensorBe10=str2double(get(handles.Sonar10,'String'));
InputSensorBe11=str2double(get(handles.Sonar11,'String'));
```

```
InputSensorBe12=str2double(get(handles.Sonar12,'String'));
InputSensorBe13=str2double(get(handles.Sonar13,'String'));
InputSensorBe14=str2double(get(handles.Sonar14,'String'));
InputSensorBe15=str2double(get(handles.Sonar15,'String'));
                if InputSensorBe10 > 10 && InputSensorBe11 > 10 &&
InputSensorBe12 > 10 && InputSensorBe13 > 10 && InputSensorBe14 >
10 && InputSensorBe15 > 10
                  TungguT=TungguT+1;
                  set(handles.delayEscape,'String',TungguT);
                  ytR= -0.5;
                  ytL= -0.5;
                  if TungguT>=8
                    KondisiT=2;
                    set(handles.kondisiEscape,'String',2);
                    set(handles.delayEscape,'String',0);
                  end
                else
                  TungguT=TungguT+1;
                  set(handles.delayEscape,'String',TungguT);
                  ytR= 0;
                  ytL= 0;
                  if TungguT>=2
                    KondisiT=0;
                    set(handles.kondisiEscape,'String',0);
                    set(handles.delayEscape,'String',0);
                  end
                end
              elseif KondisiT==2

InputSensorBe3=str2double(get(handles.Sonar3,'String'));
InputSensorBe4=str2double(get(handles.Sonar4,'String'));
InputSensorBe5=str2double(get(handles.Sonar5,'String'));
InputSensorBe6=str2double(get(handles.Sonar6,'String'));
InputSensorBe7=str2double(get(handles.Sonar7,'String'));
InputSensorBe8=str2double(get(handles.Sonar8,'String'));
InputSensorBe9=str2double(get(handles.Sonar9,'String'));
InputSensorBe10=str2double(get(handles.Sonar10,'String'));
InputSensorBe22=str2double(get(handles.Sonar22,'String'));
InputSensorBe21=str2double(get(handles.Sonar21,'String'));
InputSensorBe20=str2double(get(handles.Sonar20,'String'));
InputSensorBe19=str2double(get(handles.Sonar19,'String'));
InputSensorBe18=str2double(get(handles.Sonar18,'String'));
InputSensorBe17=str2double(get(handles.Sonar17,'String'));
InputSensorBe16=str2double(get(handles.Sonar16,'String'));
InputSensorBe15=str2double(get(handles.Sonar15,'String'));
                if InputSensorBe3 > 10 && InputSensorBe4 > 10 &&
InputSensorBe5 > 10 && InputSensorBe6 > 10 && InputSensorBe7 > 10
&& InputSensorBe8 > 10 && InputSensorBe9 > 10 && InputSensorBe10 >
10
                  TungguT=TungguT+1;
                  set(handles.delayEscape,'String',TungguT);
                  ytR= 20;
                  ytL= -20;
                  if TungguT>=18;
                    KondisiT=3;
                    set(handles.kondisiEscape,'String',3);
                    set(handles.delayEscape,'String',0);
                  end
```

```
                elseif InputSensorBe15 > 10 && InputSensorBe16 >
10 && InputSensorBe17 > 10 && InputSensorBe18 > 10 &&
InputSensorBe19 > 10 && InputSensorBe20 > 10 && InputSensorBe21 >
10 && InputSensorBe22 > 10
                    TungguT=TungguT+1;
                    set(handles.delayEscape,'String',TungguT);
                    ytR= -20;
                    ytL= 20;
                    if TungguT>=18;
                      KondisiT=3;
                      set(handles.kondisiEscape,'String',3);
                      set(handles.delayEscape,'String',0);
                    end
                  else
                    TungguT=TungguT+1;
                    set(handles.delayEscape,'String',TungguT);
                    ytR= 0;
                    ytL= 0;
                    if TungguT>=2
                      KondisiT=0;
                      set(handles.kondisiEscape,'String',0);
                      set(handles.delayEscape,'String',0);
                    end
                  end
                elseif KondisiT == 3
                    TungguT=TungguT+1;
                    set(handles.delayEscape,'String',TungguT);
                    ytR= 1;
                    ytL= 1;
                    if TungguT>=8
                      KondisiT=0;
                      set(handles.kondisiEscape,'String',0);
                      set(handles.delayEscape,'String',0);
                    end
                else
                    ytR= 0;
                    ytL= 0;
                    KondisiT=0;
                    set(handles.kondisiEscape,'String',0);
                    set(handles.delayEscape,'String',0);
                end
            %FQL
            else
                set(handles.khususBox,'BackgroundColor','green');
                fuzzTarget(hObject);
                handles=guidata(hObject);
                UpdateRuleColor(hObject);
                guidata(hObject, handles);
                handles=guidata(hObject);
                %cari nilai bobot rule dengan mengalikan bobot hasil
learning
                InputSensor1=str2double(get(handles.S2,'String'));
                InputSensor3=str2double(get(handles.S4,'String'));
                %%%%%%%%%%%%menggunakan substract%%%%%%%%%%%%%%%%%
                handles=guidata(hObject);
                handles.Rule(1) = handles.Hasil(1)*InputSensor1;
                handles.Rule(2) = handles.Hasil(1)*InputSensor3;
                handles.Rule(3) = handles.Hasil(2)*InputSensor1;
```

```
                handles.Rule(4) = handles.Hasil(2)*InputSensor3;
                handles.Rule(5) = handles.Hasil(3)*InputSensor1;
                handles.Rule(6) = handles.Hasil(3)*InputSensor3;
                handles.Rule(7) = handles.Hasil(4)*InputSensor1;
                handles.Rule(8) = handles.Hasil(4)*InputSensor3;
                handles.Rule(9) = handles.Hasil(5)*InputSensor1;
                handles.Rule(10) = handles.Hasil(5)*InputSensor3;
                %tampilkan bobot rule
                Data=[handles.Bo2,handles.Bo4,...
                      handles.Bo7,handles.Bo9,...
                      handles.Bo12,handles.Bo14,...
                      handles.Bo17,handles.Bo19,...
                      handles.Bo22,handles.Bo24];
                guidata(hObject, handles);
                handles=guidata(hObject);
                for i=1:10
                  set(Data(i),'String',handles.Rule(i));
                end
                %hitung output fuzzy
                OutMax=[];
                OutRule=[-20 -10 0 10 20]; %nilai u Puncak triangle
                %nilai out masing-masing bobot
                handles.QTable2S=get(handles.Tabel,'data');
                handles.Aksi2S=get(handles.TabelAksi,'data');
                for i=1:10

OutMax(i)=(handles.Rule(i)*OutRule(handles.Aksi2S(i)));
                end
                TotalOut=0;
                TotalBobot=0;
                %bobot output 5 data NB NB Z PS PB
                for i=1:10
                  TotalOut=TotalOut+OutMax(i);
                  TotalBobot=TotalBobot+handles.Rule(i);
                end
                %%%%%%%%%%%%%%%%%%%%%%55
                set(handles.OutNB,'string',TotalOut);
                set(handles.OutNS,'string',TotalBobot);
                if TotalBobot ~= 0
                    OutFuzzy=TotalOut/TotalBobot; %menggunakan COA
                else
                    OutFuzzy=0;
                end
                GainFuzzy=evalin('base','GainFuzzy');
                HasilFuzzy=OutFuzzy*GainFuzzy;
                set(handles.OutFuzzyBox,'string',HasilFuzzy);
                %hitung kecepatan masing-masing motor Point Tracking
                ytR= AA*VRobot + BB*(2*(handles.Alpha+HasilFuzzy));
                ytL= CC*VRobot + DD*(2*(handles.Alpha+HasilFuzzy));
            end
            %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

        %%%%%%%%%%%%%%%%%%%%%%%%%%%%3SENSOR%%%%%%%%%%%%%%%%%%%%%%
        elseif get(handles.sensor3,'Value') == 1

InputSensorBe17=str2double(get(handles.Sonar17,'String'));
InputSensorBe18=str2double(get(handles.Sonar18,'String'));
```

```
InputSensorBe19=str2double(get(handles.Sonar19,'String'));
InputSensorBe20=str2double(get(handles.Sonar20,'String'));
InputSensorBe21=str2double(get(handles.Sonar21,'String'));
InputSensorBe22=str2double(get(handles.Sonar22,'String'));
InputSensorBe23=str2double(get(handles.Sonar23,'String'));
InputSensorBe24=str2double(get(handles.Sonar24,'String'));
          InputSensorBe1=str2double(get(handles.Sonar1,'String'));
          InputSensorBe2=str2double(get(handles.Sonar2,'String'));
          InputSensorBe3=str2double(get(handles.Sonar3,'String'));
          InputSensorBe4=str2double(get(handles.Sonar4,'String'));
          InputSensorBe5=str2double(get(handles.Sonar5,'String'));
          InputSensorBe6=str2double(get(handles.Sonar6,'String'));
          InputSensorBe7=str2double(get(handles.Sonar7,'String'));
          InputSensorBe8=str2double(get(handles.Sonar8,'String'));
SensorDetect2=[InputSensorBe17,InputSensorBe18,InputSensorBe19,Inp
utSensorBe20,InputSensorBe21,InputSensorBe22,InputSensorBe23];
          minSens2=min(SensorDetect2);
SensorDetect3=[InputSensorBe20,InputSensorBe21,InputSensorBe22,Inp
utSensorBe23,InputSensorBe24,InputSensorBe1,InputSensorBe2,InputSe
nsorBe3,InputSensorBe4,InputSensorBe5];
          minSens3=min(SensorDetect3);
SensorDetect4=[InputSensorBe2,InputSensorBe3,InputSensorBe4,InputS
ensorBe5,InputSensorBe6,InputSensorBe7,InputSensorBe8];
          minSens4=min(SensorDetect4);
          if (minSens2 < 15 && minSens3<15 && minSens4 < 15 &&
get(handles.radiobutton2,'Value') == 1) ||
str2double(get(handles.kondisiEscape,'String')) > 0 %rad2 testing
%kondisi khusus perlu aksi yang unik ketika semua region
mendeteksi obstacle escape program guys
              set(handles.khususBox,'BackgroundColor','red');
              %%%EscapeAlgorithm(hObject);
              TungguT =
str2double(get(handles.delayEscape,'String'));
              KondisiT =
str2double(get(handles.kondisiEscape,'String'));
              if KondisiT == 0
                KondisiT=1;
                set(handles.kondisiEscape,'String',1);
              elseif KondisiT ==
1InputSensorBe10=str2double(get(handles.Sonar10,'String'));
InputSensorBe11=str2double(get(handles.Sonar11,'String'));
InputSensorBe12=str2double(get(handles.Sonar12,'String'));
InputSensorBe13=str2double(get(handles.Sonar13,'String'));
InputSensorBe14=str2double(get(handles.Sonar14,'String'));
InputSensorBe15=str2double(get(handles.Sonar15,'String'));
                if InputSensorBe10 > 10 && InputSensorBe11 > 10 &&
InputSensorBe12 > 10 && InputSensorBe13 > 10 && InputSensorBe14 >
10 && InputSensorBe15 > 10
                  TungguT=TungguT+1;
                  set(handles.delayEscape,'String',TungguT);
                  ytR= -0.5;
                  ytL= -0.5;
                  if TungguT>=8
                    KondisiT=2;
                    set(handles.kondisiEscape,'String',2);
                    set(handles.delayEscape,'String',0);
                  end
                else
```

```
                    TungguT=TungguT+1;
                    set(handles.delayEscape,'String',TungguT);
                    ytR= 0;
                    ytL= 0;
                    if TungguT>=2
                      KondisiT=0;
                      set(handles.kondisiEscape,'String',0);
                      set(handles.delayEscape,'String',0);
                    end
                  end
                elseif KondisiT==2;
InputSensorBe3=str2double(get(handles.Sonar3,'String'));
InputSensorBe4=str2double(get(handles.Sonar4,'String'));
InputSensorBe5=str2double(get(handles.Sonar5,'String'));
InputSensorBe6=str2double(get(handles.Sonar6,'String'));
InputSensorBe7=str2double(get(handles.Sonar7,'String'));
InputSensorBe8=str2double(get(handles.Sonar8,'String'));
InputSensorBe9=str2double(get(handles.Sonar9,'String'));
InputSensorBe10=str2double(get(handles.Sonar10,'String'));
InputSensorBe22=str2double(get(handles.Sonar22,'String'));
InputSensorBe21=str2double(get(handles.Sonar21,'String'));
InputSensorBe20=str2double(get(handles.Sonar20,'String'));
InputSensorBe19=str2double(get(handles.Sonar19,'String'));
InputSensorBe18=str2double(get(handles.Sonar18,'String'));
InputSensorBe17=str2double(get(handles.Sonar17,'String'));
InputSensorBe16=str2double(get(handles.Sonar16,'String'));
InputSensorBe15=str2double(get(handles.Sonar15,'String'));
                  if InputSensorBe3 > 10 && InputSensorBe4 > 10 &&
InputSensorBe5 > 10 && InputSensorBe6 > 10 && InputSensorBe7 > 10
&& InputSensorBe8 > 10 && InputSensorBe9 > 10 && InputSensorBe10 >
10
                    TungguT=TungguT+1;
                    set(handles.delayEscape,'String',TungguT);
                    ytR= 20;
                    ytL= -20;
                    if TungguT>=18;
                      KondisiT=3;
                      set(handles.kondisiEscape,'String',3);
                      set(handles.delayEscape,'String',0);
                    end
                  elseif InputSensorBe15 > 10 && InputSensorBe16 >
10 && InputSensorBe17 > 10 && InputSensorBe18 > 10 &&
InputSensorBe19 > 10 && InputSensorBe20 > 10 && InputSensorBe21 >
10 && InputSensorBe22 > 10
                    TungguT=TungguT+1;
                    set(handles.delayEscape,'String',TungguT);
                    ytR= -20;
                    ytL= 20;
                    if TungguT>=18;
                      KondisiT=3;
                      set(handles.kondisiEscape,'String',3);
                      set(handles.delayEscape,'String',0);
                    end
                  else
                    TungguT=TungguT+1;
                    set(handles.delayEscape,'String',TungguT);
                    ytR= 0;
```

```
                    ytL= 0;
                    if TungguT>=2
                      KondisiT=0;
                      set(handles.kondisiEscape,'String',0);
                      set(handles.delayEscape,'String',0);
                    end
                  end
                elseif KondisiT == 3

                    TungguT=TungguT+1;
                    set(handles.delayEscape,'String',TungguT);
                    ytR= 1;
                    ytL= 1;
                    if TungguT>=8
                      KondisiT=0;
                      set(handles.kondisiEscape,'String',0);
                      set(handles.delayEscape,'String',0);
                    end

                else
                    ytR= 0;
                    ytL= 0;
                    KondisiT=0;
                    set(handles.kondisiEscape,'String',0);
                    set(handles.delayEscape,'String',0);
                end
            %FQL
            else
                set(handles.khususBox,'BackgroundColor','green');
                fuzzTarget(hObject);
                handles=guidata(hObject);
                UpdateRuleColor(hObject);
                guidata(hObject, handles);
                handles=guidata(hObject);
                %cari nilai bobot rule dengan mengalikan bobot hasil
learning
                InputSensor1=str2double(get(handles.S2,'String'));
                InputSensor2=str2double(get(handles.S3,'String'));
                InputSensor3=str2double(get(handles.S4,'String'));
                %%%%%%%%%%%%%menggunakan substract%%%%%%%%%%%%%%%%%
                handles=guidata(hObject);
                handles.Rule(1) = handles.Hasil(1)*InputSensor1;
                handles.Rule(2) = handles.Hasil(1)*InputSensor2;
                handles.Rule(3) = handles.Hasil(1)*InputSensor3;
                handles.Rule(4) = handles.Hasil(2)*InputSensor1;
                handles.Rule(5) = handles.Hasil(2)*InputSensor2;
                handles.Rule(6) = handles.Hasil(2)*InputSensor3;
                handles.Rule(7) = handles.Hasil(3)*InputSensor1;
                handles.Rule(8) = handles.Hasil(3)*InputSensor2;
                handles.Rule(9) = handles.Hasil(3)*InputSensor3;
                handles.Rule(10) = handles.Hasil(4)*InputSensor1;
                handles.Rule(11) = handles.Hasil(4)*InputSensor2;
                handles.Rule(12) = handles.Hasil(4)*InputSensor3;
                handles.Rule(13) = handles.Hasil(5)*InputSensor1;
                handles.Rule(14) = handles.Hasil(5)*InputSensor2;
                handles.Rule(15) = handles.Hasil(5)*InputSensor3;
                %tampilkan bobot rule
```

```
                Data=[handles.Bo2,handles.Bo3,handles.Bo4,...
                      handles.Bo7,handles.Bo8,handles.Bo9,...
                      handles.Bo12,handles.Bo13,handles.Bo14,...
                      handles.Bo17,handles.Bo18,handles.Bo19,...
                      handles.Bo22,handles.Bo23,handles.Bo24];
                guidata(hObject, handles);
                handles=guidata(hObject);
                for i=1:15
                  set(Data(i),'String',handles.Rule(i));
                end
                %hitung output fuzzy
                OutMax=[];
                OutRule=[-20 -10 0 10 20]; %nilai u Puncak triangle
                %nilai out masing-masing bobot
                handles.QTable3S=get(handles.Tabel,'data');
                handles.Aksi3S=get(handles.TabelAksi,'data');
                for i=1:15

OutMax(i)=(handles.Rule(i)*OutRule(handles.Aksi3S(i)));
                end
                TotalOut=0;
                TotalBobot=0;
                %bobot output 5 data NB NB Z PS PB
                for i=1:15
                  TotalOut=TotalOut+OutMax(i);
                  TotalBobot=TotalBobot+handles.Rule(i);
                end
                %%%%%%%%%%%%%%%%%%%%%%55
                set(handles.OutNB,'string',TotalOut);
                set(handles.OutNS,'string',TotalBobot);
                if TotalBobot ~= 0
                    OutFuzzy=TotalOut/TotalBobot; %menggunakan COA
                else
                    OutFuzzy=0;
                end
                GainFuzzy=evalin('base','GainFuzzy');
                HasilFuzzy=OutFuzzy*GainFuzzy;
                set(handles.OutFuzzyBox,'string',HasilFuzzy);
                %hitung kecepatan masing-masing motor Point Tracking
                ytR= AA*VRobot + BB*(2*(handles.Alpha+HasilFuzzy));
                ytL= CC*VRobot + DD*(2*(handles.Alpha+HasilFuzzy));
            end
            %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

          %%%%%%%%%%%%%%%%%%%%%%%%%%%4SENSOR%%%%%%%%%%%%%%%%%%%%%
          elseif get(handles.sensor4,'Value') == 1
InputSensorBe16=str2double(get(handles.Sonar16,'String'));
InputSensorBe17=str2double(get(handles.Sonar17,'String'));
InputSensorBe18=str2double(get(handles.Sonar18,'String'));
InputSensorBe19=str2double(get(handles.Sonar19,'String'));
InputSensorBe20=str2double(get(handles.Sonar20,'String'));
InputSensorBe21=str2double(get(handles.Sonar21,'String'));
InputSensorBe22=str2double(get(handles.Sonar22,'String'));
InputSensorBe23=str2double(get(handles.Sonar23,'String'));
InputSensorBe24=str2double(get(handles.Sonar24,'String'));
            InputSensorBe1=str2double(get(handles.Sonar1,'String'));
            InputSensorBe2=str2double(get(handles.Sonar2,'String'));
```

```
          InputSensorBe3=str2double(get(handles.Sonar3,'String'));
          InputSensorBe4=str2double(get(handles.Sonar4,'String'));
          InputSensorBe5=str2double(get(handles.Sonar5,'String'));
          InputSensorBe6=str2double(get(handles.Sonar6,'String'));
          InputSensorBe7=str2double(get(handles.Sonar7,'String'));
          InputSensorBe8=str2double(get(handles.Sonar8,'String'));
          InputSensorBe9=str2double(get(handles.Sonar9,'String'));
SensorDetect1=[InputSensorBe16,InputSensorBe17,InputSensorBe18,Inp
utSensorBe19,InputSensorBe20,InputSensorBe21];
          minSens1=min(SensorDetect1);
SensorDetect2=[InputSensorBe20,InputSensorBe21,InputSensorBe22,Inp
utSensorBe23,InputSensorBe24,InputSensorBe1];
          minSens2=min(SensorDetect2);
SensorDetect4=[InputSensorBe24,InputSensorBe1,InputSensorBe2,Input
SensorBe3,InputSensorBe4,InputSensorBe5];
          minSens4=min(SensorDetect4);
SensorDetect5=[InputSensorBe4,InputSensorBe5,InputSensorBe6,InputS
ensorBe7,InputSensorBe8,InputSensorBe9];
          minSens5=min(SensorDetect5);
          if (minSens1 < 15 && minSens2<15 && minSens4 < 15 &&
minSens5 < 15 && get(handles.radiobutton2,'Value') == 1) ||
str2double(get(handles.kondisiEscape,'String')) > 0 %rad2 testing
%kondisi khusus perlu aksi yang unik ketika semua region
mendeteksi obstacle escape program guys
              set(handles.khususBox,'BackgroundColor','red');
              %%%EscapeAlgorithm(hObject);
              TungguT =
str2double(get(handles.delayEscape,'String'));
              KondisiT =
str2double(get(handles.kondisiEscape,'String'));
              if KondisiT == 0
                KondisiT=1;
                set(handles.kondisiEscape,'String',1);
              elseif KondisiT == 1
InputSensorBe10=str2double(get(handles.Sonar10,'String'));
InputSensorBe11=str2double(get(handles.Sonar11,'String'));
InputSensorBe12=str2double(get(handles.Sonar12,'String'));
InputSensorBe13=str2double(get(handles.Sonar13,'String'));
InputSensorBe14=str2double(get(handles.Sonar14,'String'));
InputSensorBe15=str2double(get(handles.Sonar15,'String'));
                if InputSensorBe10 > 10 && InputSensorBe11 > 10 &&
InputSensorBe12 > 10 && InputSensorBe13 > 10 && InputSensorBe14 >
10 && InputSensorBe15 > 10
                  TungguT=TungguT+1;
                  set(handles.delayEscape,'String',TungguT);
                  ytR= -0.5;
                  ytL= -0.5;
                  if TungguT>=8
                    KondisiT=2;
                    set(handles.kondisiEscape,'String',2);
                    set(handles.delayEscape,'String',0);
                  end
                else
                  TungguT=TungguT+1;
                  set(handles.delayEscape,'String',TungguT);
                  ytR= 0;
                  ytL= 0;
                  if TungguT>=2
```

```
                    KondisiT=0;
                    set(handles.kondisiEscape,'String',0);
                    set(handles.delayEscape,'String',0);
                  end
                end
              elseif KondisiT==2
InputSensorBe3=str2double(get(handles.Sonar3,'String'));
InputSensorBe4=str2double(get(handles.Sonar4,'String'));
InputSensorBe5=str2double(get(handles.Sonar5,'String'));
InputSensorBe6=str2double(get(handles.Sonar6,'String'));
InputSensorBe7=str2double(get(handles.Sonar7,'String'));
InputSensorBe8=str2double(get(handles.Sonar8,'String'));
InputSensorBe9=str2double(get(handles.Sonar9,'String'));
InputSensorBe10=str2double(get(handles.Sonar10,'String'));
InputSensorBe22=str2double(get(handles.Sonar22,'String'));
InputSensorBe21=str2double(get(handles.Sonar21,'String'));
InputSensorBe20=str2double(get(handles.Sonar20,'String'));
InputSensorBe19=str2double(get(handles.Sonar19,'String'));
InputSensorBe18=str2double(get(handles.Sonar18,'String'));
InputSensorBe17=str2double(get(handles.Sonar17,'String'));
InputSensorBe16=str2double(get(handles.Sonar16,'String'));
InputSensorBe15=str2double(get(handles.Sonar15,'String'));
                if InputSensorBe3 > 10 && InputSensorBe4 > 10 &&
InputSensorBe5 > 10 && InputSensorBe6 > 10 && InputSensorBe7 > 10
&& InputSensorBe8 > 10 && InputSensorBe9 > 10 && InputSensorBe10 >
10
                  TungguT=TungguT+1;
                  set(handles.delayEscape,'String',TungguT);
                  ytR= 20;
                  ytL= -20;
                  if TungguT>=18;
                    KondisiT=3;
                    set(handles.kondisiEscape,'String',3);
                    set(handles.delayEscape,'String',0);
                  end
                elseif InputSensorBe15 > 10 && InputSensorBe16 >
10 && InputSensorBe17 > 10 && InputSensorBe18 > 10 &&
InputSensorBe19 > 10 && InputSensorBe20 > 10 && InputSensorBe21 >
10 && InputSensorBe22 > 10
                  TungguT=TungguT+1;
                  set(handles.delayEscape,'String',TungguT);
                  ytR= -20;
                  ytL= 20;
                  if TungguT>=18;
                    KondisiT=3;
                    set(handles.kondisiEscape,'String',3);
                    set(handles.delayEscape,'String',0);
                  end
                else
                  TungguT=TungguT+1;
                  set(handles.delayEscape,'String',TungguT);
                  ytR= 0;
                  ytL= 0;
                  if TungguT>=2
                    KondisiT=0;
                    set(handles.kondisiEscape,'String',0);
                    set(handles.delayEscape,'String',0);
                  end
```

```
                end
            elseif KondisiT == 3
                TungguT=TungguT+1;
                set(handles.delayEscape,'String',TungguT);
                ytR= 1;
                ytL= 1;
                if TungguT>=8
                  KondisiT=0;
                  set(handles.kondisiEscape,'String',0);
                  set(handles.delayEscape,'String',0);
                end
            else
                ytR= 0;
                ytL= 0;
                KondisiT=0;
                set(handles.kondisiEscape,'String',0);
                set(handles.delayEscape,'String',0);
            end
        %FQL
        else
            set(handles.khususBox,'BackgroundColor','green');
            fuzzTarget(hObject);
            handles=guidata(hObject);
            UpdateRuleColor(hObject);
            guidata(hObject, handles);
            handles=guidata(hObject);
            %cari nilai bobot rule dengan mengalikan bobot hasil
learning
            InputSensor1=str2double(get(handles.S1,'String'));
            InputSensor2=str2double(get(handles.S2,'String'));
            InputSensor4=str2double(get(handles.S4,'String'));
            InputSensor5=str2double(get(handles.S5,'String'));
            %%%%%%%%%%%%%menggunakan substract%%%%%%%%%%%%%%%%%
            handles=guidata(hObject);
            handles.Rule(1) = handles.Hasil(1)*InputSensor1;
            handles.Rule(2) = handles.Hasil(1)*InputSensor2;
            handles.Rule(3) = handles.Hasil(1)*InputSensor4;
            handles.Rule(4) = handles.Hasil(1)*InputSensor5;
            handles.Rule(5) = handles.Hasil(2)*InputSensor1;
            handles.Rule(6) = handles.Hasil(2)*InputSensor2;
            handles.Rule(7) = handles.Hasil(2)*InputSensor4;
            handles.Rule(8) = handles.Hasil(2)*InputSensor5;
            handles.Rule(9) = handles.Hasil(3)*InputSensor1;
            handles.Rule(10) = handles.Hasil(3)*InputSensor2;
            handles.Rule(11) = handles.Hasil(3)*InputSensor4;
            handles.Rule(12) = handles.Hasil(3)*InputSensor5;
            handles.Rule(13) = handles.Hasil(4)*InputSensor1;
            handles.Rule(14) = handles.Hasil(4)*InputSensor2;
            handles.Rule(15) = handles.Hasil(4)*InputSensor4;
            handles.Rule(16) = handles.Hasil(4)*InputSensor5;
            handles.Rule(17) = handles.Hasil(5)*InputSensor1;
            handles.Rule(18) = handles.Hasil(5)*InputSensor2;
            handles.Rule(19) = handles.Hasil(5)*InputSensor4;
            handles.Rule(20) = handles.Hasil(5)*InputSensor5;
            %tampilkan bobot rule
Data=[handles.Bo1,handles.Bo2,handles.Bo4,handles.Bo5,...
handles.Bo6,handles.Bo7,handles.Bo9,handles.Bo10,...
handles.Bo11,handles.Bo12,handles.Bo14,handles.Bo15,...
```

```
handles.Bo16,handles.Bo17,handles.Bo19,handles.Bo20,...
handles.Bo21,handles.Bo22,handles.Bo24,handles.Bo25];
              guidata(hObject, handles);
              handles=guidata(hObject);
              for i=1:20
                set(Data(i),'String',handles.Rule(i));
              end
              %hitung output fuzzy
              OutMax=[];
              OutRule=[-20 -10 0 10 20]; %nilai u Puncak triangle
              %nilai out masing-masing bobot
              handles.QTable4S=get(handles.Tabel,'data');
              handles.Aksi4S=get(handles.TabelAksi,'data');
              for i=1:20
OutMax(i)=(handles.Rule(i)*OutRule(handles.Aksi4S(i)));
              end
              TotalOut=0;
              TotalBobot=0;
              %bobot output 5 data NB NB Z PS PB
              for i=1:20
                TotalOut=TotalOut+OutMax(i);
                TotalBobot=TotalBobot+handles.Rule(i);
              end
              %%%%%%%%%%%%%%%%%%%%%55
              set(handles.OutNB,'string',TotalOut);
              set(handles.OutNS,'string',TotalBobot);
              if TotalBobot ~= 0
                  OutFuzzy=TotalOut/TotalBobot; %menggunakan COA
              else
                  OutFuzzy=0;
              end
              GainFuzzy=evalin('base','GainFuzzy');
              HasilFuzzy=OutFuzzy*GainFuzzy;
              set(handles.OutFuzzyBox,'string',HasilFuzzy);
              %hitung kecepatan masing-masing motor Point Tracking
              ytR= AA*VRobot + BB*(2*(handles.Alpha+HasilFuzzy));
              ytL= CC*VRobot + DD*(2*(handles.Alpha+HasilFuzzy));
          end
          %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        %%%%%%%%%%%%%%%%%%%%%%%%%%%%5SENSOR%%%%%%%%%%%%%%%%%%%%%%
        elseif get(handles.sensor5,'Value') == 1
InputSensorBe14=str2double(get(handles.Sonar14,'String'));
InputSensorBe15=str2double(get(handles.Sonar15,'String'));
InputSensorBe16=str2double(get(handles.Sonar16,'String'));
InputSensorBe17=str2double(get(handles.Sonar17,'String'));
InputSensorBe18=str2double(get(handles.Sonar18,'String'));
InputSensorBe19=str2double(get(handles.Sonar19,'String'));
InputSensorBe20=str2double(get(handles.Sonar20,'String'));
InputSensorBe21=str2double(get(handles.Sonar21,'String'));
InputSensorBe22=str2double(get(handles.Sonar22,'String'));
InputSensorBe23=str2double(get(handles.Sonar23,'String'));
InputSensorBe24=str2double(get(handles.Sonar24,'String'));
          InputSensorBe1=str2double(get(handles.Sonar1,'String'));
          InputSensorBe2=str2double(get(handles.Sonar2,'String'));
          InputSensorBe3=str2double(get(handles.Sonar3,'String'));
          InputSensorBe4=str2double(get(handles.Sonar4,'String'));
          InputSensorBe5=str2double(get(handles.Sonar5,'String'));
          InputSensorBe6=str2double(get(handles.Sonar6,'String'));
```

```
          InputSensorBe7=str2double(get(handles.Sonar7,'String'));
          InputSensorBe8=str2double(get(handles.Sonar8,'String'));
          InputSensorBe9=str2double(get(handles.Sonar9,'String'));
InputSensorBe10=str2double(get(handles.Sonar10,'String'));
InputSensorBe11=str2double(get(handles.Sonar11,'String'));
SensorDetect1=[InputSensorBe14,InputSensorBe15,InputSensorBe16,Inp
utSensorBe17,InputSensorBe18,InputSensorBe19];
          minSens1=min(SensorDetect1);
SensorDetect2=[InputSensorBe18,InputSensorBe19,InputSensorBe20,Inp
utSensorBe21,InputSensorBe22,InputSensorBe23];
          minSens2=min(SensorDetect2);
SensorDetect3=[InputSensorBe22,InputSensorBe23,InputSensorBe24,Inp
utSensorBe1,InputSensorBe2,InputSensorBe3];
          minSens3=min(SensorDetect3);
SensorDetect4=[InputSensorBe2,InputSensorBe3,InputSensorBe4,InputS
ensorBe5,InputSensorBe6,InputSensorBe7];
          minSens4=min(SensorDetect4);

SensorDetect5=[InputSensorBe6,InputSensorBe7,InputSensorBe8,InputS
ensorBe9,InputSensorBe10,InputSensorBe11];
          minSens5=min(SensorDetect5);
          if (minSens2 < 15 && minSens3<15 && minSens4 < 15 &&
get(handles.radiobutton2,'Value') == 1) ||
str2double(get(handles.kondisiEscape,'String')) > 0 %rad2 testing
%kondisi khusus perlu aksi yang unik ketika semua region
mendeteksi obstacle escape program guys
              set(handles.khususBox,'BackgroundColor','red');
              %%%EscapeAlgorithm(hObject);
              TungguT =
str2double(get(handles.delayEscape,'String'));
              KondisiT =
str2double(get(handles.kondisiEscape,'String'));
              if KondisiT == 0
                KondisiT=1;
                set(handles.kondisiEscape,'String',1);
              elseif KondisiT == 1
InputSensorBe10=str2double(get(handles.Sonar10,'String'));
InputSensorBe11=str2double(get(handles.Sonar11,'String'));
InputSensorBe12=str2double(get(handles.Sonar12,'String'));
InputSensorBe13=str2double(get(handles.Sonar13,'String'));
InputSensorBe14=str2double(get(handles.Sonar14,'String'));
InputSensorBe15=str2double(get(handles.Sonar15,'String'));
                if InputSensorBe10 > 10 && InputSensorBe11 > 10 &&
InputSensorBe12 > 10 && InputSensorBe13 > 10 && InputSensorBe14 >
10 && InputSensorBe15 > 10
                  TungguT=TungguT+1;
                  set(handles.delayEscape,'String',TungguT);
                  ytR= -0.5;
                  ytL= -0.5;
                  if TungguT>=8
                    KondisiT=2;
                    set(handles.kondisiEscape,'String',2);
                    set(handles.delayEscape,'String',0);
                  end
                else
                  TungguT=TungguT+1;
                  set(handles.delayEscape,'String',TungguT);
                  ytR= 0;
```

```
                    ytL= 0;
                    if TungguT>=2
                      KondisiT=0;
                      set(handles.kondisiEscape,'String',0);
                      set(handles.delayEscape,'String',0);
                    end
                  end
                elseif KondisiT==2
InputSensorBe3=str2double(get(handles.Sonar3,'String'));
InputSensorBe4=str2double(get(handles.Sonar4,'String'));
InputSensorBe5=str2double(get(handles.Sonar5,'String'));
InputSensorBe6=str2double(get(handles.Sonar6,'String'));
InputSensorBe7=str2double(get(handles.Sonar7,'String'));
InputSensorBe8=str2double(get(handles.Sonar8,'String'));
InputSensorBe9=str2double(get(handles.Sonar9,'String'));
InputSensorBe10=str2double(get(handles.Sonar10,'String'));
InputSensorBe22=str2double(get(handles.Sonar22,'String'));
InputSensorBe21=str2double(get(handles.Sonar21,'String'));
InputSensorBe20=str2double(get(handles.Sonar20,'String'));
InputSensorBe19=str2double(get(handles.Sonar19,'String'));
InputSensorBe18=str2double(get(handles.Sonar18,'String'));
InputSensorBe17=str2double(get(handles.Sonar17,'String'));
InputSensorBe16=str2double(get(handles.Sonar16,'String'));
InputSensorBe15=str2double(get(handles.Sonar15,'String'));
                  if InputSensorBe3 > 10 && InputSensorBe4 > 10 &&
InputSensorBe5 > 10 && InputSensorBe6 > 10 && InputSensorBe7 > 10
&& InputSensorBe8 > 10 && InputSensorBe9 > 10 && InputSensorBe10 >
10
                    TungguT=TungguT+1;
                    set(handles.delayEscape,'String',TungguT);
                    ytR= 20;
                    ytL= -20;
                    if TungguT>=18;
                      KondisiT=3;
                      set(handles.kondisiEscape,'String',3);
                      set(handles.delayEscape,'String',0);
                    end
                  elseif InputSensorBe15 > 10 && InputSensorBe16 >
10 && InputSensorBe17 > 10 && InputSensorBe18 > 10 &&
InputSensorBe19 > 10 && InputSensorBe20 > 10 && InputSensorBe21 >
10 && InputSensorBe22 > 10
                    TungguT=TungguT+1;
                    set(handles.delayEscape,'String',TungguT);
                    ytR= -20;
                    ytL= 20;
                    if TungguT>=18;
                      KondisiT=3;
                      set(handles.kondisiEscape,'String',3);
                      set(handles.delayEscape,'String',0);
                    end
                  else
                    TungguT=TungguT+1;
                    set(handles.delayEscape,'String',TungguT);
                    ytR= 0;
                    ytL= 0;
                    if TungguT>=2
                      KondisiT=0;
                      set(handles.kondisiEscape,'String',0);
```

```
                    set(handles.delayEscape,'String',0);
                end
              end
            elseif KondisiT == 3

                TungguT=TungguT+1;
                set(handles.delayEscape,'String',TungguT);
                ytR= 1;
                ytL= 1;
                if TungguT>=8
                  KondisiT=0;
                  set(handles.kondisiEscape,'String',0);
                  set(handles.delayEscape,'String',0);
                end
            else
                ytR= 0;
                ytL= 0;
                KondisiT=0;
                set(handles.kondisiEscape,'String',0);
                set(handles.delayEscape,'String',0);
            end
        %FQL
        else
            set(handles.khususBox,'BackgroundColor','green');
            fuzzTarget(hObject);
            handles=guidata(hObject);
            UpdateRuleColor(hObject);
            guidata(hObject, handles);
            handles=guidata(hObject);
            %cari nilai bobot rule dengan mengalikan bobot hasil
learning
            InputSensor1=str2double(get(handles.S1,'String'));
            InputSensor2=str2double(get(handles.S2,'String'));
            InputSensor3=str2double(get(handles.S3,'String'));
            InputSensor4=str2double(get(handles.S4,'String'));
            InputSensor5=str2double(get(handles.S5,'String'));
            %%%%%%%%%%%%%menggunakan substract%%%%%%%%%%%%%%%%%
            handles=guidata(hObject);
            handles.Rule(1) = handles.Hasil(1)*InputSensor1;
            handles.Rule(2) = handles.Hasil(1)*InputSensor2;
            handles.Rule(3) = handles.Hasil(1)*InputSensor3;
            handles.Rule(4) = handles.Hasil(1)*InputSensor4;
            handles.Rule(5) = handles.Hasil(1)*InputSensor5;
            handles.Rule(6) = handles.Hasil(2)*InputSensor1;
            handles.Rule(7) = handles.Hasil(2)*InputSensor2;
            handles.Rule(8) = handles.Hasil(2)*InputSensor3;
            handles.Rule(9) = handles.Hasil(2)*InputSensor4;
            handles.Rule(10) = handles.Hasil(2)*InputSensor5;
            handles.Rule(11) = handles.Hasil(3)*InputSensor1;
            handles.Rule(12) = handles.Hasil(3)*InputSensor2;
            handles.Rule(13) = handles.Hasil(3)*InputSensor3;
            handles.Rule(14) = handles.Hasil(3)*InputSensor4;
            handles.Rule(15) = handles.Hasil(3)*InputSensor5;
            handles.Rule(16) = handles.Hasil(4)*InputSensor1;
            handles.Rule(17) = handles.Hasil(4)*InputSensor2;
            handles.Rule(18) = handles.Hasil(4)*InputSensor3;
            handles.Rule(19) = handles.Hasil(4)*InputSensor4;
```

```
                handles.Rule(20) = handles.Hasil(4)*InputSensor5;
                handles.Rule(21) = handles.Hasil(5)*InputSensor1;
                handles.Rule(22) = handles.Hasil(5)*InputSensor2;
                handles.Rule(23) = handles.Hasil(5)*InputSensor3;
                handles.Rule(24) = handles.Hasil(5)*InputSensor4;
                handles.Rule(25) = handles.Hasil(5)*InputSensor5;
                %tampilkan bobot rule
Data=[handles.Bo1,handles.Bo2,handles.Bo3,handles.Bo4,handles.Bo5,
...
handles.Bo6,handles.Bo7,handles.Bo8,handles.Bo9,handles.Bo10,...
handles.Bo11,handles.Bo12,handles.Bo13,handles.Bo14,handles.Bo15,.
..
handles.Bo16,handles.Bo17,handles.Bo18,handles.Bo19,handles.Bo20,.
..
handles.Bo21,handles.Bo22,handles.Bo23,handles.Bo24,handles.Bo25];
                guidata(hObject, handles);
                handles=guidata(hObject);
                for i=1:25
                  set(Data(i),'String',handles.Rule(i));
                end
                %hitung output fuzzy
                OutMax=[];
                OutRule=[-20 -10 0 10 20]; %nilai u Puncak triangle
                %nilai out masing-masing bobot
                handles.QTable5S=get(handles.Tabel,'data');
                handles.Aksi5S=get(handles.TabelAksi,'data');
                for i=1:25

OutMax(i)=(handles.Rule(i)*OutRule(handles.Aksi5S(i)));
                end
                TotalOut=0;
                TotalBobot=0;
                %bobot output 5 data NB NB Z PS PB
                for i=1:25
                  TotalOut=TotalOut+OutMax(i);
                  TotalBobot=TotalBobot+handles.Rule(i);
                end
                %%%%%%%%%%%%%%%%%%%%%%55
                set(handles.OutNB,'string',TotalOut);
                set(handles.OutNS,'string',TotalBobot);
                if TotalBobot ~= 0
                    OutFuzzy=TotalOut/TotalBobot; %menggunakan COA
                else
                    OutFuzzy=0;
                end
                GainFuzzy=evalin('base','GainFuzzy');
                HasilFuzzy=OutFuzzy*GainFuzzy;
                set(handles.OutFuzzyBox,'string',HasilFuzzy);
                %hitung kecepatan masing-masing motor Point Tracking
                ytR= AA*VRobot + BB*(2*(handles.Alpha+HasilFuzzy));
                ytL= CC*VRobot + DD*(2*(handles.Alpha+HasilFuzzy));
            end
            %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
          %%%%%%%%%%%%%%%%%%%%%%%%%%%3SENSOR%%%%%%%%%%%%%%%%%%%%%%
          end
          %%%%%%%%%%%%%%%%%%%%PEMETAAN ROBOT%%%%%%%%%%%%%%%%%
          if HasilFuzzy == 0
set(handles.DA1,'String',str2double(get(handles.DH1,'String')));
```

```
set(handles.DA2,'String',str2double(get(handles.DH2,'String')));
        end
        %maksimum kecepatan motor
        if abs(ytR) == abs(ytL)
           if abs(ytR) > maxV
               if ytR > 0
                 ytR = maxV;
               else
                 ytR = -maxV;
               end
           end
           if abs(ytL) > maxV
               if ytL > 0
                 ytL = maxV;
               else
                 ytL = -maxV;
               end
           end
        elseif abs(ytL) > abs(ytR)
           if abs(ytL) > maxV
               ytR = ytR/(abs(ytL)/maxV);
               if ytL > 0
                 ytL = maxV;
               else
                 ytL = -maxV;
               end
           end
        else
           if abs(ytR) > maxV
               ytL = ytL/(abs(ytR)/maxV);
               if ytR > 0
                 ytR = maxV;
               else
                 ytR = -maxV;
               end
           end
        end
        %data matrix posisi
        PAA=JRoda/2;     PBB=JRoda/2;
        PCC=JRoda/2;     PDD=JRoda/2;
        PEE=JRoda/hRoda; PFF=-JRoda/hRoda;
        %update posisi Robot
        xdot=((ytR*cos(deg2rad(-Tr)))*PAA) + ((ytL*cos(deg2rad(-
Tr)))*PBB);
        ydot=(ytR*sin(deg2rad(-Tr)))*PCC + ytL*sin(deg2rad(-
Tr))*PDD;
        tdot=(ytR*PEE) + (ytL*PFF);
        set(handles.XRobot,'string',Xr+xdot);
        set(handles.YRobot,'string',Yr+ydot);
        set(handles.TRobot,'string',Tr+tdot);
      end
guidata(hObject, handles);
```

Halaman ini sengaja dikosongkan

BIODATA PENULIS

Arga Dwi Pambudi, putra kedua dari tiga bersaudara dari pasangan Gamu Rianto dan Arminingsih. Penulis Dilahirkan pada tanggal 01 Juni 1990 di kabupaten malang. Penulis memulai pendidikan di SDN Pagentan 1 Singosari, SMPN 1 Singosari, SMAN 5 Malang, PENS Surabaya, UDINUS Semarang. Pada tahun 2016 penulis melanjutkan studi di bidang keahlian Teknik Sistem Pengaturan. Penulis dapat dihubungi melaui email *arga.dwi.pambudi@gmail.com*.

Halaman ini sengaja dikosongkan