TUGAS AKHIR TE-141599

PERENCANAAN JALUR *MOBILE ROBOT* QBOT MENGGUNAKAN METODE B-SPLINE DENGAN KONTROLER *NEURO FUZZY*

Dwi Wulandari
NRP 2214105056

Dosen Pembimbing
Ir. Joko Susila, MT.

JURUSAN TEKNIK ELEKTRO
Fakultas Teknologi Industri
Institut Teknologi Sepuluh Nopember
Surabaya 2016

FINAL PROJECT **TE-141599**

PATH PLANNING MOBILE ROBOT QBOT USING B-SPLINE METHOD WITH NEURO FUZZY CONTROLLER

Dwi Wulandari
NRP 2214105056

Supervisor
Ir. Joko Susila, MT.

DEPARTMENT OF ELECTRICAL ENGINEERING
Faculty of Industrial Technology
Sepuluh Nopember Institute of Technology
Surabaya 2016

PERENCANAAN JALUR *MOBILE ROBOT*QBOT MENGGUNAKAN METODE*B-SPLINE* DENGAN KONTROLER*NEURO-FUZZY*

TUGAS AKHIR

Diajukan Guna Memenuhi Sebagian Persyaratan
Untuk Memperoleh Gelar SarjanaTeknik
Pada
Bidang Studi Teknik Sistem Pengaturan
Jurusan Teknik Elektro
Institut Teknologi Sepuluh Nopember

Menyetujui:

Dosen Pembimbing,

Ir. Joko Susila M.T.
NIP.196606061991021001

SURABAYA
JUNI,2016

PERENCANAAN JALUR *MOBILE ROBOT* QBOT MENGGUNAKAN METODE *B-SPLINE* DENGAN KONTROLER *NEURO-FUZZY*

Nama	**: Dwi Wulandari**
NRP	**: 2214105056**
Dosen Pembimbing	**: Ir. Joko Susila, MT.**
NIP	**: 196606061991021001**

ABSTRAK

Mobile robot adalah sebuah mesin otomatis yang mampu bergerak pada suatu kondisi tertentu. Pengenalan lingkungan sekitar *mobile robot* merupakan hal penting agar *mobile robot* dapat menjalankan misinya. Dalam hal ini perencanaan jalur dilakukan untuk mendapatkan informasi rute yang akan dilalui *Mobile Robot* dari posisi awal ke posisi akhir yang telah ditentukan. Untuk kontroler yang digunakan yaitu *neuro-fuzzy* agar Qbot *mobile robot* dapat mengikuti referensi jalur yang telah ditentukan menggunakan metode *B-spline*. Metode *B-spline* ini juga digunakan agar pergerakan *mobile robot* lebih *smooth* saat menghindari *obstacle*. Hasil pengujian kontroler *neuro fuzzy* pada Qbot *mobile robot* untuk *training* pola 1 waktu yang dibutuhkan untuk mencapai target adalah 34,2 detik untuk bergerak dari titik (0,0)cm menuju titik (0,275)cm, sedangkan waktu yang dibutuhkan kontroler *neuro fuzzy* pada percobaan pertama adalah 32,02 detik, percobaan kedua 32,31 detik dan percobaan ketiga 30,92 detik. Untuk pola 2 saat *training* membutuhkan waktu 39,32 detik untuk bergerak dari titik (0,0)cm menuju titik (0,275)cm, sedangkan kontroler *neuro fuzzy* pada percobaan pertama adalah 37,67 detik, percobaan kedua 36,74 detik dan percobaan ketiga 36,54 detik. Untuk perbedaan respon sinyal kontrol *neuro fuzzy* dari tiga kali kali percobaan disebabkan faktor eksternal seperti kondisi kalibrasi sensor dan lingkungan yang dapat mempengaruhi sensor.

Kata Kunci : *B-spline*, *Neuro-fuzzy*, perencanaan jalur, navigasi, kontroler, Qbot *mobile robot* dan *training*

PATH PLANNING MOBILE ROBOT QBOT USING B-SPLINE METHOD WITH NEURO FUZZY CONTROLLER

Name	**: Dwi Wulandari**
Register Number	**: 2214105056**
Supervisor	**: Ir. Joko Susila, MT.**
ID Number	**: 196606061991021001**

ABSTRACT

Mobile robot is an automated machine that is able to move in a certain condition. The introduction of mobile robot environment is an important thing that the mobile robot can carry out its mission. In this case the path planning is done to get the information that will be passed Mobile Robot from an initial position to a final position has been determined. For controllers used are neuro-fuzzy Qbot mobile robot that can follow a predetermined path references using B-spline. Results of testing the neuro fuzzy controller on Qbot mobile robot for training to move from the point (0, 0) cm towards the point (0, 275) cm, while the time required neuro fuzzy controller on the first try is 32.02 seconds, the second experiment 32.31 seconds and the third experiment 30.92 sec. For pattern 2 when training takes 39.32 seconds to move from the point (0, 0) cm towards the point (0, 275) cm, while the neuro fuzzy controller on the first try is 37.67 seconds, the second experiment 36.74 seconds and 36.54 seconds third trial. For the difference in response neuro fuzzy control signals from three trials due to external factors such as sensor calibration and environmental conditions that can affect the sensor.

Keyword : *B-spline, Neuro fuzzy, path planning, navigation, controller, Qbot mobile robot and training*

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

BAB I
PENDAHULUAN

Pada bab satu akan dibahas mengenai latar belakang, permasalahan, tujuan, metodologi, sistematika, dan relevansi tugas akhir yang dikerjakan.

1.1 Latar Belakang

Perkembangan teknologi yang semakin pesat mendorong manusia untuk senantiasa menciptakan inovasi-inovasi guna mempermudah pekerjaan. Salah satu inovasi yang terus menerus dikembangkan adalah dalam bidang robotika. Penggunaan robot telah diteliti secara ekstensif dan diharapkan dapat diterapkan dalam kehidupan sehari-hari manusia.

Mobile robot adalah sebuah mesin otomatis yang mampu bergerak pada suatu kondisi tertentu. Tetapi terdapat beberapa kendala pada sistem yang menggunakan metode berdasarkan masukan sensor saat akan menghindari *obstacle*.

Pada umumnya dalam merancang suatu sistem kendali harus diketahui terlebih dahulu parameter-parameter dari sistem yang akan dikendalikan. Oleh karena itu, perlu adanya proses identifikasi terlebih dahulu dari sistem yang akan dikendalikan secara akurat. Tetapi pada kenyataanya seringkali parameter sistem tersebut sulit ditentukan, baik karena kompleksitas sistem maupun kondisi dinamik sistem (proses). Sebuah pendekatan dalam pengendalian sistem yang parameter-parameternya tidak diketahui atau sulit untuk ditentukan dapat dilakukan dengan menggunakan jaringan syaraf tiruan. Dengan jaringan syaraf tiruan maka parameter–parameter sistem dapat ditentukan tanpa adanya proses identifikasi terhadap sistem yang akan dikendalikan. *B-spline* merupakan salah satu jenis jaringan syaraf tiruan yang dapat digunakan sebagai komponen pengendali sistem yang parameter-parameternya tidak diketahui atau sulit ditentukan tanpa adanya proses identifikasi terlebih dahulu.

Pada tugas akhir ini dilakukan perencanaan rute *Mobile Robot* untuk mendapatkan informasi jalur yang akan dilalui dari posisi awal ke posisi akhir yang telah ditentukan. *Mobile Robot* yang digunakan yaitu Quanser Qbot yang merupakan robot *autonomous* yang terdiri dari *platform* iRobot yang banyak digunakan dalam aplikasi robot. Perencanaan jalur pada Qbot *mobile robot* dilakukan dengan cara

training menggunakan metode *B-spline* agar robot dapat menentukan titik awal robot hingga dapat bergerak menuju titik akhirnya. Untuk itu akan didesain perencanaan jalur *Mobile Robot* QBOT menggunakan Metode *B-spline* dengan kontroler *Neuro-Fuzzy* agar memperoleh pergerakan *mobile robot* lebih *smooth* dan sesuai dengan yang diinginkan.

1.2 Permasalahan

Dalam pergerakan *mobile robot* diperlukan sebuah navigasi untuk mengendalikan pergerakan robot dari titik awal menuju tujuan akhir. Permasalahan yang dibahas pada tugas akhir ini adalah bagaimana *mobile robot* Qbot mengikuti jalur yang telah ditentukan di mana pada jalur yang akan ditempuh tersebut terdapat *obstacles* dan robot dapat lebih *smooth* saat menghindari *obstacles* tersebut. Oleh karena itu dibuat sebuah kontroler *neuro fuzzy* agar robot dapat mengikuti referensi jalur dan dapat lebih *smooth* saat menghindari *obstacles*.

1.3 Tujuan

Tujuan dari penelitian tugas akhir ini yaitu agar *mobile robot* dapat bergerak lebih *smooth* saat menghindari *obstacle* dalam mengikuti refrensi jalur yang telah ditentukan menggunakan metode *B-spline* dengan kontroler *Neuro-Fuzzy.*

1.4 Metodologi

Metodologi yang digunakan pada penelitian tugas akhir ini adalah sebagai berikut:

1. Tinjauan Pustaka
   Dilakukan dengan mengumpulkan dan mempelajari literatur yang mendukung tugas akhir melalui *internet* dan media cetak berupa buku atau jurnal.
2. Pengambilan Data dan Identifikasi
   Pada tahap ini akan dilakukan pengambilan data dari Qbot *Mobile robot*. Setelah memperoleh data yang dibutuhkan, akan dilakukan identifikasi dari *plant* untuk memperoleh parameter yang akan digunakan untuk mendesain kontroler sesuai yang diinginkan.
3. Perancangan Kontroler

Pada tahap ini dibuat struktur kontroler *neuro-fuzzy* dengan metode *B-spline* untuk navigasi Qbot *mobile robot* dan kontroler disimulasikan secara *real time*.

4. Implementasi dan Uji Coba
   Kontroler yang sudah dirancang lalu diimplementasikan pada *plant* berupa Qbot *Mobile robot* pada ruang yang sudah direncanakan jalurnya dan terdapat *obstacles*
5. Penulisan Buku Tugas Akhir
   Penyusunan buku Tugas Akhir meliputi pendahuluan, teori dasar, perancangan sistem, implementasi dan analisa serta penutup.

1.5 Sistematika

Pembahasan Tugas Akhir ini dibagi menjadi lima bab dengan sistematika sebagai berikut :

Bab 1 : Pendahuluan
Bab ini meliputi latar belakang, permasalahan, tujuan penelitian, dan sistematika penulisan.

Bab 2 : Teori Penunjang
Bab ini membahas tinjauan pustaka yang membantu penelitian, diantaranya teori Qbot *mobile robot*, teori identifikasi sistem kontroler *neuro- fuzzy* dan Metode *B-spline*

Bab 3 : Perancangaan Sistem
Pada bab ini dijelaskan mengenai perancangan jalur yang akan dilalui *mobile robot* Qbot dengan menggunakan metode *B-spline* serta identifikasi sistem Qbot *mobile robot* serta kontroler *neuro fuzzy*

Bab 4 : Hasil dan Analisa
Bab ini memuat hasil implementasi kontroler dan pengujiannya pada sistem

Bab 5 : Penutup
Bab ini berisi kesimpulan dan saran dari hasil penelitian yang telah dilakukan

1.6 Relevansi

Hasil yang diperoleh dari tugas akhir ini diharapkan menjadi referensi pengembangan teknologi dan perbandingan metode kontrol yang tepat untuk navigasi Qbot *mobile robot* dimasa mendatang.

BAB II
TEORI PENUNJANG

Pada bab dua akan dibahas mengenai tinjauan pustaka dan teori-teori yang berhubungan dengan permasalahan yang dibahas pada bab sebelumnya.

2.1 Tinjauan Pustaka

Robot merupakan perangkat mekanik yang dapat dikendalikan oleh perangkat lunak yang menggunakan sensor untuk memandu satu atau lebih efektor melalui gerakan terprogram pada suatu ruang kerja dalam hal untuk manipulasi obyek fisik. *Mobile robot* adalah sebuah mesin otomatis yang mampu bergerak pada suatu kondisi tertentu. Tetapi terdapat beberapa kendala pada sistem yang menggunakan metode berdasarkan masukan sensor saat akan menghindari *obstacle*.

Pada penelitian Qbot *mobile robot* sebelumnya yang dilakukan oleh Afrizal dalam tugas akhirnya telah dibahas mengenai Implementasi Perencanaan Jalur *Mobile Robot* Qbot Menggunakan Metode *Neuro-Fuzzy*. Dalam penelitian Tugas Akhir kali ini, akan dibuat perencanaan jalur robot dari titik awal menuju tujuan akhir yang mampu menghindari *obstacles* di mana untuk perencanaan jalur akan menggunakan metode *neuro fuzzy* sebagai kontroler untuk navigasi robot dengan metode *B-spline* agar pergerakan robot dapat lebih *smooth* saat menghindari *obstacle*.

Perencanaan jalur pada Qbot *mobile robot* dilakukan dengan cara *training* menggunakan metode *B-spline* agar robot dapat menentukan titik awal robot hingga dapat bergerak menuju titik akhirnya. Pada perencanaan jalur yang akan dilakukan oleh Qbot *mobile robot* ditentukan terlebih dahulu jarak yang akan ditempuh robot pada bidang (x,y), saat Qbot *mobile robot* bergerak *obstacles* yang berada pada jalur akan terbaca oleh sensor jarak dan robot akan menghindarinya dengan pergerakan yang *smooth* kemudian melanjutkan pergerakan menuju titik posisi tujuan.

Untuk navigasi pergerakan Qbot *mobile robot* ini digunakan metode kontrol *neuro fuzzy* di mana kontroler ini untuk mengendalikan kecepatan pergerakan robot untuk mencapai titik tujuan akhirnya dan dapat menghindari *obstacles* yang berada pada jalur yang akan dilalui Qbot *mobile robot*.

Halaman ini sengaja dikosongkan

2.2 Qbot *Mobile Robot* [1]

Qbot *Mobile robot* merupakan sistem inovatif robot darat *autonomous* yang menggabungkan ilmu kendaraan darat dengan *Quanser Controller Module* (QCM). *Mobile robot* ini terdiri dari iRobot Create® *Robotic Platform*, sensor inframerah dan sonar serta sebuah kamera Logitech Quickcam Pro 9000 USB seperti terlihat pada Gambar 2.1. QCM yang terpasang pada Qbot yang mana menggunakan komputer Gumstix untuk menjalankan QuaRC, perangkat lunak kontrol Quanser, dan Qbot kartu data akuisisi (DAC).

Gambar 2.1 Qbot *Mobile Robot* [1]

Berikut ini spesifikasi dan model parameter Qbot *mobile robot* dapat dilihat pada Tabel 2.1 berikut:

Tabel 2.1 Spesifikasi Sistem dan Parameter Model Qbot *Mobile Robot*

Simbol	Deskripsi	Value	Unit
D	Diameter Qbot *mobile robot*	0,34	m
H	Tinggi Qbot *mobile robot* (dengan tambahan kamera)	0,19	m
vmax	Kecepatan maksimum dari Qbot *mobile robot*	0,5	m/s
M	Berat total Qbot *mobile robot*	2,92	kg

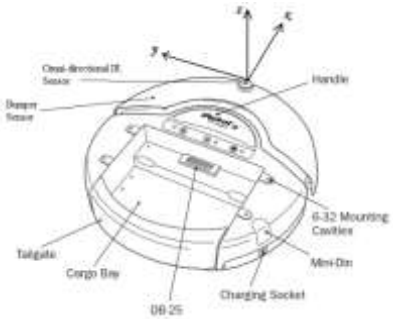
Antarmuka untuk QCM adalah MATLAB Simulink dengan QuaRC. Qbot dapat menerima perintah, melalui tiga blok yang berbeda: blok Roomba untuk menggerakkan Qbot, blok HIL untuk membaca dari sensor dan/atau menulis untuk *output* servo dan blok *Open CV* untuk mengakses kamera. Kontroler dikembangkan pada Simulink dengan

QuaRC dan model ini diunduh dan disusun pada target (Gumstix). Diagram dari konfigurasi ini ditampilkan dalam Gambar 2.2.

Gambar 2.2 Hirarki Komunikasi Qbot *Mobile Robot* [1]

2.2.1 Sistem Perangkat Keras Qbot

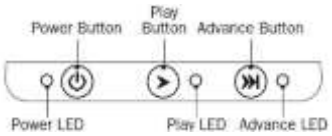
Qbot menggunakan rangka iRobot Create® seperti pada Gambar 2.3. Qbot mengikuti standar Quanser untuk sumbu rangka tubuh, di mana sumbu x pada arah maju, sumbu y sebelah kiri dan sumbu z ada di atas. Diameter dari rangka ini 34 cm dan tinggi (tanpa tambahan kamera) adalah 7 cm. dan Qbot digerakkan oleh dua roda kemudi yang berbeda. iRobot Create® terdapat *bumper* sensor dan inframerah. QCN dapat mengakses data dari sensor ini.



Gambar 2.3 Rangka Qbot *Mobile Robot* [1]

Qbot dihidupkan dengan menekan tombol *Power*. Gambar 2.4 menunjukkan tombol yang digunakan untuk mengoperasikan Qbot.

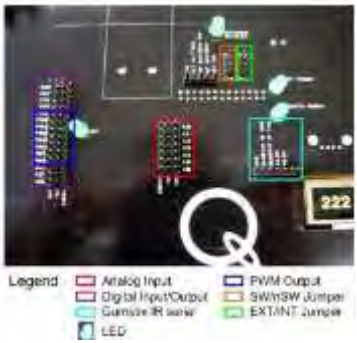
Tombol *Play* dan *Advance* untuk menjalankan dalam demo untuk Qbot dan tidak diperlukan untuk tujuan Qbot.



Gambar 2.4 Tombol pada Rangka Qbot [1]

2.2.2 *Printed Circuit Board* (PCB)

Kabel dan sirkuit untuk Qbot dalam PCB yang terletak di penutup hitam Qbot atas komputer Gumstix dan DAC. Sensor dan kamera yang juga terpasang pada PCB. Gambar 2.5 menunjukkan pin diakses bagi pengguna. Secara khusus, DIO, *output* PWM, dan pin *input* analog telah diberi label untuk lebih jelas.



Gambar 2.5 PCB Qbot

2.2.2.1 *Pin Digital Input/Output (DIO)*

Saluran DIO (0 samapai 6) ditetapkan sebgai *input* secara bawaan. Saluran DIO perlu dikonfigurasi baik sebagai *input* (atau *ouput*, tapi tidak keduanya) menggunakan blok HIL *Initialize*. Jika *output* harus dalam keadaan yang dikenal *power up,* dianjurkan bahwa resistor 10K diletakkan dari I/O untuk 5V atau GND sesuai kebutuhan. Ada saluran digital akhir (7) yang merupakan *output* tetap, dan itu diwakili oleh LED berlabel DIO7.

2.2.2.2 Pin Serial Gumstix IR

Qbot menyediakan serial koneksi TTL ke *port* serial Gumstix IR (*port* no 2). *Port* serial terdiri dari *ground* (GND), menerima Gumstix IR RXD, mengirimkan Gumstix IR TXD dan pin daya (+3.3V atau +5.0V). *Port* serial diakses melalui blok QuaRC *Stream* atau *Stream* API.

2.2.2.3 SW/nSW dan INT/EXT Jumpers

INT/EXT *jumper switch* Qbot dari internal daya dari baterai iRobot Create (INT) dan eksternal baterai *power supply* (EXT). Tidak ada baterai eksternal yang disertakan pada Qbot, jadi *jumper* ini harus dibiarkan dalam posisi INT untuk daya Qbot tersebut. Saat *jumper power supply* dalam posisi INT, jumper SW/ NSW menunjukkan apakah iRobot Create harus diaktifkan (SW) untuk Qbot menerima daya atau apakah Qbot harus selalu membutuhkan daya bahkan ketika iRobot Create dalam keadaan mati (nSW).

2.2.3 Komponen *Mobile Robot*

Mobile robot ini terdiri dari iRobot Create® *Robotic Platform*, sensor inframerah dan sonar serta sebuah kamera Logitech Quickcam Pro 9000 USB.

2.2.3.1 Qbot Data Acquisition Card (DAC)

Qbot DAC adalah kartu data akuisisi, yang mampu menerima *input* analog dan *input* lainnya (untuk sensor sonar). Qbot DAC dapat juga membaca *output* PWM untuk servo aktuatuor. Qbot DAC terletak di bawah penutup hitam Qbot.

2.2.3.2 Kamera USB

Kamera USB Logitech Quickcam Pro 9000 terpasang pada bagian atas Qbot seperti pada Gambar 2.6. Blok QuaRC menggunakan *Open Source Computer Vision* yang menyediakan pengguna untuk mengambil dan menampilkan gambar secara *real time*, memproses dan menyimpannya untuk analisis.

Gambar 2.6 Kamera USB Logitech Quickcam 9000 [1]

2.2.3.3 Gumstix

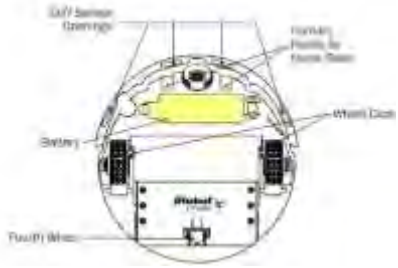
Gumstix skala kecil berfungsi penuh pada komputer *open source* di mana Simulink MATLAB secara langsung dapat diunduh, dikompilasi dan dijalankan melalui perangkat lunak QuaRC. *Motherboard* Gumstix terhubung secara langsung pada Qbot DAC. Pada Gumstix juga terdapat tambahan Wifi untuk menyediakan koneksi nirkabel antara target Gumstix dan komputer *host*.

2.2.3.4 Baterai Qbot

Qbot ini didukung oleh baterai *Advance Power System* (APS) yang disediakan oleh iRobot seperti pada Gambar 2.7. Baterai ini terletak di bawah Qbot dan dapat berlangsung terus menerus selama sekitar dua jam setelah terisi penuh.

Gambar 2.7 Baterai Qbot [1]

Baterai terletak di bagian bawah seperti pada Gambar 2.8. Lampu daya Qbot mengindikasikan tingkat daya baterai. Lampu hijau menandakan baterai masih terisi penuh, kemudian secara bertahap berubah menjadi merah bila baterai akan habis. Baterai memerlukan waktu kurang dari 3 jam untuk pengisian baterai. Ketika pengisian, lampu daya akan berkedip perlahan dengan warna jingga dan akan berubah menjadi hijau ketika terisi penuh.



Gambar 2.8 Letak Baterai pada Qbot [1]

2.2.3.5 Sensor Inframerah dan Sonar

SHARP 2Y0A02 seperti pada Gambar 2.9 merupakan sensor inframerah dengan jarak 20-150cm. Ada lima sensor inframerah yang terpasang pada Qbot. Sensor terhubung kesaluran *input* analog dari Qbot DAC, yang kemudian dapat dibaca dengan menggunakan blok HIL *Read Write.*

Gambar 2.9 Sensor Inframerah SHARP 2Y0A02 [1]

MaxSonar-EZ0 seperti pada Gambar 2.10 mendeteksi objek dari 0 inci sampai 254 inci dengan resolusi 1 inci. Objek antara 0 inci dan 6 inci berkisar sebagai 6 inci. Terdapat tiga sensor pada Qbot. Sensor yang terhubung ke saluran masukan lain dari Qbot DAC, yang kemudian dapat dibaca dengan menggunakan blok HIL *Read Write.*

Gambar 2.10 Sensor Sonar MaxSonar-EZ0 [1]

2.2.4 QUARC

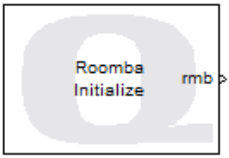
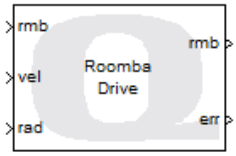
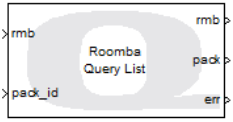
QUARC *(Quanser Real-Time Control)* adalah perangkat lunak yang menyediakan fasilitas untuk mengembangkan dan menguji coba rancangan kontroler pada komputer *host* dengan menggunakan perangkat lunak Simulink MATLAB, dengan model yang dirancang dapat langsung diunduh dan dieksekusi pada Gumstix dengan menggunakan komunikasi

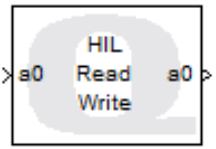
wireless secara *real time*. Pada waktu yang bersamaan operator dapat memantau dan mengukur nilai-nilai dari sensor dan mengatur parameter-parameter kontroler langsung dari komputer *host*. Pada MATLAB Simulink terdapat beberapa blok yang digunakan untuk melakukan komunikasi dengan Qbot seperti yang terdapat pada Tabel 2.2 dan Tabel 2.3.

Tabel 2.2 Deskripsi Blok *Set* Qbot yang Relevan untuk Qbot di QUARC

Blok *Set*	**Deskripsi**
Interface	Blok set ini mengimplementasikan dasar *Aplication Program Interfaces* (API) yang disediakan oleh iRobot Create®. API dapat dikategorikan berdasarkan fungsional berikut: 1. Sambungkan konfigurasi serial antara kontroler tingkat tinggi (misalnya, PC atau Gumstix) dan Qbot. 2. Pengaturan *mode* operasi Qbot 3. Mengakses informasi sensorik Qbot 4. Konfigurasi *hardware* lainnya (misalnya, pengaturan *port* I/O digital dan analog, dan mengubah warna LED)
Aplikasi	Blok set ini memungkinkan pengguna untuk menerapkan algoritma navigasi menggunakan informasi sensorik yang tersedia. Blok ini hanya menggunakan data *encoder* roda dan *bump sensors* untuk navigasi dan menghindari rintangan.
Image Processing	Blok set ini mengimplementasikan akuisisi citra dan pengolahan fungsi menggunakan *Library Open CV*. Blok set pengolahan citra memungkinkan untuk menangkap gambar dari kamera USB, untuk proses *online* dan menyimpannya ke *disk* untuk analisis lebih lanjut.

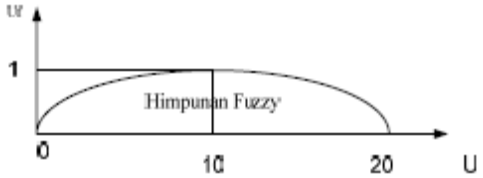
Tabel 2.3 Penjelasan Deskripsi Blok QUARC pada Qbot

Blok	Deskripsi
	Blok ini diperlukan untuk membuat sambungan serial ke Qbot. Qbot diidentifikasi oleh *Universal Resource Identifier* (URI), seperti *serial://localhost:1baud=57600,word=8, parity=none,stop=1*, di mana *port* komunikasi 1 dari target terhubung ke *port* serial Qbot dengan parameter yang ditentukan. Catatan: *Output* "rmb" dari blok ini harus terhubung ke *input* "rmb" dari blok di blok Roomba yang ditetapkan dalam rangka untuk mengakses Roomba.
 Roomba Drive	Blok ini menggerakkan Qbot dengan dua *input*: kecepatan dan radius. Untuk menggerakkan lurus, *input* radius mengambil 32768 atau 32767.
 Roomba Query List	Blok ini mengambil berbagai data sensor dari Qbot. *Input pack_id* berkisar 7-42, dan setiap *input* akan menampilkan nilai dari *output pack*. Lihat halaman *Help* MATLAB untuk deskripsi lengkap untuk masing-masing paket ID. Nilai *pack_id* yang penting: 8 = Dinding (0 = tidak ada dinding, 1 = dinding terlihat) 19 = Jarak (Jarak yang Qbot telah melakukan perjalanan dalam milimeter) 20 = Sudut (Sudut dalam derajat Qbot berubah) 22 = Tegangan (Tegangan dari baterai Qbot dalam milivolt) 23 = Arus (Arus dalam miliamper mengalir atau keluar dari baterai Qbot)

Blok	Deskripsi
	Blok inisialisasi yang diperlukan dalam semua model QuaRC menggunakan *HIL Card*. Hanya satu blok HIL *Initialize* yang dibutuhkan dalam model untuk mengatur satu kendaraan. Blok HIL *Read Write* harus referensi blok ini untuk menentukan *board* yang digunakan.
	Versi saat ini dari Qbot DAC yang dapat membaca dari saluran analog (untuk sensor inframerah) dan saluran sensor sonar. Saluran PWM yang didukung untuk operasi *write.*

2.3 Logika *Fuzzy* [2]

Logika *Fuzzy* dikenalkan oleh L.A. Zadeh pada tahun 1965, dengan mengembangkan teori himpunan logika biner. Logika biner hanya mengenal dua macam kondisi yaitu "1" dan "0", sehingga terdapat batasan yang tegas. L.A. Zadeh kemudian memodifikasi teori himpunan di mana setiap anggotanya memiliki derajat keanggotaan yang bernilai kontinyu antara 0 sampai 1 atau ditulis [0 1], seperti tampak pada Gambar 2.11.

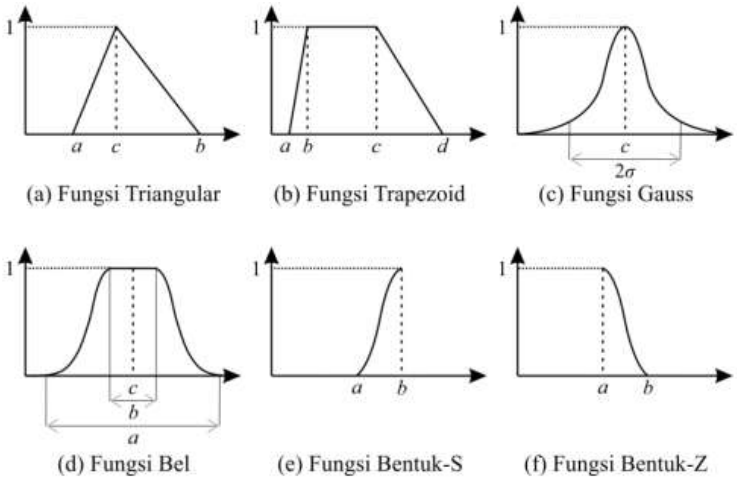


Gambar 2.11 Himpunan *Fuzzy*

Himpunan *Fuzzy* merupakan suatu himpunan yang beranggotakan sejumlah istilah dalam pengertian bahasa yang menyatakan level kualitatif dari semesta pembicaraan. Misalnya pengukuran kecepatan putaran motor dapat diterjemahkan ke dalam beberapa istilah bahasa yang menyatakan level kualitatif dari kecepatan

putaran motor. Sehingga apabila semesta pembicaraan berupa kecepatan putaran motor, maka dapat dibuat suatu himpunan *fuzzy* yaitu "Sangat Lambat", "Lambat, "Sedang", "Cepat", "Sangat Cepat".

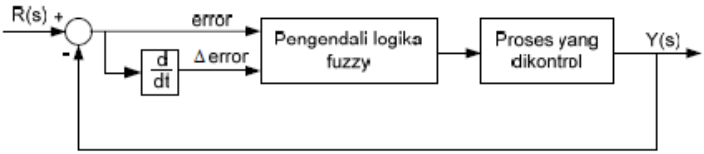
Fungsi keanggotaan *(membership function)* adalah suatu kurva yang menunjukkan pemetaan titik-titik *input* data ke dalam nilai kenggotaannya (atau sering disebut dengan derajat keanggotaan). Ada beberapa macam bentuk kurva yang sering digunakan untuk menyatakan derajat keanggotaan pada suatu sistem *Fuzzy*, antara lain bentuk kurva S (*Shape*), segitiga, trapesium, dan sebagainya. Bentuk-bentuk kurva yang sering digunakan untuk menunjukkan fungsi keanggotaan antara lain terlihat pada Gambar 2.12



Gambar 2.12 Berbagai Tipe *Membership Function* [2]

2.4 Kontroler Logika *Fuzzy* [3]

Pengendali *Fuzzy* merupakan suatu sistem kendali yang berdasarkan pada pengetahuan manusia, di mana masukan, keluaran, serta tanggapan sistem diperoleh berdasarkan pengetahuan manusia. Pengendali logika *Fuzzy* tidak memerlukan model matematis dari proses yang dikendalikan. Blok sistem loop tertutup dengan kendali logika *Fuzzy* ditunjukkan pada Gambar 2.13.



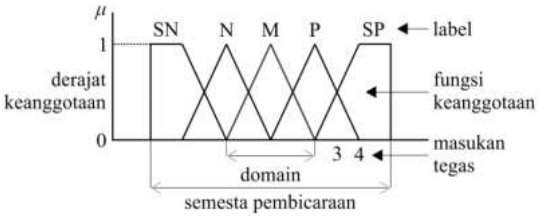
Gambar 2.13 Sistem Loop Tertutup dengan Kontroler *Fuzzy.*

Gambar 2.13 menunjukkan dua masukan (*crisp input*), yaitu masukan kesalahan (*error*) dan perubahan kesalahan (Δ*error*) yang diperoleh dari nilai referensi, keluaran *plant*, dan *error* sebelumnya. Dua masukan tersebut akan diolah oleh pengendali logika *Fuzzy*. Struktur dasar pengendali logika *Fuzzy* ditunjukkan pada Gambar 2.14, yang meliputi empat bagian utama yaitu fuzzifikasi, basis pengetahuan, logika pengambilan keputusan dan defuzzifikasi.

Gambar 2.14 Struktur Dasar Kontroler *Fuzzy*.

2.4.1 Fuzzifikasi

Fuzzifikasi diperlukan untuk mengubah masukan tegas atau nyata (*crisp inputs*) yang bersifat bukan *fuzzy* (nilai *real*) ke dalam himpunan *fuzzy*. Data yang berbentuk tegas atau nyata (*crisp*), dipetakan menjadi nilai linguistik pada semesta pembicaraan tertentu yang selanjutnya dinamakan masukan *fuzzy*.



Gambar 2.15 Struktur *Membership Function Fuzzy*

2.4.2 Basis Pengetahuan

Basis pengetahuan berisi pengetahuan sistem kendali sebagai pedoman evaluasi keadaan sistem untuk mendapatkan keluaran kendali sesuai yang diinginkan oleh perancang. Basis pengetahuan terdiri dari basis data dan basis aturan *fuzzy*.

2.4.2.1 Basis Data

Basis data mencakup perancangan fungsi keanggotaan untuk variabel masukan dan keluaran, pendefinisian semesta pembicaraan dan penentuan variabel linguistik setiap variabel masukan dan keluaran.

2.4.2.2 Basis Aturan

Basis aturan kendali *fuzzy* digunakan untuk menghubungkan variabel-variabel masukan dan variabel-variabel keluaran. Basis aturan *fuzzy* merupakan kumpulan pernyataan aturan „JIKA-MAKA" atau *'IF–THEN'* yang didasarkan pada pengetahuan manusia untuk mengolah variabel masukan sehingga menghasilkan variabel keluaran dalam bentuk himpunan *fuzzy*.

2.4.3 Mekanisme Pertimbangan *Fuzzy*

Terdapat beberapa tipe mekanisme inferensi *fuzzy* antara lain Mamdani, Larsent dan Takagi sugeno. Perbedaan dari metode ini terletak pada pengambilan kesimpulan logika *fuzzy*. Pada metode Mamdani maupun Larsent, kesimpulan logika *fuzzy* berupa derajat keanggotaan sehingga dalam menyimpulkan suatu logika *fuzzy* dibutuhkan proses defuzzifikasi. Sedangkan pada tipe Takagi Sugeno, kesimpulan logika *fuzzy* berupa suatu persamaan sehingga tidak diperlukan proses defuzzifikasi. Kelebihan pada logika *fuzzy* tipe Mamdani dan Larsent lebih sederhana, akan tetapi diperlukan kemampuan untuk mengetahui karakteristik *plant* untuk menentukan batasan keluaran kontroler. Pada tipe Takagi-Sugeno tidak diperlukan pengetahuan mengenai karakteristik dari *plant* akan tetapi diperlukan perhitungan yang lebih rumit untuk persamaan pada bagian konsekuen.

2.4.4 Defuzzifikasi

Defuzzifikasi ditujukan untuk menghasilkan suatu aksi kontrol *non fuzzy* (*crisp output*) dalam merepresentasikan kemungkinan distribusi aksi kontrol *fuzzy* yang telah dihasilkan. Metode *defuzzifikasi* yang sering digunakan adalah metode *Mean of Maximum* (MOM) dan

metode rata-rata terbobot (*weighted average*) atau lebih dikenal sebagai *Center of Area* (COA).

2.4.4.1 Center of Area

Metode *center of area* digunakan untuk menentukan nilai titik tengah area yang merupakan titik pusat massa dari kombinasi fungsi-fungsi keanggotaan. Secara umum, persamaan untuk metode *Center of Area* ditunjukkan dengan Persamaan 2.1.

$$U_0 = \frac{\sum_{k=1}^{m} u_k(T) \cdot \mu_u(u_k(T))}{\sum_{k=1}^{m} \mu_k(u_k(T))}, \forall\, u \in U(T) \tag{2.1}$$

2.4.4.2 Mean of Maximum (MOM)

Metode *mean of maxima* mengambil semua nilai tiap fungsi keanggotaan dengan derajat keanggotaan maksimum dan menghitung rata-rata dari nilai-nilai tersebut sebagai keluaran tegas. Persamaan 2.2 menunjukkan persamaan umum metode tersebut.

$$U_0 = \frac{\sum_n \max(\mu_A^n).yn}{\sum_n \max(\mu_A^n)} \tag{2.2}$$

2.4.4.3 Weighted Sum

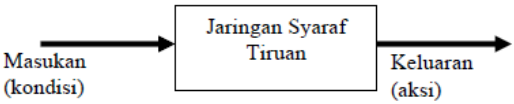
Metode *weigthed sum* merupakan metode defuzzifikasi dengan fungsi keanggotaan berupa fungsi *singleton.* Persamaan 2.3 menunjukkan persamaan defuzzifikasi fungsi *singleton*.

$$U_0 = \frac{\sum_i \mu_A^i s_i}{\sum_i \mu_A^i} \tag{2.3}$$

2.5 Jaringan Syaraf Tiruan *(Neural Network)* [4]

Jaringan syaraf tiruan merupakan salah satu representasi buatan dari otak manusia yang selalu mencoba mensimulasikan proses menyimpan, belajar, dan mengambil kembali pengetahuan yang tersimpan dalam sel syaraf atau *neuron* pada otak manusia tersebut. Jaringan syaraf tiruan ini pada dasarnya adalah fungsi pemetaan masukan keluaran sistem yang bebas model matematis atau dikenal dengan istilah *estimator* bebas model. Sistem ini memetakan kondisi ke aksi, dalam hal ini sistem-sistem dinamis yang dimodelkan tidak diekspresikan secara matematis

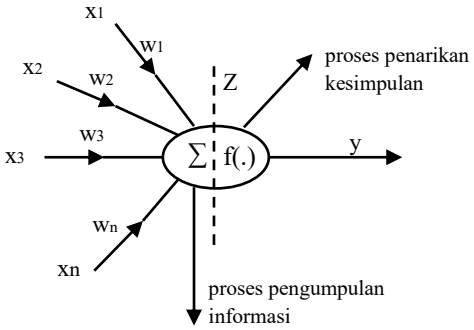
menggunakan fungsi alih tetapi direpresentasikan dengan menggunakan kotak fungsional yang mengestimasi fungsi-fungsi dari data pelatihan.



Gambar 2.16. Jaringan Syaraf Tiruan sebagai Fungsi Pemetaan

Seperti halnya otak manusia, jaringan syaraf tiruan juga terdiri dari beberapa *neuron* di mana tiap-tiap *neuron* dengan *neuron* yang lainnya saling berhubungan. *Neuron* tersebut akan memproses tiap informasi yang diterima kemudian dimodelkan dalam suatu bentuk model *neuron*.

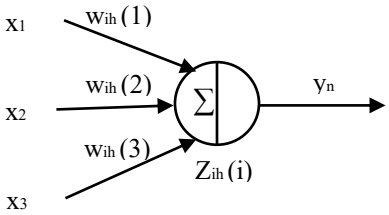
Jaringan syaraf tiruan memiliki kemampuan untuk belajar terhadap suatu kondisi lingkungan dan meningkatkan performansinya melalui pembelajaran. Proses pembelajaran jaringan dilakukan melalui proses iterasi terhadap nilai bobot dan idealnya jaringan menjadi adaptif setelah proses pembelajaran.



Gambar 2.17 Struktur Model Jaringan Saraf Tiruan

2.5.1 Formulasi *Forward (Forward Propagation)*

Formulasi *forward* digunakan untuk menghitung keluaran dari setiap *neuron* pada jaringan saraf tiruan. Gambar 2.18 menunjukkan struktur formulasi *forward* pada jaringan saraf tiruan yang terdiri dari masukkan, bobot, fungsi aktivasi dan keluaran.



Gambar 2.18 Struktur Formulasi F*orward*

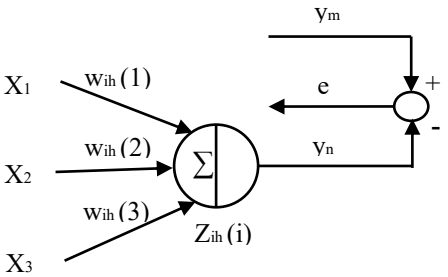
Untuk formulasinya menggunakan Persamaan 2.4 dan 2.5.

$$z_{ih}(i) = \sum_{i=1}^{n} w_{ih}(i,j) \times x(i) \quad (2.4)$$

$$y_{ih}(i) = \lambda \times z_{ih} \quad (2.5)$$

2.5.2 Formulasi *Backward (Backward Propagation)*

Formulasi *backward* digunakan untuk merevisi bobot dari nilai *error* yang diperoleh dari proses adaptasi jaringan terhadap keluaran model yang diinginkan. Setiap ada *error* baru, jaringan dapat belajar dari *error* tersebut dengan merevisi nilai bobot untuk menyesuaikan karakter nilai.



Gambar 2.19 Struktur Formulasi *Backward*

Gambar 2.19 menunjukkan struktur formulasi *backward* pada jaringan saraf tiruan. Persamaan 2.6 dan Persamaan 2.7 menunjukkan formulasi revisi bobot pada proses *backward.*

$$e = y_m - y_n \quad (2.6)$$

$$w^b = w^l + \alpha \times f'(z) \times x(i) \quad (2.7)$$

2.6 Metode *B-spline* [4]

B-spline adalah salah satu jenis jaringan syaraf tiruan (JST) yang dapat digolongkan dalam kelas AMN (*Associative Memory Network*) yang dapat menyimpan informasi secara lokal. Hal ini menyebabkan laju pembelajaran berlangsung relative lebih cepat dan secara efisien dapat digunakan sebagai komponen pengendali *plant* secara *online.* Seperti halnya jaringan syaraf tiruan jenis AMN lainnya, keluaran jaringan *B-spline* merupakan kombinasi bobot-bobot adaptif dari jumlah fungsi basis yang diaktifkan oleh masukan tertentu.

Secara historis *B-spline* telah digunakan secara umum sebagai algoritma pencocokan fungsi (*surface fitting*) di dalam bidang visualisasi grafis selama 20 tahun belakangan ini. Penggunaan pertama *B-spline* pertama kali digunakan pada tahun 1972 oleh Cox dan DeBoor. Ketika *B-spline* digunakan sebagai model *set* data terdapat kemungkinan memodifikasi data secara lokal, dan juga perubahan yang terjadi secara bersamaan pada respon jaringan lokal. Hal ini menjadi salah satu kemampuan yang menjadikan *B-spline* menarik untuk model adaptif dan kontrol.

B-spline telah digunakan pada beberapa bidang yang berbeda dalam riset robotik. Penerapannya adalah pada pembangkitan lintasan (jalan) di mana kehalusan lintasan yang dihasilkan. Penggunakan lainnya adalah algoritma pemarkiran mobil secara otomatis, kompensasi non-linier robotik, dan pemodelan aktuator robotik. Kemampuan utama algoritma *B-spline* adalah nilai keluarannya yang halus yang disebabkan oleh bentuk fungsi basis. Keluaran dari fungsi basis secara otomatis ditentukan oleh algoritmanya.

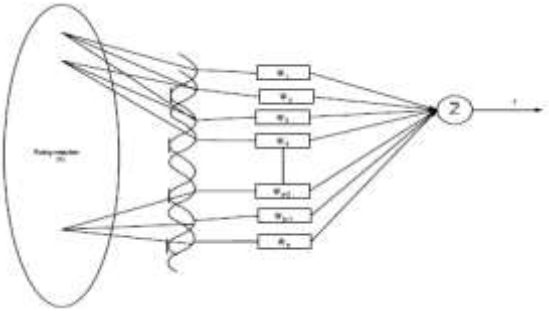
Beberapa elemen penting pada pembentukan kurva *B-spline*s adalah:

1. Derajat (p) - mengatur seberapa dekat kurva tersebut melewati titik kontrol dari kurva *B-spline*s. Semakin kecil derajat dari kurva *B-spline*s tersebut, maka semakin dekat pula kurva tersebut akan melewati titik-titik kontrol pembentuknya, dan sebaliknya apabila derajat kurva tersebut semakin besar, maka jarak titik kontrol kurva dengan kurva akan semakin jauh.
2. *Blending function* atau *basis function* (N) – merupakan fungsi yang menentukan seberapa besar lengkungan dari kurva *B-spline*s, yang dipengaruhi oleh besarnya derajat, knot vektor dan t.

Hal menarik dari AMN jenis ini adalah adanya hubungan langsung antara JST dengan sistem *fuzzy*. Dari sudut pandang *Fuzzy logic*, fungsi-fungsi basis *B-spline univariate* merepresentasikan statemen-statemen linguistik *fuzzy*, seperti „*error* positif kecil", „*error* besar", dan sebagainya. Hal ini menyebabkan JST *B-spline* dapat diinterpretasikan sebagai himpunan aturan *fuzzy*.

2.6.1 Struktur *B-spline* Standar

Jumlah fungsi basis yang memberi kontribusi pada keluaran *B-spline* adalah konstan yaitu sebanyak ρ. Dalam hal ini ada kaitan langsung antara jumlah basis fungsi yang diaktifkan oleh masukan tertentu dengan orde basis *B-spline* yang dipilih. Untuk masukan X dengan dimensi n dan keluaran skalar y seperti diperlihatkan oleh Gambar 2.20.



Gambar 2.20. Diagram Blok Jaringan Saraf Tiruan *B-spline* [4]

Maka keluaran *B-spline* adalah:

$$y(k) = \sum_{i=1}^{p} N_i(x)\, w_{ij}(x) \tag{2.8}$$

Di mana:

$w_{ij}(x)$ = nilai bobot ke j dengan indeks bobot ke i
$N_i(x)$= keluaran fungsi basis ke i
$y(k)$= keluaran jaringan syaraf *B-spline*

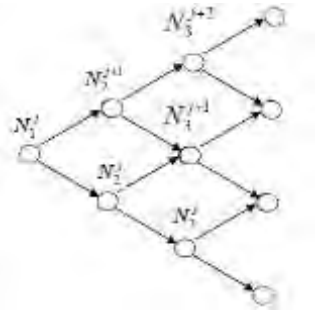
2.6.2 *Univariate Basis Function*

Keluaran basis fungsi yang diaktifkan oleh masukan tertentu (X) dapat dihitung dengan menggunakan hubungan *recurrence* dibawah ini:

$$N_k^j(x) = \left(\frac{x-\lambda_{j-k}}{\lambda_{j-1}-\lambda_{j-k}}\right) N_{k-1}^{j-1}(x) + \left(\frac{\lambda_j - x}{\lambda_j - \lambda_{j-k+1}}\right) N_{k-1}^{j}(x) \qquad (2.9)$$

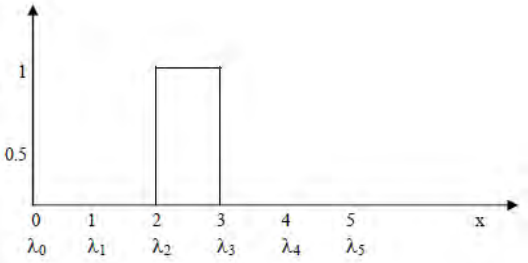
$$N_k^j(x) = \begin{cases} 1, & jika\ x \in I_j\ (\lambda_{j-1}, \lambda_j) \\ 0, & lainnya \end{cases}$$

Dengan λ_j adalah *knot* (posisi) ke j dan $I_j = (\lambda_{j-1}, \lambda_j)$ adalah interval ke j sedangkan k adalah orde dari basis fungsi tersebut, hubungan *recurrence* tersebut diilustrasikan oleh Gambar 2.21 berikut:



Gambar 2.21. Hubungan *Recurrence* [4]

2.6.2.1 Fungsi Basis Orde 1 (Konstan Sebagian-sebagian)



Gambar 2.22. Fungsi Basis Orde 1

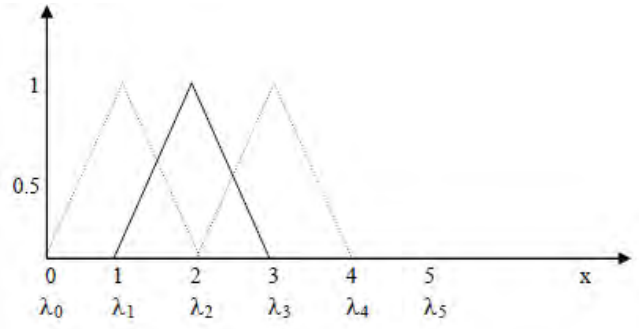
Misal N_I^j adalah fungsi basis k j dan I_j adalah interval ke- j $(\lambda_{j-1}, \lambda_j)$ seperti terlihat pada Gambar 2.22. Maka keluaran fungsi basisnya untuk masukan x adalah:

$$N_I^j(x) = \begin{cases} 1, & jika\ x \in I_j\ (\lambda_{j-1}, \lambda_j) \\ 0, & lainnya \end{cases} \tag{2.10}$$

Secara matematis Persamaan 2.10 diatas dapat diimplementasikan oleh fungsi berikut:

$$j = \lceil x \rceil$$
$$N(j) = 1$$

2.6.2.2 Fungsi Basis Orde 2 (Linier Sebagian-sebagian)



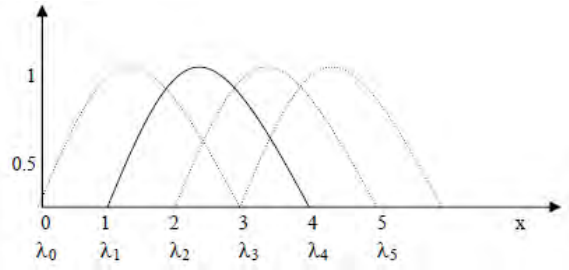
Gambar 2.23. Fungsi Basis Orde 2

Keluaran basis untuk masukan x dihitung :

$$N_2^j(x) = \left(\frac{x-\lambda_{j-2}}{\lambda_{j-1}-\lambda_{j-2}}\right) N_1^{j-1}(x) + \left(\frac{\lambda_j - x}{\lambda_j - \lambda_{j-1}}\right) N_1^j(x) \tag{2.11}$$

$$N_1^j(x) = \begin{cases} 1, & jika\ x \in I_j\ (\lambda_{j-1}, \lambda_j) \\ 0, & lainnya \end{cases}$$

2.6.2.3 Fungsi Basis Orde 3 (Kuadratik Sebagian-sebagian)



Gambar 2.24. Fungsi Basis Orde 3

Keluaran basis untuk masukan x dihitung :

$$N_3^j(x) = \left(\frac{x-\lambda_{j-3}}{\lambda_{j-1}-\lambda_{j-3}}\right) N_2^{j-2}(x) + \left(\frac{\lambda_j - x}{\lambda_j - \lambda_{j-2}}\right) N_2^j(x) \qquad (2.12)$$

Dalam hal ini $N_2^{j-2}(x)$ dan $N_2^j(x)$ dihitung dengan menggunakan fungsi basis orde 2 berikut:

$$N_2^j(x) = \left(\frac{x-\lambda_{j-2}}{\lambda_{j-1}-\lambda_{j-2}}\right) N_1^{j-1}(x) + \left(\frac{\lambda_j - x}{\lambda_j - \lambda_{j-1}}\right) N_1^j(x) \qquad (2.13)$$

2.6.3 Fungsi Keanggotaan *Fuzzy B-spline* Orde 2 [5]

B-spline adalah permukaan parametik yang memberikan fleksibilitas untuk visualisasi dan pemodelan dari berbagai jenis data. Semua fungsi keanggotaan *(membership function) fuzzy B-spline* dari aturan logika *fuzzy* dapat dihitung menggunakan algoritma dari orde 2. Fungsi keanggotaan *Fuzzy B-spline* dinyatakan sebagai :

$$\mu_{ij}(x_i) = \sum_{q=0}^{k-1} c_q N_{q,2}(x_i) \qquad (2.14)$$

Di mana :

$\mu_{ij}(x_i)$ = fungsi keanggotaan *variable fuzzy* x_i untuk himpunan *fuzzy* c_q ; $q = 0, 1, 2, \ldots, k-1$

k = jumlah titik kontrol

$N_{q,2}(x_i)$ = fungsi *B-spline* orde 2

Bentuk umum dari fungsi keanggotaan *B-spline* dinyatakan sebagai :

$$N_{q,k}(x_i) = \frac{x_i - t_q}{t_{q+k-1} - t_q} N_{q,k-1}(x_i) + \frac{t_{q+k} - x_i}{t_{q+k} - t_{q+1}} N_{q+1,k-1}(x_i) \qquad (2.15)$$

Fungsi basis *B-spline* untuk orde $(k = 2)$. $N_{q,2}$ dapat ditulis sebagai:

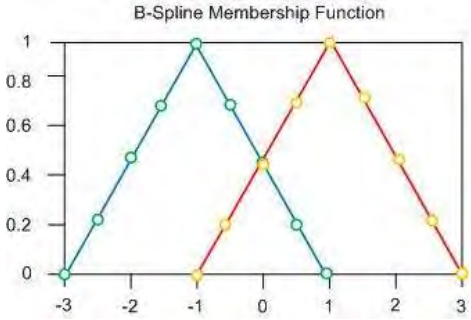
$$N_{q,2}(x_i) = \frac{x_i - t_q}{t_{q+1} - t_q} N_{q,1}(x_i) + \frac{t_{q+2} - x_i}{t_{q+2} - t_{q+1}} N_{q+1,1}(x_i) \qquad (2.16)$$

$$N_{q,1}(x_i) = \begin{cases} 1, & x \in [t_q, t_{q+1}] \\ 0, & lainnya \end{cases}$$

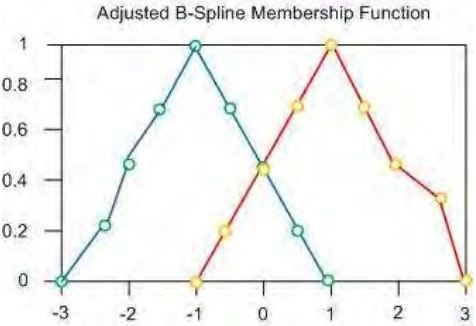
$$N_{q+1,1}(x_i) = \begin{cases} 1, & x \in [t_{q+1}, t_{q+2}] \\ 0, & lainnya \end{cases}$$

Di mana $t_q, t_{q+1} \, dan \, t_{q+2}$ adalah *node* yang didefinisikan dalam interval dari x_i. Dengan mensubstitusikan Persamaan 2.15 dan Persamaan 2.16 ke dalam Persamaan 2.14, menjadi :

$$t_{ij}(x_i) = \begin{cases} c_o \dfrac{x_i - t_0}{t_1 - t_0} & x_i \in [t_0, t_1] \\ c_o \dfrac{t_2 - x_i}{t_2 - t_1} + c_1 \dfrac{x_i - t_1}{t_2 - t_1} & x_i \in [t_1, t_2] \\ \vdots & \vdots \\ c_q \dfrac{t_{q+2} - x_i}{t_{q+2} - t_{q+1}} + c_{q+1} \dfrac{x_i - t_{q+1}}{t_{q+2} - t_{q+1}} & x_i \in [t_{q+1}, t_{q+2}] \\ \vdots & \vdots \\ c_{k-2} \dfrac{t_k - x_i}{t_k - t_{k-1}} + c_{z-1} \dfrac{x_i - t_{k-1}}{t_k - t_{k-1}} & x_i \in [t_{k-1}, t_k] \\ c_{k-1} \dfrac{t_{k+1} - x_i}{t_{k+1} - t_k} & x_i \in [t_k, t_{k+1}] \end{cases}$$



Gambar 2.25 *B-spline Membership Function* Orde 2



Gambar 2.26 *Adjusted B-spline Membership Function* Orde 2

2.6.4 Fungsi Keanggotaan *Fuzzy B-spline* Orde 3 [6]

Fungsi keanggotaan *Fuzzy B-spline* orde 3 dapat dinyatakan sebagai berikut

$$t_{ij}(x_i) = \sum_{q=0}^{s-1} c_q N_{q,3}(x_i) \tag{2.17}$$

Di mana :

$t_{ij}(x_i)$ = fungsi keanggotaan *variable fuzzy* x_i ke himpunan *fuzzy* j

c_q = titik kontrol, $q = 0, 1, 2, \dots, s-1$

s = jumlah titik kontrol

$N_{q,3}(x_i)$ = fungsi *B-spline* orde 3

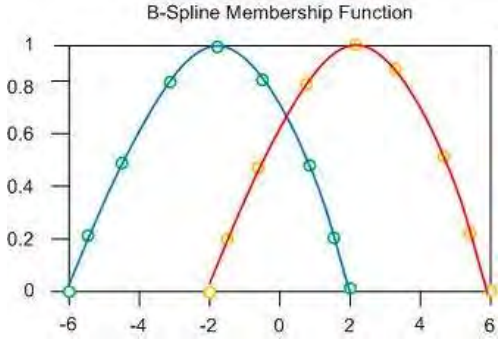
Fungsi basis *B-spline* $N_{q,3}$ dapat dinyatakan sebagai berikut :

$$N_{q,3}(x_i) = \frac{x_i - t_q}{t_{q+2} - t_q} N_{q,2}(x_i) + \frac{t_{q+3} - x_i}{t_{q+3} - t_{q+1}} N_{q+1,2}(x_i) \quad (2.18)$$

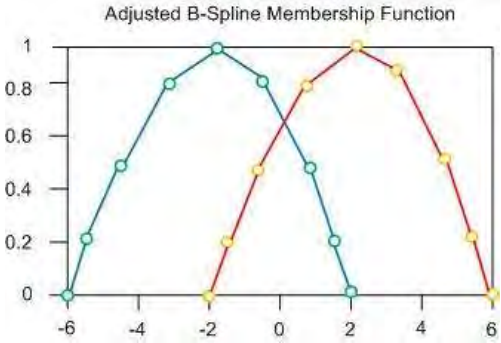
$$N_{q,2}(x_i) = \begin{cases} 1, & x \in [t_q, t_{q+2}] \\ 0, & lainnya \end{cases}$$

$$N_{q+1,2}(x_i) = \begin{cases} 1, & x \in [t_{q+1}, t_{q+3}] \\ 0, & lainnya \end{cases}$$

Di mana t_q, t_{q+1}, t_{q+2} *dan* t_{q+3} adalah *node* yang didefinisikan dalam interval x_i .



Gambar 2.27 *B-spline Membership Function* Order 3

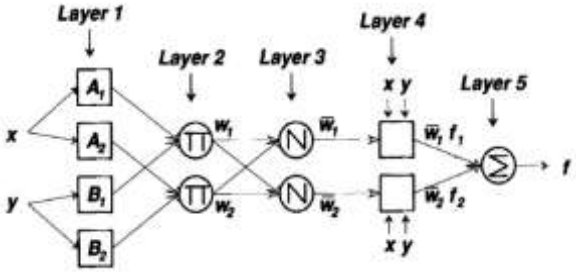


Gambar 2.28 *Adjusted B-spline Membership Function* Orde 3

2.7 Kontroler *Neuro-Fuzzy* [6]

Sistem hibrida yang menggabungkan logika *fuzzy*, jaringan saraf tiruan, algoritma genetika, dan kecerdasan buatan lainnya sudah banyak dikembangkan untuk menyelesaikan berbagai masalah yang berkaitan dengan optimalisasi. Konsep hibrida dikembangkan karena pada metode kecerdasan buatan ini memiliki kelebihan dan kekurangan masing-masing, sehingga dengan konsep hibrida akan diperolehkan suatu metode yang lebih baik dalam menyelesaikan permasalahan optimasi.

Gambar 2.29 menunjukkan skema atau model dari kontroler *Neuro fuzzy*. Terdapat beberapa skema atau struktur *Neuro fuzzy,* seperti *Neuro fuzzy* Mamdani atau Larsent, *Neuro fuzzy* TakagiSugeno (ANFIS / *Artificial Neuro fuzzy Inference System*), *Neuro fuzzy* Biner (Multi *input*) atau *Neuro fuzzy* formulasi Wang. Dari Gambar 2.29 dapat diketahui bahwa dalam struktur *Neuro fuzzy* terdapat 2 tahapan yaitu *forward propagation* dan *backward propagation.*



Gambar 2.29 Struktur Kontroler *Neuro-Fuzzy* dengan 2 *Input*

Kontroler NF merupakan sistem hibrida yang menggabungkan konsep logika *fuzzy* dengan jaringan saraf tiruan (JST). Logika *fuzzy* memiliki kelebihan dalam pengambilan keputusan, sedangkan JST memiliki kelebihan dalam adaptasi melalui kemampuan pembelajaran yang dimilikinya. Kekurangan logika *fuzzy* dalam hal penentuan parameter yang sifatnya intuitif dapat diatasi dengan JST. Pada kontroler NF, logika *fuzzy* akan direpresentasikan dalam JST yang memungkinkan terjadinya pembelajaran di dalamnya dan akan terjadi perubahan bobot untuk mengubah parameter pada logika *fuzzy*.

2.7.1 Tahap *Training*

Pendekatan pemodelan dalam fungsi *neuro fuzzy* serupa dengan teknik-teknik identifikasi sistem pada umumnya. Pertama *neuro fuzzy* mengasumsikan adanya sebuah struktur tertentu yang menghubungkan *input* dengan *output*. Kemudian *neuro fuzzy* harus diberikan pasangan data *input* dan *output* dalam format yang kompatibel untuk *training*. Jika kedua tahap terpenuhi maka *neuro fuzzy* siap digunakan untuk melatih FIS sehingga mampu menirukan kelakuan sistem yang sedang dimodelkan. *Neuro fuzzy* melatih FIS dengan memodifikasi parameter-parameter fungsi keanggotaan sampai diperoleh selisih *(error)* minimal antara keluaran FIS dengan data pelatihan *output*.

Pola *training* yang dilakukan untuk *mobile robot* ada berbagai macam cara seperti menjalankan robot menggunakan *remote control* kemudian data kecepatan dan pembacaan sensor disimpan untuk dijadikan data *training.* Selain itu bisa dilakukan dengan cara membuat sebuah persamaan kurva untuk *mobile robot* sehingga robot dapat membuat jalur yang menyerupai kurva kemudian data kecepatan dan pembacaan sensornya disimpan untuk data *training*. Dapat juga dilakukan dengan membuat algoritma sederhana gerakan melingkar untuk menghindari *obtstacles* kemudian data kecepatan dan pembacaan sensornya disimpan untuk data *training,* dan masih banyak cara lain untuk melakukan *training* terhadap *mobile robot* tersebut.

2.7.2 Tahap *Forward Propagation*

Tahapan *forward propagation* merupakan tahapan perhitungan sistem *fuzzy* yang direpresentasikan ke dalam beberapa *layer* seperti dalam *neural network.* Tahapan ini bertujuan untuk menentukan *output layer* dari suatu *node*. Dalam sistem ini terdapat 3 lapis data yang dalam beberapa lapisan diantaranya *input layer*, *hidden layer*, dan *output layer*.

2.7.2.1 Input Layer:

Input layer merupakan *node* masukkan yang mengirim sinyal masukkan ke *hidden layer*. Pada lapisan ini nilai keluaran *node* merupakan hasil dari fuzzifikasi *input variable* sistem. Lapisan ini berfungsi sebagai *membership function* untuk mengekspresikan nilai linguistik dari variabel linguistik masukkan.

2.7.2.2 Hidden Layer :

Hidden layer digunakan untuk basis aturan (*rule-base*) *fuzzy,* setiap *node* pada lapisan mempunyai masukkan dari dua nilai pada lapisan kedua. Operasi pada lapisan ini berbeda-beda tergantung dari struktur *Neuro fuzzy* yang digunakan. Misalnya jika digunakan struktur *Neuro fuzzy* Mamdani maka digunakan operasi MIN sedangkan jika menggunakan *Neuro fuzzy* Larsent digunakan operasi *product.* Persamaan 2.19 menunjukkan persamaan basis aturan (*rule-base*) pada struktur *Neuro fuzzy* Larsent.

$$R_{1,3} = O_{n,2} \times O_{m,2} \; ; untuk \; i = 1,2, \dots, n \times m \tag{2.19}$$

Lapisan ini juga menjalankan proses inferensi sehingga keluaran pada *node* ini diperoleh dari operasi inferensi. Sama halnya pada proses sebelumnya, pada lapisan ini dilakukan operasi inferensi yang bergantung pada struktur *Neuro fuzzy* yang digunakan. Misalnya jika digunakan struktur *Neuro fuzzy* Mamdani maka digunakan operasi MAX sedangkan jika menggunakan *Neuro fuzzy* Larsent digunakan operasi jumlah. Persamaan 2.20 dan Persamaan 2.21 menunjukkan operasi jumlah pada struktur *Neuro fuzzy* Larsent.

$$r_{i,4} = \sum_{j=1}^{n \times m} w_{ij} R_{(j,3)} \; ; \; untuk \; i = 1,2, \dots, n \times m \tag{2.20}$$

$$O_{i,4} = \sum_{i=1}^{n} r_{i,4} \; ; untuk \; i = 1,2, \dots, n \tag{2.21}$$

Pada struktur *Neuro fuzzy* Mamdani atau Larsent bobot w_{ij} bernilai satu karena pada struktur ini tidak terdapat revisi nilai bobot pada proses inferensi. Revisi bobot w_{ij} pada lapisan ini dilakukan apabila menggunakan struktur *Neuro fuzzy* Takagi-Sugeno, yang mana revisi bobot digunakan untuk merevisi *rule-base fuzzy.*

2.7.2.3 Output Layer

Output layer merupakan lapisan yang menjalankan proses defuzzikasi untuk menghitung sinyal keluaran dari kontroler *Neuro fuzzy.* Apabila digunakan fungsi *fuzzy singleton* maka Persamaan 2.22 menunjukkan persamaan dari fungsi *center of gravity.* Dengan bobot adalah w_i yang merupakan nilai tengah dari fungsi keanggotaan sinyal keluaran.

$$y = \frac{\sum_{i=1}^{n} w \times Uy_i}{\sum_{i=1}^{n} Uy_i} \quad (2.22)$$

2.7.3 Tahap *Backward Propagation*

Pada kontroler *Neuro-fuzzy,* proses *backward* dilakukan untuk merevisi nilai tengah dari *membership function.* Proses tersebut terdapat pada struktur *Neuro-fuzzy* Mamdani atau Larsent. Proses *backward* hanya dilakukan satu kali yaitu untuk merevisi nilai tengah dari fungsi keanggotaan sinyal kontrol saja. Berbeda dengan struktur *Neuro-fuzzy* Takagi-sugeno (ANFIS), proses *backward* dilakukan sampai ke proses inferensi. Jadi proses *backward* tidak hanya merevisi bobot sinyal kontrol tetapi juga merevisi *rule base* pada proses inferensi. Persamaan 2.23 menunjukkan persamaan revisi nilai tengah pada struktur *Neuro-fuzzy* Mamdani.

$$w_y^B = w_y^l(i) + \alpha \times e_{num} + \lambda + Uy(i) \quad (2.23)$$

BAB III
PERANCANGAN SISTEM

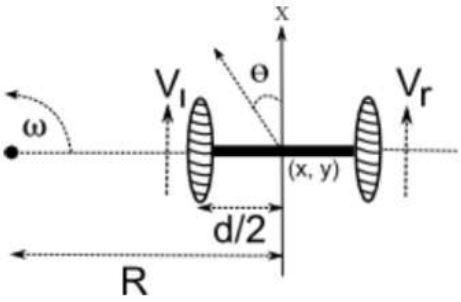
Pada bab tiga ini akan dibahas mengenai proses perancangan implementasi perencanaan jalur *mobile robot* Qbot menggunakan metode *B-spline* dengan kontroler *Neuro-Fuzzy.*

3.1 Kinematika *Mobile Robot* Qbot [7]

Pada bagian ini akan dibahas tentang kinematika dari *mobile robot* yaitu *forward kinematics* dan *invers kinematics.*

3.1.1 *Differential Drive Kinematics*

Sebuah *differential drive mobile robot* terdiri dari dua roda yang terpasang pada aksis yang sama dan tiap roda bisa dikontrol secara bebas untuk menggerakkan robot kearah depan atau ke belakang. Ketika kecepatan tiap roda dapat bervariasi untuk meraih gerakan melingkar, robot harus memutar sekitar titik yang disebut sebagai ICC (*Instantaneous Center of Curvature*), seperti yang ditunjukkan pada Gambar 3.1.



Gambar 3.1 Kinematik *Differential Robot*

Lintasan robot dapat dikendalikan dengan cara mengatur kecepatan pada tiap rodanya. Kecepatan rotasi ω sekitar ICC harus sama pada kedua roda. Maka, persamaan berikut ini menyusun hubungan antara parameter gerak dari *differential drive mobile robot.*

$$\omega\left(R+\frac{d}{2}\right)=V_r \tag{3.1}$$

$$\omega\left(R-\frac{d}{2}\right)=V_l \tag{3.2}$$

Di mana d adalah jarak antara titik pusat dari kedua roda, V_r dan V_l adalah kecepatan roda kanan dan kiri saat menyusuri lantai,dan R adalah jarak dari ICC ke titik tengah antar roda.

Persamaan 3.1 dan 3.2 dapat diselesaikan pada setiap kasus waktu untuk R dan ω sebagai berikut

$$R=\frac{d(V_r+V_l)}{2(V_r-V_l)} \tag{3.3}$$

$$\omega=\frac{V_r-V_l}{d} \tag{3.4}$$

3.1.2 Lokasi ICC

Pada Gambar 3.1 diasumsikan bahwa robot berada di sebuah posisi (x,y) dan menuju kearah yang membuat sudut θ dengan X aksis. Dengan mengetahui kecepatan Vr, V_l dan menggunakan Persamaan 3.3 dan 3.4, lokasi ICC (ICC_x , ICC_y) dapat ditentukan sebagai berikut:

$$\begin{aligned} ICC_x &= x - Rsin\theta \\ ICC_y &= y + Rcos\theta \end{aligned} \tag{3.5}$$

3.1.3 *Forward Kinematics*

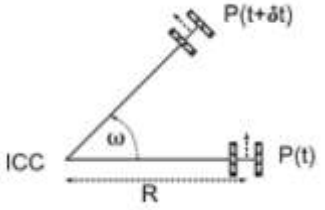
Berikut ini penjelasan permasalahan *forward kinematics* tentang bagaimana kecepatan yang diberikan pada roda dan konfigurasi posisi awal robot $(x, y, \theta)_{t=0}$ dapat menentukan posisi robot (x, y, θ).

3.1.3.1 Pose Relative Robot ke Lokasi ICC

Diberikan sebuah kecepatan yang tidak bervariasi pada Vr dan Vl, lokasi ICC (ICC_x, ICC_y) akan menjadi posisi tetap. Karena itu , pada waktu t+δt *pose* robot akan menjadi:

$$\begin{bmatrix} x' \\ y' \\ \theta' \end{bmatrix} = \begin{bmatrix} \cos(\omega \times \delta t) & -\sin(\omega \times \delta t) & 0 \\ \sin(\omega \times \delta t) & \cos(\omega \times \delta t) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x - ICC_x \\ y - ICC_y \\ \theta \end{bmatrix} + \begin{bmatrix} ICC_x \\ ICC_y \\ \omega \times \delta t \end{bmatrix} \tag{3.6}$$

Di mana (x, y, θ) dan (x', y', θ') adalah posisi robot pada waktu t dan t+δt, berturut-turut. Persamaan ini menggambarkan gerak berputar robot sekitar ICC nya dengan sebuah radius dari lengkungan R sebuah kecepatan sudut ω, seperti pada Gambar 3.2



Gambar 3.2 *Forward Kinematics Relative to ICC*

3.1.3.2 Pose Relative Robot pada Posisi Awal Robot

Persamaan gerak umum robot yang mampu bergerak dalam arah θ(t) tertentu pada kecepatan V(t) yang diberikan,dijelaskan sebagai berikut.

$$\begin{aligned} x(t) &= \int_0^t V(t)\cos[\theta(t)]\,dt \\ y(t) &= \int_0^t V(t)\sin[\theta(t)]\,dt \\ \theta(t) &= \int_0^t \omega(t)dt \end{aligned} \tag{3.7}$$

Untuk sebuah robot *differential drive* seperti Qbot, Persamaan 3.7 menjadi

$$\begin{aligned} x(t) &= \frac{1}{2}\int_0^t [V_r(t) + V_l(t)]\cos[\theta(t)]\,dt \\ y(t) &= \frac{1}{2}\int_0^t V(t)\sin[\theta(t)]\,dt \\ \theta(t) &= \frac{1}{d}\int_0^t \omega(t)dt \end{aligned} \tag{3.8}$$

Persamaan 3.7 dapat disederhanakan untuk menentukan *pose* robot saat V sebagai berikut

$$\begin{bmatrix} x' \\ y' \\ \theta' \end{bmatrix} = \begin{bmatrix} x + V\cos(\theta)\delta t \\ y + V\sin(\theta)\delta t \\ \theta + \omega\delta t \end{bmatrix} \tag{3.9}$$

Demikian pula pada Persamaan 3.8

$$\begin{bmatrix} x' \\ y' \\ \theta' \end{bmatrix} = \begin{bmatrix} x + \frac{1}{2}[V_r + V_l]\cos(\theta)\,\delta t \\ y + \frac{1}{2}[V_r + V_l]\sin(\theta)\,\delta t \\ \theta + \frac{1}{d}[V_r - V_l]\delta t \end{bmatrix} \tag{3.10}$$

Dengan demikian, untuk kasus khusus dari $V_l = V_r = V$ dan $V_l = -V_r = V$, Persamaan 3.9 menjadi Persamaan 3.11 dan 3.12 berikut

$$\begin{bmatrix} x' \\ y' \\ \theta' \end{bmatrix} = \begin{bmatrix} x + Vcos(\theta)\delta t \\ y + Vsin(\theta)\delta t \\ \theta \end{bmatrix} \tag{3.11}$$

$$\begin{bmatrix} x' \\ y' \\ \theta' \end{bmatrix} = \begin{bmatrix} x \\ y \\ \theta + \frac{2V\delta t}{d} \end{bmatrix} \tag{3.12}$$

3.1.4 *Inverse Kinematics*

Berikut ini penjelasan permasalahan *inverse kinematics* tentang robot dengan konfigurasi posisi awal $(x, y, \theta)_{t=0}$ dapat mencapai konfigurasi target (x, y, θ).

$$x(t) = \frac{d(V_r+V_l)}{2(V_r-V_l)} sin\left[\frac{t}{d}(V_r - V_l)\right]$$

$$y(t) = -\frac{d(V_r+V_l)}{2(V_r-V_l)} cos\left[\frac{t}{d}(V_r - V_l)\right] + \frac{d(V_r+V_l)}{2(V_r-V_l)} \tag{3.13}$$

$$\theta(t) = \frac{d(V_r+V_l)}{2(V_r-V_l)}$$

Dengan waktu target t dan posisi target (x, y), V_r dan V_l dapat diketahui dengan Persamaan 3.13 tetapi tidak untuk menemukan nilai θ.

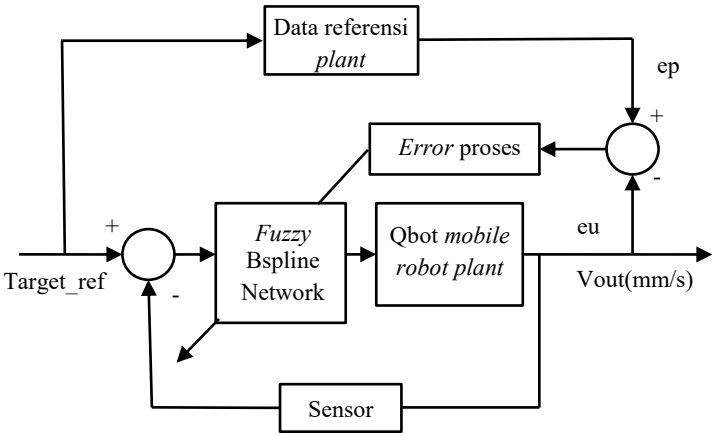
3.2 Perancangan Kontroler

Kontroler dirancang untuk mengatur kecepatan roda dan mencapai posisi target yang diinginkan. Selain itu, kontroler disini juga dirancang agar pergerakan robot menuju posisi target dapat lebih *smooth* ketika ada *obstacle* di sekitarnya.

3.2.1 Blok Diagram Sistem

Kontroler *neuro-fuzzy* yang digunakan adalah kontroler *neuro-fuzzy* mamdani karena hanya diperlukan untuk perbaikan nilai tengah dari himpunan pendukung sinyal kontrol dan *rule base* yang digunakan tetap.

Diagram blok pada Gambar 3.3 di bawah ini menunjukkan keseluruhan dari sistem perencanaan jalur pada *mobile robot* Qbot menggunakan metode *B-spline* dengan kontroler *Neuro-Fuzzy.*



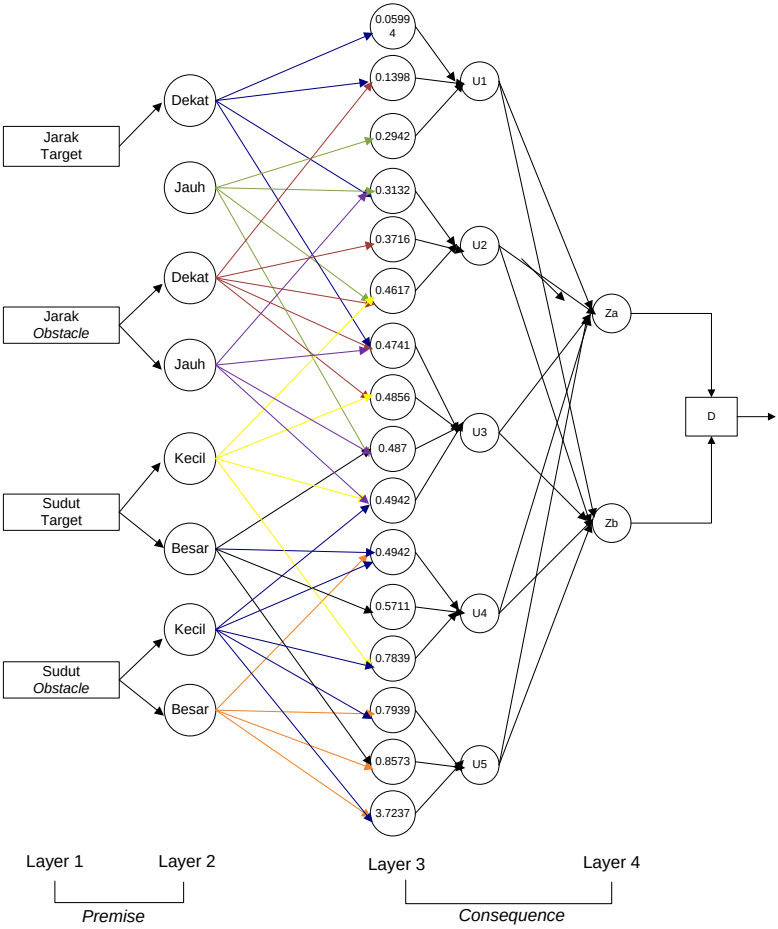
Gambar 3.3 Diagram Blok Sistem

Pada diagram blok di atas menunjukkan bahwa sistem ini terdiri dari beberapa subsistem atau bagian. Bagian target referensi merupakan bagian masukan berupa data jarak target, sudut target, jarak *obstacles*, dan sudut *obstacles* yang kemudian diolah ke dalam blok kontroler *neuro-fuzzy* dengan menggunakan metode Bspline untuk navigasi *mobile robot* Qbot.

Blok data referensi *plant* didapat dari data kecepatan robot yang sebelumnya dilakukan tahap *Forward propagation* dari data target referensi. Blok prediksi *error* merupakan selisih dari data referensi *plant* dengan data referensi *plant* yang diolah kembali pada kontroler *neuro-fuzzy* yang kemudian digunakan untuk proses *learning* dan revisi bobot

3.2.2 Struktur *Fuzzy B-spline Neural Network* [8]

Sebuah sistem logika *fuzzy* biasanya terakumulasi data dalam bentuk algoritma *fuzzy*, terdiri dari aturan *fuzzy* linguistik yang berkaitan dengan *input* dan *output* jaringan. Struktur untuk *fuzzy B-spline neural network* digambarkan dalam Gambar 3.4.



Gambar 3.4 Struktur *Fuzzy B-spline Neural Network*

Sistem ini memiliki empat lapisan, yaitu :

1. **Layer I :** Lapisan ini merupakan lapisan *input*, yaitu *input node* yang mewakili masukan variabel linguistik. Terdapat 4 *input* variabel yaitu Jarak Target, Jarak *Obstacle*, Sudut Target dan Sudut *Obstacle*. Setiap *input* mempunyai 2 *membership function*.
2. **Layer II :** Dalam lapisan ini proses fuzzifikasi dilakukan dan *neuron* mewakili *fuzzy set* digunakan dari aturan *fuzzy* linguistik. *Output* dari lapisan ini adalah nilai-nilai fungsi keanggotaan, yaitu ij. Keanggotaan variabel *input* i ke j himpunan *fuzzy* didefinisikan oleh fungsi *B-spline* sebagai:

$$\mu_{ij}(x) = \sum_{q=0}^{z-1} c_q N_{q,2}(x) \tag{3.14}$$

   Di mana μ_{ij} menunjukan fungsi keanggotaan *input* i ke j himpunan *fuzzy*. Dengan nilai $i = 1, 2, \dots, m$ dan $j = 1, 2$.
3. **Layer III :** Lapisan ini adalah inferensi lapisan *fuzzy*. Dalam lapisan ini setiap *node* mewakili aturan *fuzzy*.
4. **Layer IV :** Lapisan ini adalah lapisan *output*. Dalam lapisan ini, proses defuzzifikasi dibuat untuk menghitung *output* dari seluruh jaringan, yaitu menghitung *output* keseluruhan sistem. Oleh karena itu, *output fuzzy B-spline neural network* dapat dinyatakan sebagai:

$$u = \frac{\sum_{i=1}^{m} \mu_i w_{pi}}{\sum_{i=1}^{m} \mu_i} \tag{3.15}$$

   Di mana u adalah *output* untuk seluruh jaringan. Pelatihan jaringan dimulai setelah memperkirakan nilai *output fuzzy B-spline neural network* dan w_{pi} adalah bobot antara layer III dan IV dan $p = 1, 2, \dots, n$, n adalah nomor kelas.

3.2.3 Fungsi Keanggotaan *Fuzzy B-spline* [9]

Ada beberapa keuntungan dalam menggunakan jaringan *B-spline*, salah satunya yaitu dapat memberikan pendekatan yang baik untuk fungsi linier dan fungsi nonlinier. Pelatihan jaringan *B-spline* cocok digunakan untuk kontrol secara *real-time* karena memerlukan lebih sedikit perhitungan dan penyimpanan dari fungsi dasar lainnya,

Fungsi keanggotaan *fuzzy B-spline* yang digunakan pada tugas akhir ini didefinisikan sebagai:

$$\mu_{ij}(x_q) = \sum_{i=0}^{r-1} c_i N_{i,3}(x_q) \tag{3.16}$$

Di mana :

1. $\mu_{ij}(x_q)$ = fungsi keanggotan variable *fuzzy* x_q ke-j himpunan *fuzzy*
2. c_i = titik kontrol c_q, $q = 1, 2, \dots, r-1$
3. r = jumlah titik kontrol
4. $N_{i,3}$ = fungsi *B-spline* orde 3

Fungsi *B-spline* $N_{i,3}$ didefinisikan sebagai

$$N_{i,3}(x_q) = \left(\frac{x_q - \lambda_{i-2}}{\lambda_{i-1} - \lambda_{i-2}}\right) N_{i,1}(x_q) + \left(\frac{\lambda_i - x_q}{\lambda_i - \lambda_{i-1}}\right) N_{i,1}(x_q) \tag{3.17}$$

$$N_{i,1}(x_q) = \begin{cases} 1, & \lambda_{i-1} \leq t \leq \lambda_i \\ 0, & lainnya \end{cases}$$

$$N_{i-1,1}(x_q) = \begin{cases} 1, & \lambda_{i-2} \leq t \leq \lambda_{i-1} \\ 0, & lainnya \end{cases}$$

Di mana $\lambda_{i-2}, \lambda_{i-1}, \lambda_i$ adalah *node* yang didefinisikan dalam interval x_i.

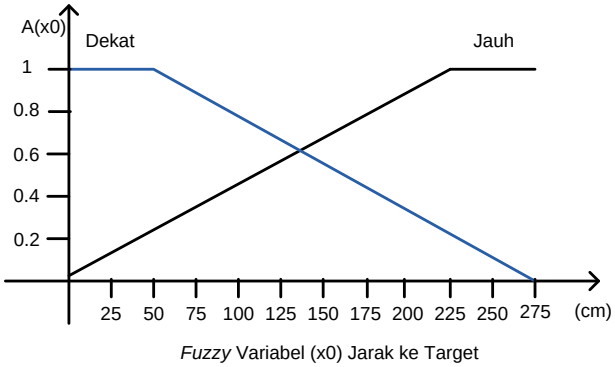
3.3 Perancangan Sistem Simulasi *B-spline*

Ada beberapa keuntungan dalam menggunakan jaringan *B-spline*, salah satunya yaitu dapat memberikan pendekatan yang baik untuk fungsi linier dan fungsi nonlinier. Pelatihan jaringan *B-spline* cocok digunakan untuk kontrol secara *real-time* karena memerlukan lebih sedikit perhitungan dan penyimpanan dari fungsi dasar lainnya.

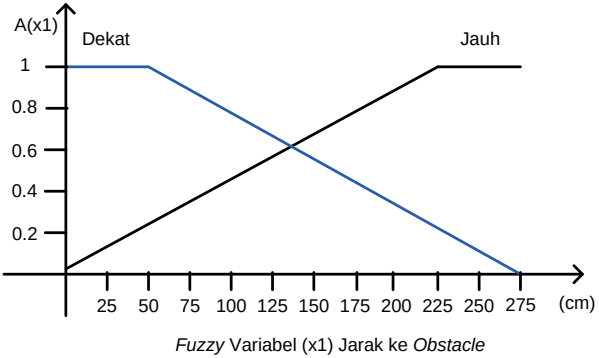
Metode *B-spline* pada *mobile robot* Qbot ini digunakan untuk menentukan jalur serta robot dapat bergerak lebih *smooth* saat menghindari *obstacle* yang terdapat disekitarnya. Dalam hal ini akan digunakan 2 pola lintasan yang nantinya akan dilalui oleh *mobile robot* Qbot. Perencanaan jalur atau lintasan *mobile robot* Qbot dibuat dengan 2 pola jalur atau lintasan.

3.3.1 Simulasi *B-spline* Jalur 1

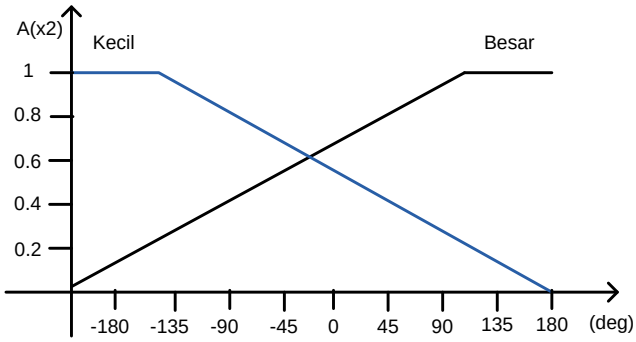
Pada simulasi jalur 1 ini, ditentukan *b-spline* dengan derajat kontrol 3 dan mempunyai 8 buah titik kontrol. Dengan menggunakan Persamaan 3.16, maka diperoleh *membership function fuzzy B-spline* seperti pada Gambar 3.5.



Gambar 3.5 *Membership Function* Jarak ke Target Jalur 1

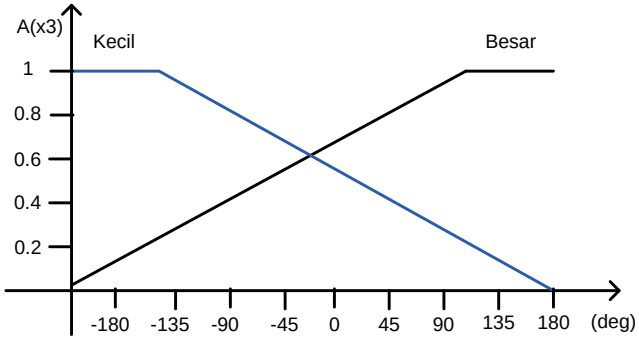


Gambar 3.6 *Membership Function* Jarak ke *Obstacle* Jalur 1



Fuzzy Variabel (x2) Sudut ke Target

Gambar 3.7 *Membership Function* Sudut ke Target Jalur 1



Fuzzy Variabel (x2) Sudut ke *Obstacle*

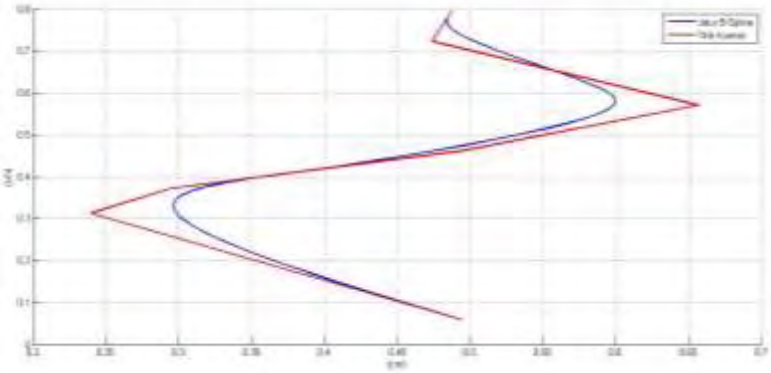
Gambar 3.8 *Membership Function* Sudut ke *Obstacle* Jalur 1

Gambar di atas menunjukkan tentang identifikasi fungsi keanggotaan dari *fuzzy b-spline* dengan *input variable* linguistik. *Premises* dari *control rules* dan identifikasi dari *consequences* dapat dilihat pada Tabel 3.1. Untuk x_0 dan x_1, nilai 0 dan 1 mewakili variabel *fuzzy* untuk "Dekat" dan "Jauh". Untuk x_2 dan x_3, nilai 0 dan 1 mewakili variabel *fuzzy* untuk "Kecil" dan "Besar". Sedangkan w_{ij} mewakili bobot dari *output* yang akan di*learning*.

Tabel 3.1 *Control Rules* Logika *Fuzzy* Jalur 1

Rule	***Premises***				***Consequences***
	x_0	x_1	x_2	x_3	W_{ij}
0	0	0	1	0	0,05994
1	0	0	1	1	0,2398
2	0	0	0	0	0,2942
3	0	0	0	1	0,3132
4	0	1	1	0	0,3716
5	0	1	1	1	0,4617
6	0	1	0	0	0,4751
7	0	1	0	1	0,4856
8	1	0	1	0	0,487
9	1	0	1	1	0,4942
10	1	0	0	0	0,4942
11	1	0	0	1	0,5711
12	1	1	1	0	0,6573
13	1	1	1	1	0,7237
14	1	1	0	0	0,7839
15	1	1	0	1	0,7939

Sehingga dari Tabel 3.1 dapat dibuat simulasi pada program Matlab yang mewakili metode *B-spline* untuk menentukan lintasan yang nantinya akan dilalui oleh *mobile robot.*

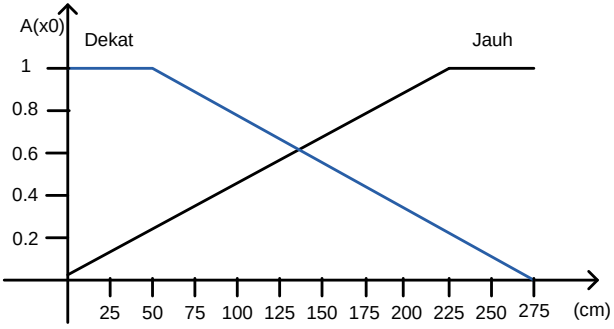


Gambar 3.9 Perencanaan Jalur 1 *B-spline*

3.3.2 Simulasi *B-spline* Jalur 2

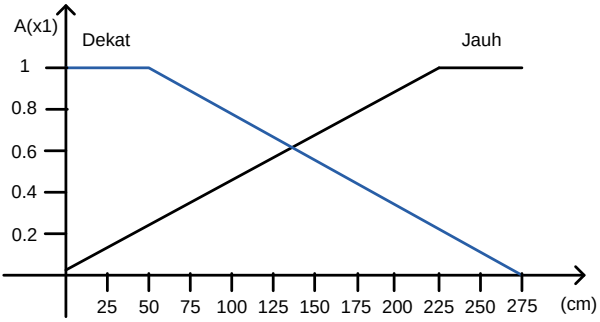
Sama seperti pada *B-spline* jalur 1, simulasi jalur 2 ini juga menggunakan derajat kontrol 3 dan titik kontrol berjumlah 8. Yang membedakan dengan jalur 1 yaitu penempatan dari titik-titik kontrolnya sehingga bentuk lintasannya pun akan berubah.

Gambar 3.10 sampai Gambar 3.11 menunjukkan fungsi keanggotan dari *fuzzy b-spline* yang tidak jauh berbeda pada jalur 1.



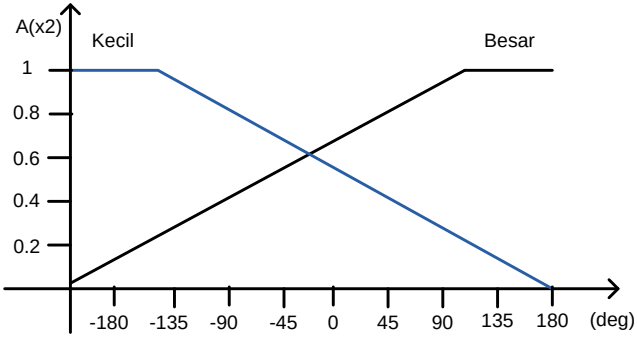
Fuzzy Variabel (x0) Jarak ke Target

Gambar 3.10 *Membership Function* Jarak ke Target Jalur 2

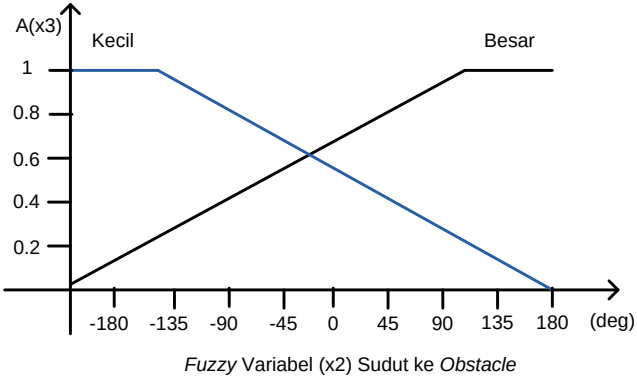


Fuzzy Variabel (x1) Jarak ke *Obstacle*

Gambar 3.11 *Membership Function* Jarak ke *Obstacle* Jalur 2



Gambar 3.12 *Membership Function* Sudut ke Target Jalur 2



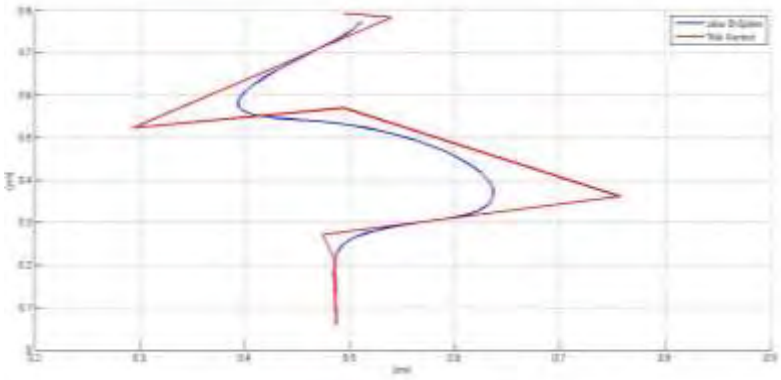
Gambar 3.13 *Membership Function* Sudut ke *Obstacle* Jalur 2

Gambar di atas menunjukkan tentang identifikasi fungsi keanggotaan dari *fuzzy b-spline* dengan *input variable* linguistik. *Premises* dari *control rules* dan identifikasi dari *consequences* dapat dilihat pada Tabel 3.2. Untuk x_0 dan x_1, nilai 0 dan 1 mewakili variabel *fuzzy* untuk "Dekat" dan "Jauh". Untuk x_2 dan x_3, nilai 0 dan 1 mewakili variabel *fuzzy* untuk "Kecil" dan "Besar". Sedangkan w_{ij} mewakili bobot dari *output* yang akan di*learning*.

Tabel 3.2 *Control Rules* Logika *Fuzzy* Jalur 2

Rule	*Premises*				*Consequences*
	x_0	x_1	x_2	x_3	W_{ij}
0	0	0	1	0	0,05994
1	0	0	1	1	0,2132
2	0	0	0	0	0,2716
3	0	0	0	1	0,2942
4	0	1	1	0	0,3617
5	0	1	1	1	0,4741
6	0	1	0	0	0,4856
7	0	1	0	1	0,487
8	1	0	1	0	0,4942
9	1	0	1	1	0,4942
10	1	0	0	0	0,5237
11	1	0	0	1	0,5398
12	1	1	1	0	0,5711
13	1	1	1	1	0,7573
14	1	1	0	0	0,7839
15	1	1	0	1	0,7939

Sehingga dari Tabel 3.2 dapat dibuat simulasi pada program Matlab yang mewakili metode *B-spline* untuk menentukan lintasan yang nantinya akan dilalui oleh *mobile robot.*



Gambar 3.14 Perencanaan Jalur 2 *B-spline*

BAB IV
HASIL DAN ANALISA

Bab ini akan membahas mengenai implementasi metode *B-spline* dengan sistem kontrol *neuro-fuzzy* yang telah dirancang pada bab sebelumnya. Untuk implementasi kontroler *neuro-fuzzy* terlebih dahulu dilakukan pengambilan data *traning* dengan menggunakan metode bspline sebagai penentu jalur atau lintasan untuk *output* yang diinginkan pada *plant*. Selanjutnya *output* tersebut dibandingkan dengan *output neuro-fuzzy* untuk mengetahui *error propagation* lalu dilakukan proses *offline learning* untuk mendapatkan bobot yang baru.

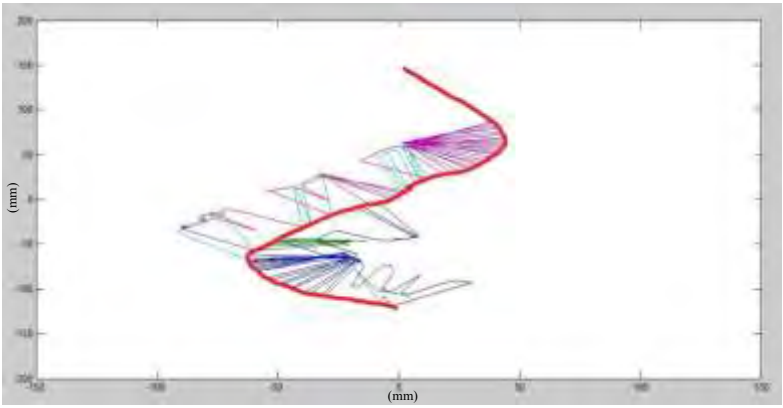
4.1 Implementasi Perencanaan Jalur 1

Perencanaan jalur 1 yang diimplementasikan memiliki beberapa tahapan. Tahapan pertama yaitu dilakukannya proses *training* dengan nilai bobot awal *random.* Setelah itu dilakukan proses *offline learning* untuk menghasilkan nilai bobot baru. Tahapan selanjutnya yaitu implementasi menggunakan metode *fuzzy b-spline neural network* dengan menggunakan nilai bobot yang baru.

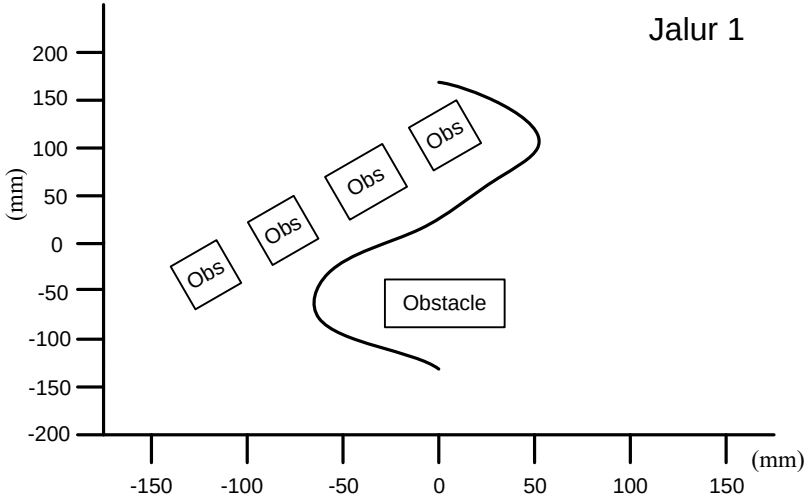
4.1.1 Tahap *Training*

Pada tahap *training* perencanaan jalur 1 ini dilakukan dengan memberi nilai bobot awal *random*. Jalur yang dilalui dibuat sesuai dengan perencanaan jalur menggunakan metode *B-spline* yang telah dibahas pada Bagian 3.3.1.

Gambar 4.1 Posisi Robot *Training* Jalur 1



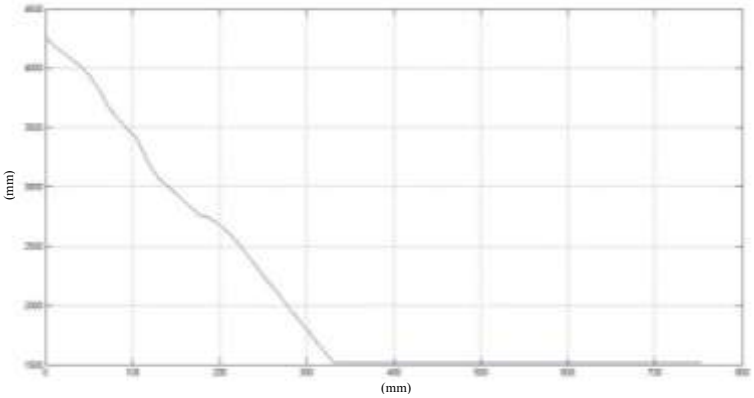
Gambar 4.2 Hasil *Training* Perencanaan Jalur 1



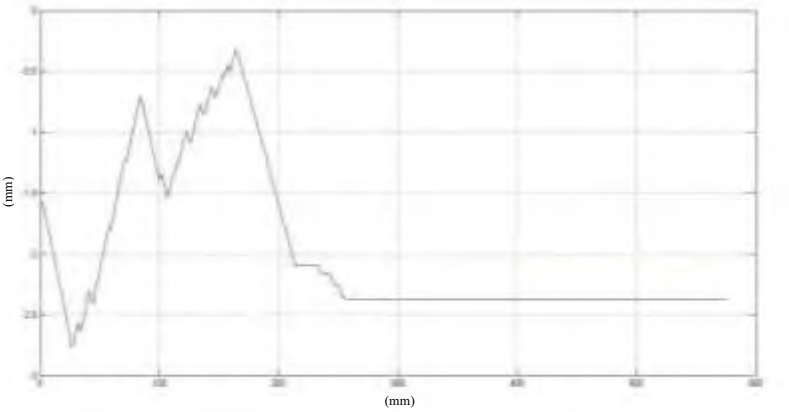
Gambar 4.3 *Training* Perencanaan Jalur 1

Berdasarkan Gambar 4.3 di atas, dapat dilihat bahwa untuk *training* jalur 1 pergerakan dari *mobile robot* masih belum seperti yang diharapkan. Untuk itu dari tahap *training* yang telah dilakukan, diambil data berupa Jarak ke Target, Jarak ke *Obstacle*, Sudut ke Target dan Sudut ke *Obstacle* yang dijadikan sebagai *input* untuk proses *learning*.

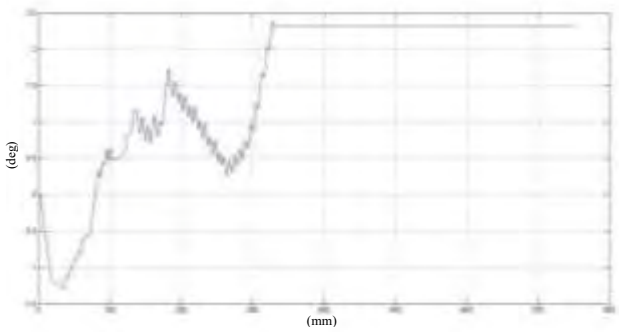
Hasil respon dari Jarak ke Target ditunjukan pada Gambar 4.4. Gambar selanjutnya menunjukkan respon dari Jarak ke *Obstacle* seperti pada Gambar 4.5. Sedangkan Gambar 4.6 menunjukkan respon dari Sudut ke Target dan Sudut ke *Obstacle* ditunjukkan pada Gambar 4.7. Pada tahap *training* ini juga didapatkan sinyal dari kontroler *fuzzy b-spline neural network* yang kemudian dibuat dalam bentuk hasil respon seperti pada Gambar 4.8.



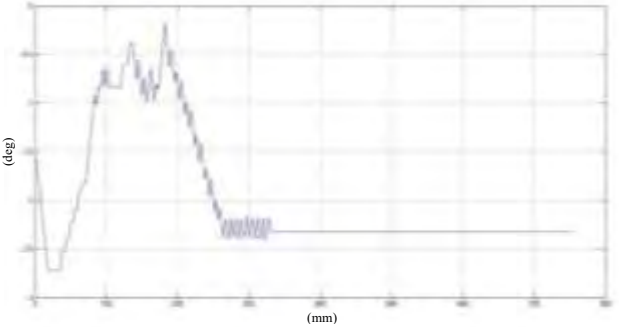
Gambar 4.4 Respon Jarak ke Target Tahap *Training* Jalur 1



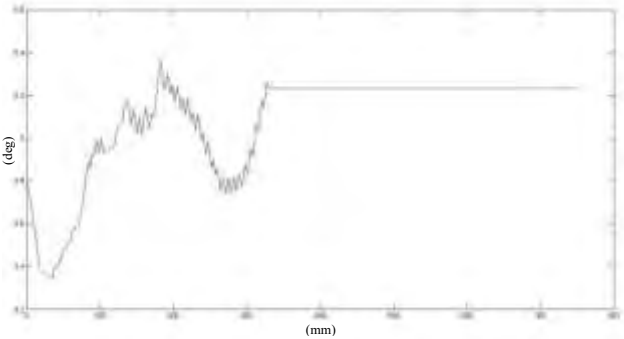
Gambar 4.5 Respon Jarak ke *Obstacle* Tahap *Training* Jalur 1



Gambar 4.6 Respon Sudut ke Target Tahap *Training* Jalur 1



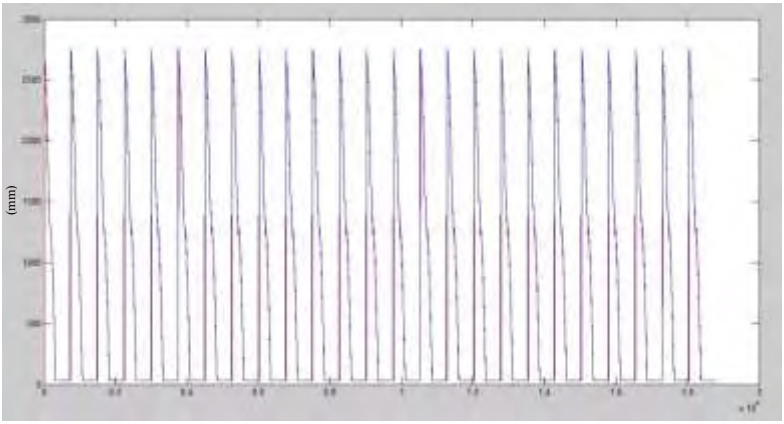
Gambar 4.7 Respon Sudut ke *Obstacle* Tahap *Training* Jalur 1



Gambar 4.8 Sinyal Kontrol *Fuzzy B-spline Neural Network* Tahap *Training* Jalur 1

4.1.2 Simulasi *Offline Learning*

Pada tahap ini, data yang telah didapatkan dari proses *training* di *learning* secara *offline* untuk mendapatkan nilai bobot baru. Data yang di *learning* yaitu Jarak ke Target, Jarak ke *Obstacle*, Sudut ke Target dan Sudut ke *Obstacle*. Gambar 4.9 menunjukkan hasil dari proses *learning* secara *offline* dengan menggunakan 25 iterasi.

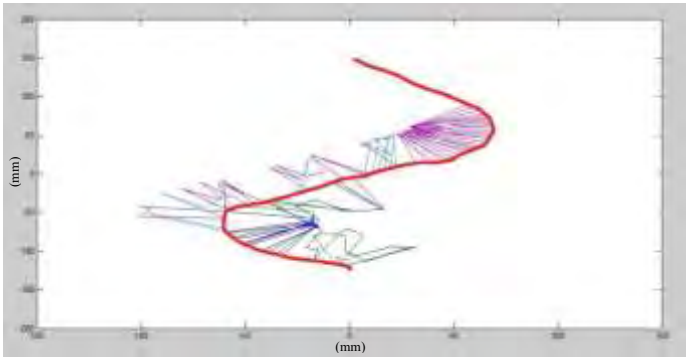


Gambar 4.9 Hasil Proses *Learning* Perencanaan Jalur 1

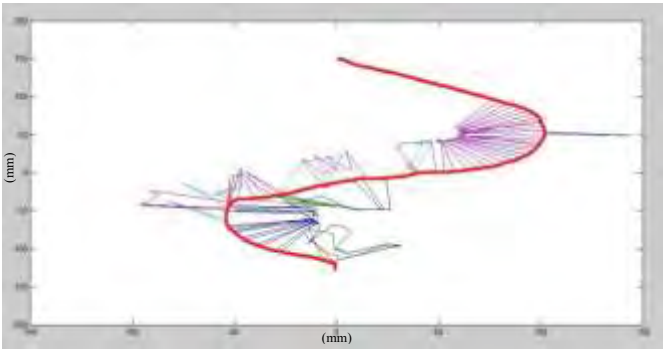
4.1.3 Implementasi *Fuzzy B-spline Neural Network*

Implementasi jalur 1 dengan *fuzzy b-spline neural network* ini dilakukan sebanyak 3 kali percobaan. Percobaan dilakukan sebanyak 3 kali untuk membandingkan hasil dari kontrol ini dapat bekerja dengan baik atau tidak. Dari setiap percobaan data yang diambil untuk dibandingkan yaitu kecepatan roda kanan, kecepatan roda kiri serta kontrol *fuzzy B-spline neural network*.

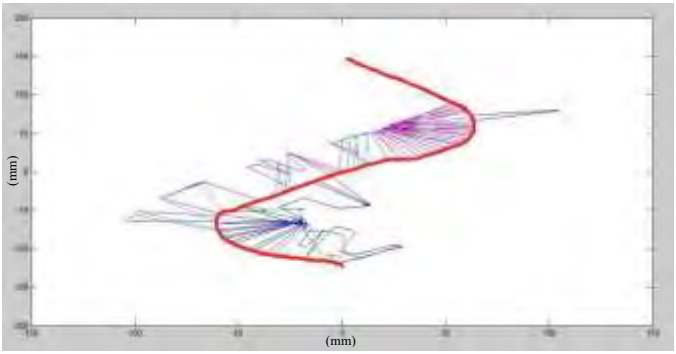
Percobaan pertama ditunjukkan pada Gambar 4.10. Terlihat bahwa bentuk dari lintasan yang dilalui *mobile robot* Qbot ini sudah sesuai mengikuti perencanaan jalur 1 dengan menggunakan metode *b-spline*. Begitu pula yang ditunjukkan pada Gambar 4.11 serta Gambar 4.12 yang merupakan percobaan kedua dan percobaan ketiga yang telah dilakukan.



Gambar 4.10 Percobaan 1 Perencanaan Jalur 1

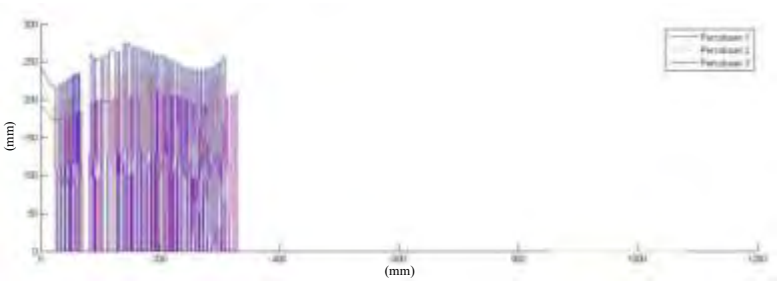


Gambar 4.11 Percobaan 2 Perencanaan Jalur 1

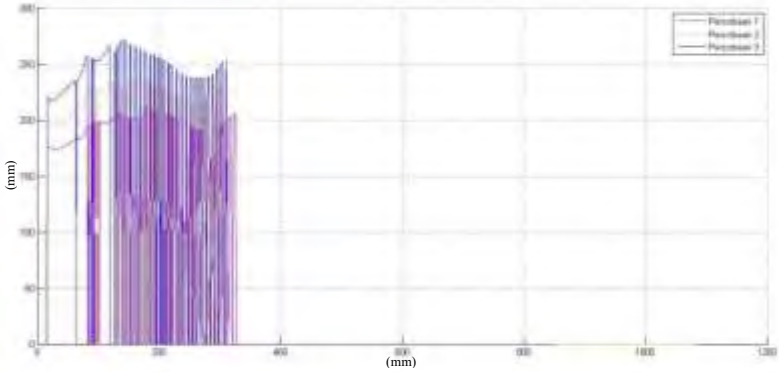


Gambar 4.12 Percobaan 3 Perencanaan Jalur 1

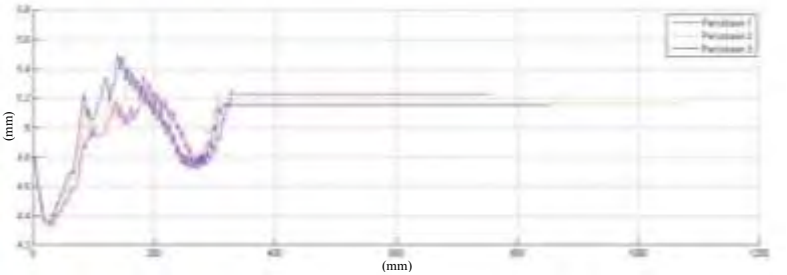
Dari ketiga percobaan tersebut kemudian dibandingkanlah kecepatan dari roda kanan dan kiri serta sinyal kontrol *fuzzy b-spline neural network* yang bekerja. Gambar 4.13 menunjukkan perbandingan dari kecepatan roda kanan, Gambar 4.14 menunjukkan perbandingan dari kecepatan roda kiri dan perbandingan dari sinyal kontrol yang bekerja terlihat pada Gambar 4.15



Gambar 4.13 Perbandingan Kecepatan Roda Kanan Jalur 1



Gambar 4.14 Perbandingan Kecepatan Roda Kiri Jalur 1



Gambar 4.15 Perbandingan Sinyal Kontrol *Fuzzy B-spline Neural Network* Jalur 1

Tabel 4.1 Perbandingan Waktu Tempuh Perencanaan Jalur 1

Training	Percobaan 1	Percobaan 2	Percobaan 3
34,02 detik	32,02 detik	32,31 detik	30,92 detik

Setelah melakukan 3 kali percobaan dengan menggunakan nilai bobot baru yang didapat dari proses *learning*, waktu tempuh *mobile robot* dengan jalur yang sama berbeda-beda. Perbedaan waktu ini dikarenakan faktor *eksternal* berupa pembacaan sensor dalam mendeteksi adanya *obstacle*. Akan tetapi, waktu tempuh yang dibutuhkan *mobile robot* setelah proses *learning* tetap lebih cepat dibandingkan dengan waktu tempuh pada saat *training.*

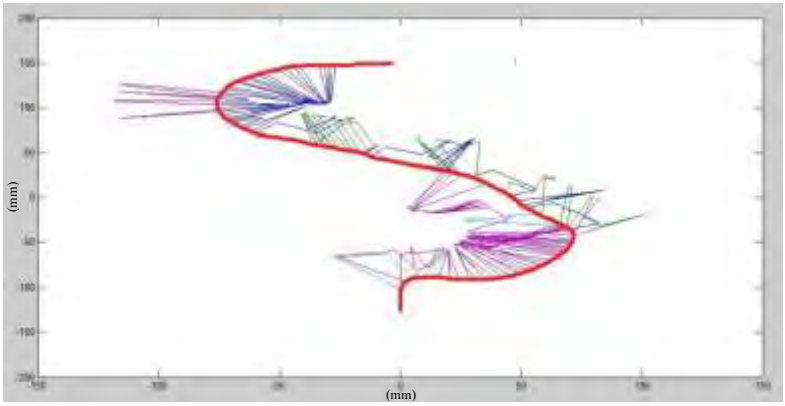
4.2 Implementasi Perencanaan Jalur 2

Perencanaan jalur 2 yang diimplementasikan juga memiliki tahapan yang sama seperti pada implementasi pada jalur 1. Tahapan pertama yaitu dilakukannya proses *training* dengan nilai bobot awal *random.* Setelah itu dilakukan proses *offline learning* untuk menghasilkan nilai bobot baru. Tahapan selanjutnya yaitu implementasi menggunakan metode *fuzzy b-spline neural network* dengan menggunakan nilai bobot yang baru.

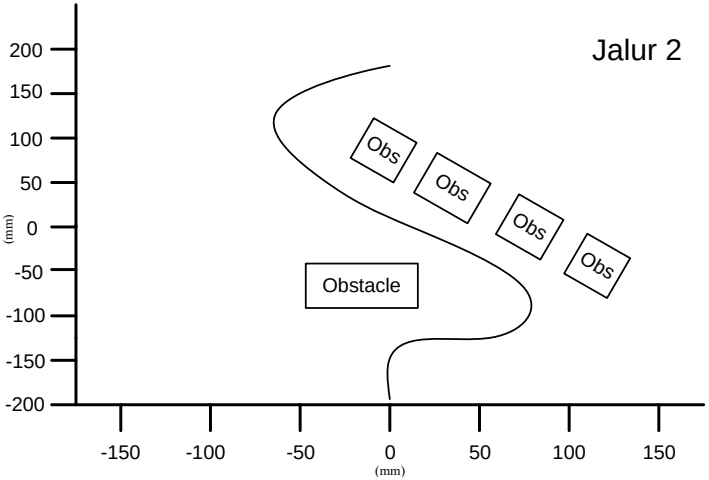
4.2.1 Tahap Training

Pada tahap *training* perencanaan jalur 2 ini juga dilakukan dengan memberi nilai bobot awal *random*. Jalur yang dilalui dibuat sesuai dengan perencanaan jalur menggunakan metode *B-spline* yang telah dibahas pada Bagian 3.3.1.

Gambar 4.16 Posisi Robot *Training* Jalur 2

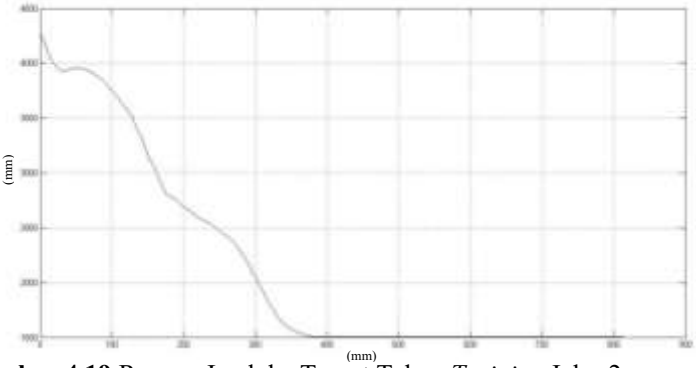


Gambar 4.17 Hasil *Training* Perencanaan Jalur 2

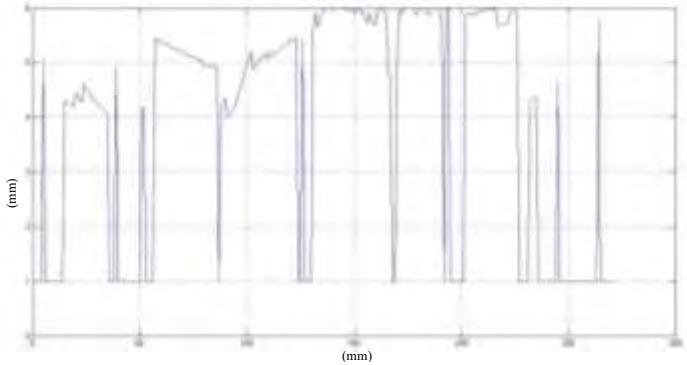


Gambar 4.18 *Training* Perencanaan Jalur 2

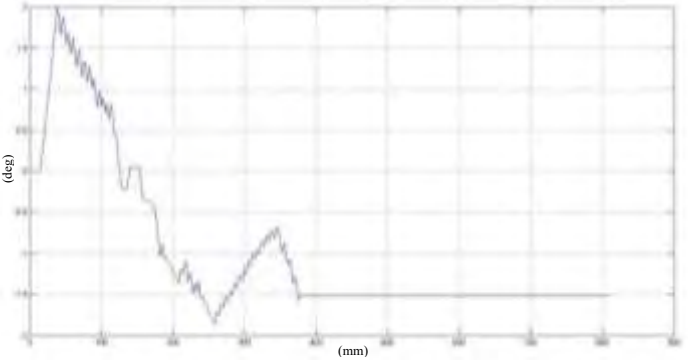
Dari tahap *training* yang telah dilakukan, didapatkan data berupa hasil respon dari Jarak ke Target seperti yang ditunjukan pada Gambar 4.19. Gambar selanjutnya menunjukkan respon dari Jarak ke *Obstacle* seperti pada Gambar 4.20. Sedangkan Gambar 4.21 menunjukkan respon dari Sudut ke Target dan Sudut ke *Obstacle* ditunjukkan pada Gambar 4.22. Pada tahap *training* ini juga didapatkan sinyal dari kontroler *fuzzy b-spline neural network* seperti pada Gambar 4.23.



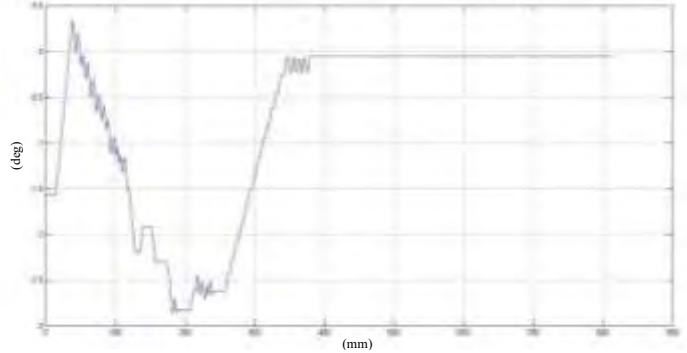
Gambar 4.19 Respon Jarak ke Target Tahap *Training* Jalur 2



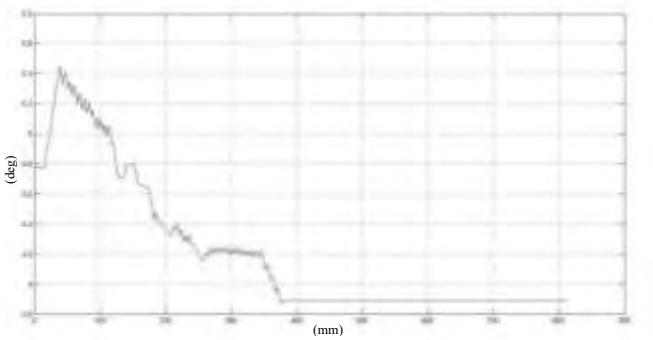
Gambar 4.20 Respon Jarak ke *Obstacle* Tahap *Training* Jalur 2



Gambar 4.21 Respon Sudut ke Target Tahap *Training* Jalur 2



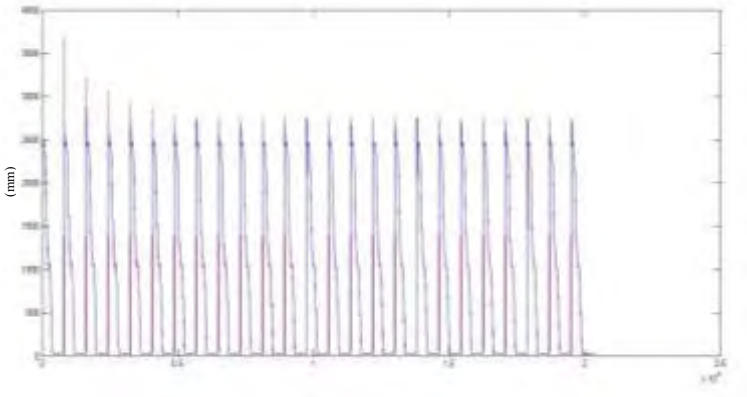
Gambar 4.22 Respon Sudut ke *Obstacle* Tahap *Training* Jalur 2



Gambar 4.23 Sinyal Kontrol *Fuzzy B-spline Neural Network* Tahap *Training* Jalur 2

4.2.2 Simulasi Offline Learning

Pada tahap ini, data yang telah didapatkan dari proses *training* di *learning* secara *offline* untuk mendapatkan nilai bobot baru. Data yang di *learning* yaitu Jarak ke Target, Jarak ke *Obstacle*, Sudut ke Target dan Sudut ke *Obstacle*. Gambar 4.24 menunjukkan hasil respon dari proses *learning* secara *offline* dengan menggunakan 25 iterasi.

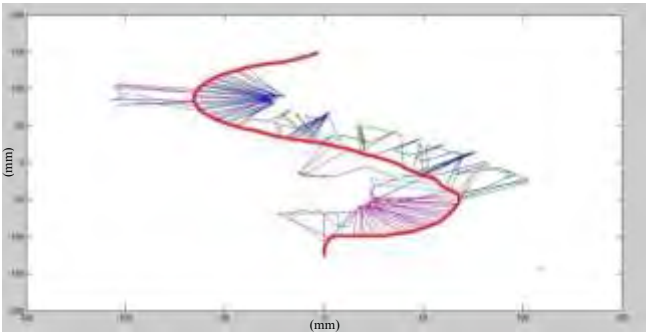


Gambar 4.24 Hasil Proses *Learning* Perencanaan Jalur 2

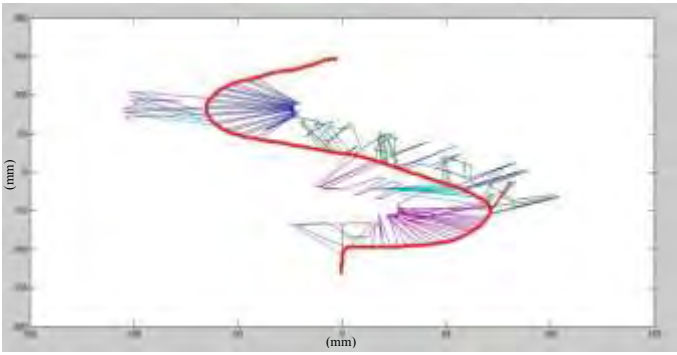
4.2.3 Implementasi *Fuzzy B-spline Neural Network*

Implementasi jalur 2 dengan *fuzzy b-spline neural network* ini dilakukan sebanyak 3 kali percobaan. Percobaan dilakukan sebanyak 3 kali untuk membandingkan hasil dari kontrol ini dapat bekerja dengan baik atau tidak. Dari setiap percobaan data yang diambil untuk dibandingkan yaitu kecepatan roda kanan, kecepatan roda kiri serta kontrol *fuzzy b-spline neural network*.

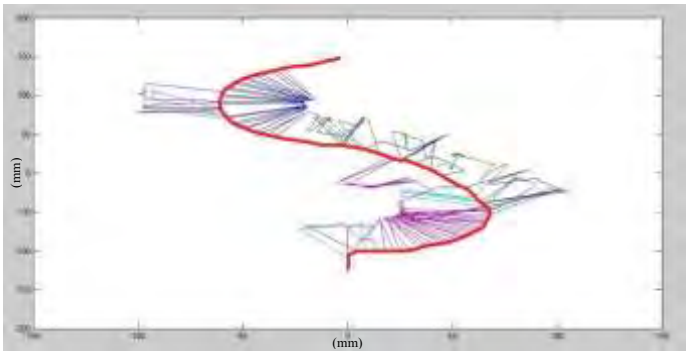
Percobaan pertama ditunjukkan pada Gambar 4.25. Terlihat bahwa bentuk dari lintasan yang dilalui *mobile robot* Qbot ini sudah sesuai mengikuti perencanaan jalur 2 dengan menggunakan metode *b-spline*. Begitu pula yang ditunjukkan pada Gambar 4.26 serta Gambar 4.27 yang merupakan percobaan kedua dan percobaan ketiga yang telah dilakukan.



Gambar 4.25 Percobaan 1 Perencanaan Jalur 2

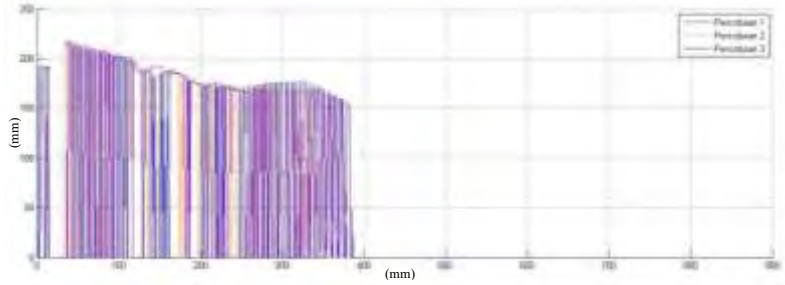


Gambar 4.26 Percobaan 2 Perencanaan Jalur 2

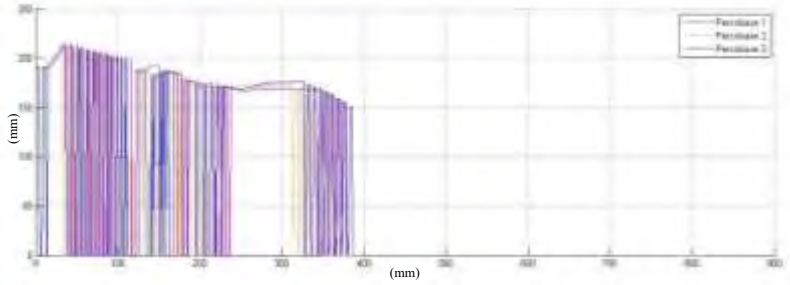


Gambar 4.27 Percobaan 3 Perencanaan Jalur 2

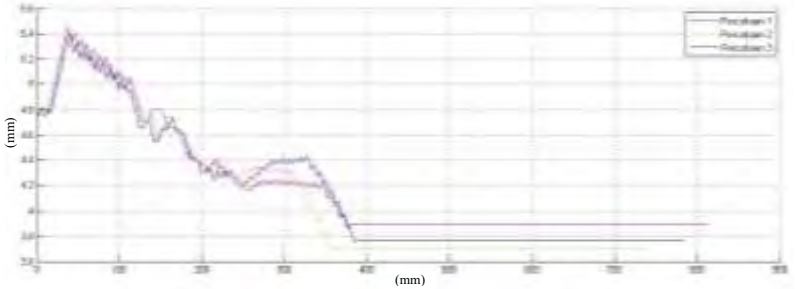
Dari ketiga percobaan tersebut kemudian dibandingkanlah kecepatan dari roda kanan dan kiri serta sinyal kontrol *fuzzy b-spline neural network* yang bekerja. Gambar 4.28 menunjukkan perbandingan dari kecepatan roda kanan, Gambar 4.29 menunjukkan perbandingan dari kecepatan roda kiri dan perbandingan dari sinyal kontrol yang bekerja terlihat pada Gambar 4.30.



Gambar 4.28 Perbandingan Kecepatan Roda Kanan Jalur 2



Gambar 4.29 Perbandingan Kecepatan Roda Kiri Jalur 2



Gambar 4.30 Perbandingan Sinyal Kontrol *Fuzzy B-spline Neural Network* Jalur 2

Setelah melakukan 3 kali percobaan dengan menggunakan nilai bobot baru yang didapat dari proses *learning*, waktu tempuh *mobile robot* dengan jalur yang sama berbeda-beda. Perbedaan waktu ini dikarenakan faktor *eksternal* berupa pembacaan sensor dalam mendeteksi adanya *obstacle*. Akan tetapi, waktu tempuh yang dibutuhkan *mobile robot* setelah proses *learning* tetap lebih cepat dibandingkan dengan waktu tempuh pada saat *training*.

Tabel 4.2 Perbandingan Waktu Tempuh Perencanaan Jalur 2

Training	Percobaan 1	Percobaan 2	Percobaan 3
39,32 detik	37,67 detik	36,74 detik	36,54 detik

Halaman ini sengaja dikosongkan

BAB V
PENUTUP

5.1 Kesimpulan

Dari hasil pengimplementasian perencanaan jalur pada Qbot *mobile robot* menggunakan metode *fuzzy b-spline neural network* dapat disimpulkan bahwa sistem *fuzzy b-spline neural network* ini dapat mengikuti data referensi dari hasil *training* yang telah dilakukan. Metode *fuzzy b-spline neural network* memiliki respon yang lebih cepat dibandingkan dengan respon saat *training.* Sebagaimana dari hasil pengujian kontroler *neuro fuzzy* pada Qbot *mobile robot* untuk *training* jalur 1 waktu yang dibutuhkan untuk mencapai target adalah 34,2 detik untuk bergerak dari titik (0,0)cm menuju titik (0,275)cm, sedangkan waktu yang dibutuhkan kontroler *neuro fuzzy* pada percobaan pertama adalah 32,02 detik, percobaan kedua 32,31 detik dan percobaan ketiga 30,92 detik. Untuk pola 2 saat *training* membutuhkan waktu 39,32 detik untuk bergerak dari titik (0,0)cm menuju titik (0,275)cm, sedangkan kontroler *neuro fuzzy* pada percobaan pertama adalah 37,67 detik, percobaan kedua 36,74 detik dan percobaan ketiga 36,54 detik. Untuk perbedaan respon sinyal kontrol *neuro fuzzy* dari tiga kali kali percobaan disebabkan faktor *eksternal* seperti kondisi kalibrasi sensor dan lingkungan yang dapat mempengaruhi sensor.

5.2 Saran

Untuk pengembangan penelitian mengenai Qbot *mobile robot* diharapkan pada penelitian selanjutnya dapat mengembangkan pergerakan Qbot *mobile robot* ini menggunakan kontroler lain yang sifatnya *adaptive* dengan dilengkapi komponen lain yang mendukung kinerja robot seperti menggunakan kamera *optick track* untuk proses trayektori.

DAFTAR PUSTAKA

[1] ……,"*User Manual Quanser Qbot*", Inovative Educate, Quanser Document Number 830. Revision 7, 2012.

[2] Leotman, Baradiant Ivano, "Desain Kontroler Fuzzy PD untuk Pelacakan Objek pada Qbot Mobile Robot", *Tugas Akhir* Teknik Elektro, Institut Teknologi Sepuluh Nopember, Surabaya, 2014.

[3] Afrizal, "Implementasi Perencanaan Jalur Qbot Mobile Robot Menggunakan Metode Neuro-Fuzzy", *Tugas Akhir* Teknik Elektro, Institut Teknologi Sepuluh Nopember, 2016.

[4] Istichori, "Aplikasi Jaringan Syaraf Tiruan B-Spline pada Sistem Pengaturan Suhu Cairan", *Tugas Akhir* Teknik Elektro, Universitas Diponegoro, 2011.

[5] Qamar, Shahid, Laiq Khan and Saima Ali, "Adaptive B-Spline Based Neuro-Fuzzy Control for Full Car Active Suspension System", *Middle-East Journal of Scientific Research 16(10)*: 1348-1360, 2013.

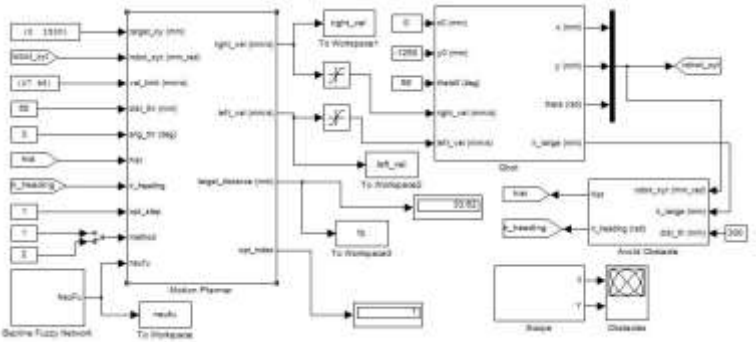
[6] Ye, Wenjun, "B-spline-based Neuro-Fuzzy Velocity Field Navigation and Control for a Nonholonomic Mobile Robot", *Proceedings of the 31st Chinese Cotrol Conference*, July 25-27, 2012

[7] ……."*Kinematics Modeling*", Inovative Educate, Quanser Document Number 830. Revision 7, Agustus, 2012.

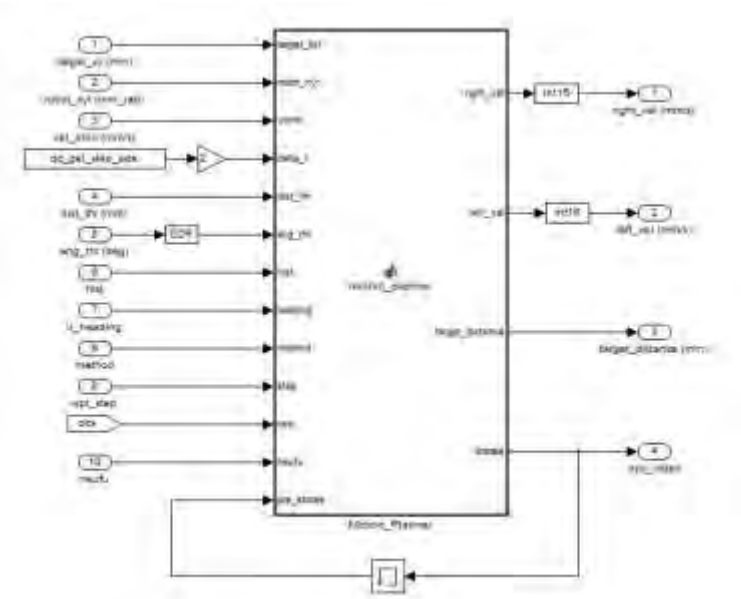
[8] Cong, Shuang and Ruixiang Song, "An Improved B-Spline Fuzzy-Neural Network Controller", *Proceedings of the 3rd World Congress on Intelligent Control and Automation*, June 28-July 2, 2000.

[9] Sayed, Tarek, Arash Travakolie and Abdolmehdi Razavi, "Comparison of Adaptive Network Based Fuzzy Inference Systems and B-spline Neuro-Fuzzy Mode Choice Models", *Journal of Computing in Civil Engineering,* April, 2013.

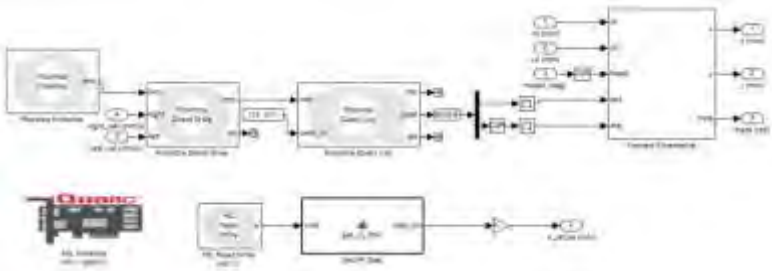
LAMPIRAN A



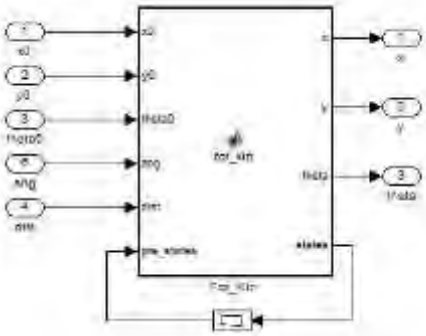
Gambar A.1 Blok Program Keseluruhan



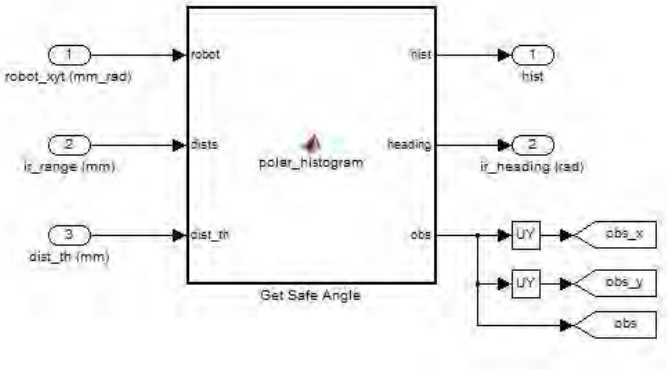
Gambar A.2 Blok Program Utama *Motion Planner*



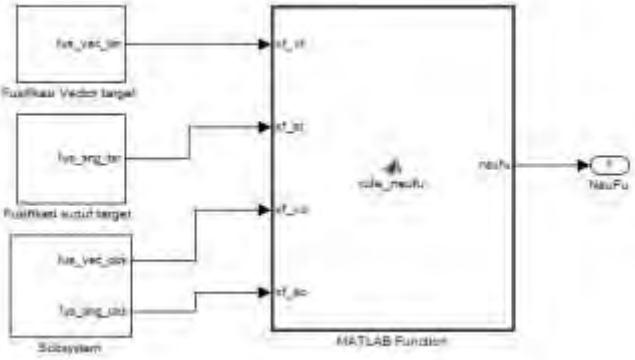
Gambar A.3 Blok *Direct Drive Qbot*



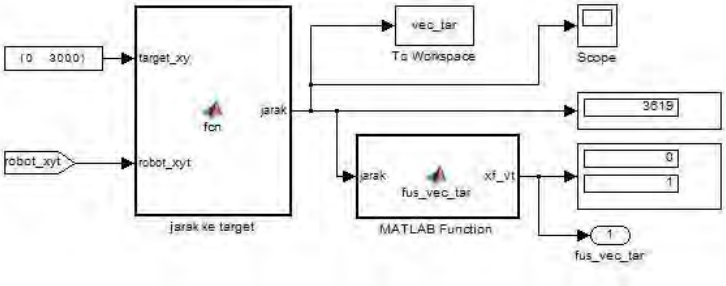
Gambar A.4 Blok Program *Forward Kinematic*



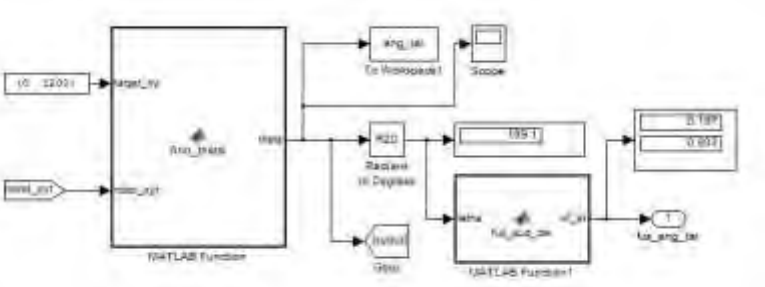
Gambar A.4 Blok *Avoid Obstacle*



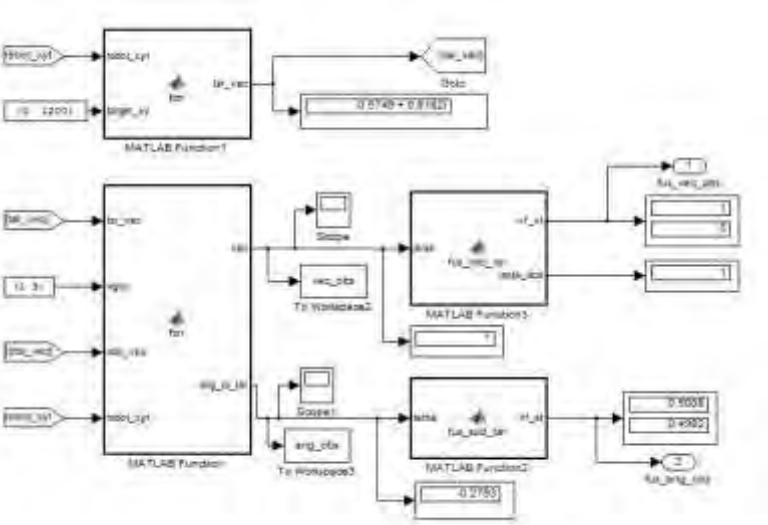
Gambar A.5 Blok Kontrol *Fuzzy B-Spline Neural Network*



Gambar A.6 Blok Fuzzifikasi Jarak ke Target



Gambar A.7 Blok Fuzzifikasi Sudut ke Target



Gambar A.8 Blok Jarak dan Sudut ke *Obstacle*

BAB 1 LAMPIRAN B

1.1 B.1 Program *Motion Planner*

```
function [right_vel, left_vel, target_distance, states] = ...
   motion_planner(target_list, robot_xyt, vlimit, ...
   delta_t, dist_thr, ang_thr, hist, heading, method, step, obs, neufu,...
pre_states)

% Initialize output variables
right_vel = int16(0);
left_vel = int16(0);
states = pre_states;

rx = robot_xyt(1);
ry = robot_xyt(2);
rtheta = check_angle(robot_xyt(3));
target_xy = [rx ry];
[n xy] = size(target_list);
if n==2 && xy==1
   target_xy = [target_list(1,1) target_list(2,1)];
   n = 1;
else
   for i=1:n
      if states(1) == i
         for j=1:xy
            target_xy(j) = target_list(i, j);
         end
      end
   end
end

tx = target_xy(1);
ty = target_xy(2);
target_distance = find_dist(rx, ry, tx, ty);

if ((states(1) == n) && (target_distance <= dist_thr)) %|| states(1) == -1
%        states(1) = -1;
       right_vel = int16(0);
       left_vel = int16(0);
```

```
%       target_distance = 0;
else
   flg = 0;
   nO = size(obs, 1);
   for k=1:nO
      if obs(k,1)~=rx && obs(k,2)~=ry && states(1)~=n
         dist = abs(complex((tx-obs(k,1)), (ty-obs(k,2))));
         if dist <= 500
            flg = 1;
            break;
         end
      end
   end
   if target_distance <= dist_thr || flg==1
      states(1) = pre_states(1) + step;%1;
      if states(1) > n
         states(1) = n;
      end
   else
      [ang_to_tar]= find_theta(rx, ry, rtheta, tx, ty);
      histsum = sum(hist);
      if histsum ~= 0
         ang_to_tar = check_angle(heading - check_angle(rtheta));
      end
      if method == 2
         [right_vel, left_vel] = solve_inv_kin(target_distance, ...
            ang_to_tar, vlimit, delta_t, neufu);
      else
         [right_vel, left_vel] = rotate_and_go(ang_to_tar, ...
            vlimit, ang_thr);
      end

   end
end

return;

function [y] = check_angle(x)
y = x;
```

```
if x > pi
   y = x - 2*pi;
elseif x < -pi
   y = x + 2*pi;
end
return;

function dist = find_dist(rx, ry, tx, ty)
dist = sqrt((rx-tx)^2 + (ry-ty)^2);
return;

function [theta]= find_theta(rx, ry, rtheta, tx, ty)

X = tx - rx;
Y = ty - ry;
theta = atan2(Y, X);
theta = check_angle(theta - check_angle(rtheta));
return;

function [vr, vl] = solve_inv_kin(dist, theta, vlimit, delta_t, neufu)

d = 252.5;
vmax = vlimit(2);
wmax = (2*vmax)/d;
w = theta/delta_t;
w_sign = sign(w);
if abs(w) > wmax
   w = w_sign*wmax;
end
v = dist/delta_t;
if v > vmax
   v = vmax;
end
vr_tmp = (2*v + d*w)/2;
vl_tmp = 2*v - vr_tmp;

max_of_vrvl = abs(max(vr_tmp, vl_tmp));
if max_of_vrvl > vmax
   vr_tmp = (vr_tmp/max_of_vrvl)*vmax;
```

```
    vl_tmp = (vl_tmp/max_of_vrvl)*vmax;
end
vr = int16(vr_tmp)*neufu;
vl = int16(vl_tmp)*neufu;
return;

function [vr, vl] = rotate_and_go(theta, vlimit, ang_thr)

if abs(theta) > ang_thr
    if theta >= 0
        vr = int16(vlimit(1));
        vl = int16(-vlimit(1));
    else
        vr = int16(-vlimit(1));
        vl = int16(vlimit(1));
    end
else
        vr = int16(vlimit(2));
        vl = int16(vlimit(2));
end

return;
```

B.2 Program Kontroler Fuzzy B-Spline Neural Network

```
function neufu = rule_neufu(xf_vt,xf_at,xf_vo,xf_ao)

Q=[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]';
m=0;
for i=1:2
    for j=1:2
        for k=1:2
            for l=1:2
                m=m+1;
                Q(m)=xf_vt(i)*xf_at(j)*xf_vo(k)*xf_ao(l);  %Larsent
                %p=min(xf_vt,xf_at);
                %q=min(xf_vo,xf_ao);
                %Q(m)=min(p,q);              %mamdani
            end
        end
```

```
   end
end
%rul
u=[0 0 0 0 0]';
u(1)=Q(1)+Q(2)+Q(3);
u(2)=Q(4)+Q(5)+Q(6);
u(3)=Q(7)+Q(8)+Q(9)+Q(10);
u(4)=Q(11)+Q(12)+Q(13);
u(5)=Q(14)+Q(15)+Q(16);

%normalisasi
su=u(1)+u(2)+u(3)+u(4)+u(5);
for i=1:5
   u(i)=u(i)/su;
end

w=[2.6604   4.9975   3.7257   5.6011   6.0663];
atas=w(1)*u(1)+w(2)*u(2)+w(3)*u(3)+w(4)*u(4)+w(5)*u(5);
bawah=u(1)+u(2)+u(3)+u(4)+u(5);
defu=atas/bawah;
```

B.3 Program Simulasi B-Spline

```
function example_bsplineapprox

      k = 4;
      t = [0 0 0 0 0.4 0.4 0.6 0.8 0.9 1 1 1];
      w = 2*[0.9 1.5 1.9 0.7 1.8 1.6 3.5 5.4];
      P = [0.4942 0.2398 0.2942 0.4942 0.6573 0.4741 0.4856
0.487;0.05994 0.3132 0.3716 0.4617 0.5711 0.7237 0.7839 0.7939];
      [M_0,x] = bspline_wdeboor(k,t,P,w);

figure;
hold all;
plot(M_0(1,:), M_0(2,:), 'b');
plot(P(1,:), P(2,:), 'r');
hold off;

function [C,u] = bspline_wdeboor(n,t,P,w,u)
w = transpose(w(:));
```

```
P = bsxfun(@times, P, w);
P = [P ; w];  % add weights to control points

if nargin >= 5
   [Y,u] = bspline_deboor(n,t,P,u);
else
   [Y,u] = bspline_deboor(n,t,P);
end

C = bsxfun(@rdivide, Y(1:end-1,:), Y(end,:));

function [C,U] = bspline_deboor(n,t,P,U)
m = size(P,1);  % dimension of control points
t = t(:).';     % knot sequence
U = U(:);
S = sum(bsxfun(@eq, U, t), 2);  % multiplicity of u in t (0 <= s <= d+1)
I = bspline_deboor_interval(U,t);

Pk = zeros(m,d+1,d+1);
a = zeros(d+1,d+1);

C = zeros(size(P,1), numel(U));
for j = 1 : numel(U)
    u = U(j);
    s = S(j);
    ix = I(j);
    Pk(:) = 0;
    a(:) = 0;

    % identify d+1 relevant control points
    Pk(:, (ix-d):(ix-s), 1) = P(:, (ix-d):(ix-s));
    h = d - s;

    if h > 0
        % de Boor recursion formula
        for r = 1 : h
            q = ix-1;
            for i = (q-d+r) : (q-s);
                a(i+1,r+1) = (u-t(i+1)) / (t(i+d-r+1+1)-t(i+1));
```

```
            Pk(:,i+1,r+1) = (1-a(i+1,r+1)) * Pk(:,i,r) + a(i+1,r+1) *
Pk(:,i+1,r);
          end
       end
       C(:,j) = Pk(:,ix-s,d-s+1);  % extract value from triangular
computation scheme
   elseif ix == numel(t)  % last control point is a special case
       C(:,j) = P(:,end);
   else
       C(:,j) = P(:,ix-d);
   end
end

function ix = bspline_deboor_interval(u,t)

i = bsxfun(@ge, u, t) & bsxfun(@lt, u, [t(2:end) 2*t(end)]);  % indicator
of knot interval in which u is
[row,col] = find(i);
[row,ind] = sort(row);  %#ok<ASGLU> % restore original order of data
points
ix = col(ind);
```

Halaman ini sengaja dikosongkan

RIWAYAT PENULIS

Dwi Wulandari adalah nama penulis dan akrab dipanggil dwi. Penulis dilahirkan di Jakarta pada tanggal 5 Juni 1993, merupakan putra bungsu dari dua bersaudara. Penulis memulai pendidikan dari TK Bengawan Jakarta, SD Negeri Rawa Badak Selatan 07 Pagi Jakarta, SMP Negeri 121 Jakarta, dan SMA Trimurti Surabaya. Setelah menyelesaikan pendidikan SMA pada tahun 2011, penulis melanjutkan pendidikan tinggi di D3 Teknik Elektro Institut Teknologi Sepuluh Nopember tepatnya pada program studi Komputer Kontrol dan lulus pada tahun 2014. Selanjutnya pada tahun 2014 penulis meneruskan pendidikan sarjana di perguruan tinggi negeri yang sama yaitu Institut Teknologi Sepuluh Nopember jurusan Teknik Elektro, tepatnya pada bidang studi Teknik Sistem Pengaturan. Pada bulan Juni 2016 penulis mengikuti seminar dan ujian Tugas Akhir sebagai salah satu syarat untuk memperoleh gelar Sarjana Teknik Elektro dari Institut Teknologi Sepuluh Nopember Surabaya.