Institut
Teknologi
Sepuluh Nopember

TUGAS AKHIR - EE 184801

PERANCANGAN SISTEM PEMANDUAN *UNMANNED SURFACE VEHICLE* UNTUK *OBSTACLE AVOIDANCE* MENGGUNAKAN *FUZZY LOGIC CONTROL*

Naufan Hafiyyan
NRP 07111640000150

Dosen Pembimbing
Ir. Rusdhianto Effendie A.K., M.T.
Mochammad Sahal, S.T., M.Sc..

DEPARTEMEN TEKNIK ELEKTRO
Fakultas Teknologi Elektro Dan Informatika Cerdas
Institut Teknologi Sepuluh Nopember
Surabaya 2020

Halaman ini sengaja dikosongkan

TUGAS AKHIR - EE 184801

PERANCANGAN SISTEM PEMANDUAN *UNMANNED SURFACE VEHICLE* UNTUK *OBSTACLE AVOIDANCE* MENGGUNAKAN *FUZZY LOGIC CONTROL*

Naufan Hafiyyan
NRP 07111640000150

Dosen Pembimbing
Ir. Rusdhianto Effendie A.K., M.T.
Mochammad Sahal, S.T., M.Sc..

DEPARTEMEN TEKNIK ELEKTRO
Fakultas Teknologi Elektro Dan Informatika Cerdas
Institut Teknologi Sepuluh Nopember
Surabaya 2020

Halaman ini sengaja dikosongkan

FINAL PROJECT - EE 184801

UNMANNED SURFACE VEHICLE GUIDANCE SYSTEM DESIGN FOR OBSTACLE AVOIDANCE USING FUZZY LOGIC CONTROL

Naufan Hafiyyan
NRP 07111640000150

Supervisor
Ir. Rusdhianto Effendie A.K., M.T.
Mochammad Sahal, S.T., M.Sc..

ELECTRICAL ENGINEERING DEPARTMENT
Faculty of Intelligent Electrical and Informatics Technology
Institut Teknologi Sepuluh Nopember
Surabaya 2020

Halaman ini sengaja dikosongkan

PERNYATAAN KEASLIAN
TUGAS AKHIR

Dengan ini menyatakan bahwa isi sebagian maupun keseluruhan Tugas Akhir saya dengan judul "**Perancangan Sistem Pemanduan *Unmanned Surface Vehicle* untuk *Obstacle Avoidance* Menggunakan *Fuzzy Logic Control***" adalah hasil karya intelektual mandiri, diselesaikan tanpa menggunakan bahan-bahan yang tidak diizinkan dan bukan merupakan karya pihak lain yang saya akui sebagai karya sendiri.

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

Surakarta, Agustus 2020

Naufan Hafiyyan
NRP. 07111640000150

Halaman ini sengaja dikosongkan

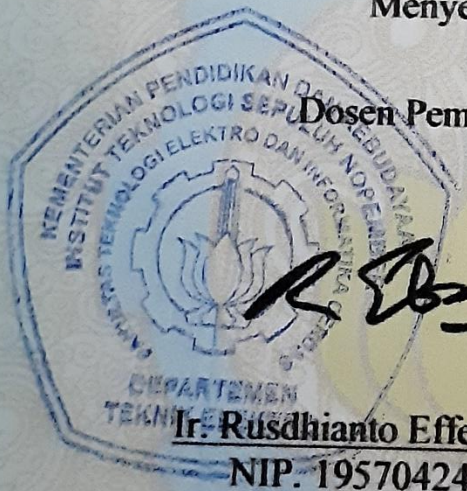
PERANCANGAN SISTEM PEMANDUAN *UNMANNED SURFACE VEHICLE* UNTUK *OBSTACLE AVOIDANCE* MENGGUNAKAN *FUZZY LOGIC CONTROL*

TUGAS AKHIR

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

Menyetujui:

Dosen Pembimbing I



Ir. Rusdhianto Effendie A.K., M.T.
NIP. 195704241985021001

Surabaya,
Juli 2020

PERANCANGAN SISTEM PEMANDUAN *UNMANNED SURFACE VEHICLE* UNTUK *OBSTACLE AVOIDANCE* MENGGUNAKAN *FUZZY LOGIC CONTROL*

TUGAS AKHIR

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

Menyetujui:

Dosen Pembimbing II

Mochammad Sahal, S.T., M.Sc..
NIP. 197011191998021002

Surabaya,
Juli 2020

PERANCANGAN SISTEM PEMANDUAN *UNMANNED SURFACE VEHICLE* UNTUK *OBSTACLE AVOIDANCE* MENGGUNAKAN *FUZZY LOGIC CONTROL*

Naufan Hafiyyan
07111640000150

Dosen Pembimbing : 1. Ir. Rusdhianto Effendie A.K., M.T.
2. Mochammad Sahal, S.T., M.Sc..

ABSTRAK

Sistem kemudi otomatis merupakan suatu hasil perkembangan teknologi yang dapat diterapkan pada alat transportasi baik di darat, laut, maupun udara. *Unmanned Surface Vehicle* (USV) merupakan salah satu bentuk penerapan dari teknologi sistem kemudi otomatis pada kendaraan laut. *Unmanned Surface Vehicle* (USV) merupakan alat transportasi laut yang beroperasi tanpa perlu campur tangan manusia dalam operasinya. Untuk menunjang operasi kapal tanpa awak, diperlukan sistem pemanduan yang memiliki kemampuan menghindari halangan. Pada penelitian ini akan dibahas mengenai sistem pemanduan *Unmanned Surface Vehicle* (USV) menggunakan algoritma *Fuzzy Logic Control* untuk menghindari halangan yang ada. *Fuzzy Logic Control* memiliki aturan-aturan yang berguna untuk mengambil keputusan guna mengaktifkan sistem *obstacle avoidance*. Untuk menguji keandalan sistem, dilakukan simulasi di aplikasi MATLAB. Pada pengujian yang dilakukan di penetilian ini didapatkan bahwa sistem dalam menghindari halangan dalam lintasan yang telah disimulasikan dengan jarak paling dekat sebesar 15,84 meter di mana nilainya lebih besar dari jarak minimum yang telah ditentukan (13 meter) dengan kondisi jarak aman ideal untuk mengaktifkan *Fuzzy Logic Control* sebesar 20 meter.

Kata Kunci : USV, Pemanduan, *Obstacle Avoidance*, *Fuzzy Logic Control*

Halaman ini sengaja dikosongkan

UNMANNED SURFACE VEHICLE GUIDANCE SYSTEM DESIGN FOR OBSTACLE* AVOIDANCE USING *FUZZY LOGIC CONTROL

Naufan Hafiyyan
07111640000150

Supervisor : 1. Ir. Rusdhianto Effendie A.K., M.T.
2. Mochammad Sahal, S.T., M.Sc..

ABSTRACT

Autopilot system is a technology innovation that has been applied in various autopilot vehicle. Unmanned Surface Vehicle is one of the implementation form of naval vehicle using autopilot system, without using any manual control crew in its operation. To support the autopilot operation, it needs a guidance system that have the ability to avoid obstacles. This research be reviewing the guidance system of Unmanned Surface. The guidance system is designed to perform obstacle avoidance. Unmanned Surface Vehicle uses Fuzzy Logic Control algorithm to avoid the obstacles that detected by the obstacle detection system. The Fuzzy Logic Control Algorithm uses the rules to create decisions that will create the decision for obstacle avoidance. For the purposes of system validation a simulation will be performed using MATLAB *software. The system in this research was able to avoid the obstacle properly in many trials with 15,84 meter as the minimum distance from USV to obstacle detected which was considered as a success trial because it surpassed the safety distance limit (13 meter) and the system performed well when the distance that triggered Fuzzy Logic Control set into 20 meter.*

Kata Kunci : USV, Guidance, *Obstacle Avoidance, Fuzzy Logic Control*

Halaman ini sengaja dikosongkan

KATA PENGANTAR

Puji syukur penulis panjatkan kepada Tuhan YME yang telah memberikan rahmat dan karunianya, sehingga penulis dapat membuat dan menyelesaikan tugas akhir ini dengan semestinya serta tepat waktu.

Kegiatan tugas akhir ini termasuk salah satu mata kuliah yang wajib ditempuh di Departemen Teknik Elektro Institut Teknologi Sepuluh Nopember ini. Penelitian yang penulis lakukan mengambil topik Perancangan Sistem Pemanduan *Unmanned Surface Vehicle* untuk *Obstacle Avoidance* Menggunakan *Fuzzy Logic Control*. Laporan tugas akhir ini disusun untuk melengkapi hasil capaian dari penelitian yang telah dilaksanakan.

Dalam penyusunan skripsi ini banyak hambatan serta rintangan yang penulis hadapi namun pada akhirnya dapat melaluinya berkat adanya bimbingan dan bantuan dari berbagai pihak baik secara moral dan spiritual. Untuk itu pada kesempatan ini penulis menyampaikan ucapan terimakasih kepada:

1. Kedua Orang Tua beserta keluarga penulis yang senantiasa memberi dukungan dan fasilitas penunjang untuk menyelesaikan Tugas Akhir.
2. Bapak Ir. Rusdhianto Effendie A.K., M.T. selaku dosen pembinbing 1 yang senantiasa memberi dukungan dan arahan kepada penulis untuk menyelesaikan Tugas Akhir.
3. Bapak Mochammad Sahal, S.T., M.Sc. selaku dosen pembimbing 2 dan dosen wali yang senantiasa memberi dukungan dan arahan kepada penulis untuk menyelesaikan Tugas Akhir.
4. Kedua sahabat penulis, Adrian Aryaputra Firmansyah dan Fian Ilham Pratama yang telah memberi dukungan penulis untuk menyelesaikan Tugas Akhir.
5. Teman – teman bidang studi Teknik Sistem Pengaturan yang selalu menemani dan menghibur penulis ketika menulis Tugas Akhir.
6. Teman - teman dekat wanita penulis yang tidak dapat disebutkan namanya yang selalu memberi dukungan moral kepada penulis untuk menyelesaikan Tugas Akhir.
7. Pihak-pihak lain yang memberi dukungan kepada penulis untuk menyelesaikan Tugas Akhir.

Penulis menyadari bahwa banyak kekurangan dari laporan tugas akhir ini, baik dari segi materi maupun teknis penyajiannya, mengingat masih kurangnya pengetahuan dan pengalaman penulis. Namun pengalaman dan wawasan baru yang penulis banyak dapatkan juga patut untuk penulis syukuri dan diterapkan menjadi ilmu yang manfaat.

Surakarta, 18 Agustus 2020

Naufan Hafiyyan

DAFTAR ISI

DAFTAR GAMBAR

Halaman ini sengaja dikosongkan

DAFTAR TABEL

BAB 1
PENDAHULUAN

Pada bab ini akan dipaparkan tentang latar belakang penelitian, permasalahan yang akan diselesaikan, tujuan dari penelitian, metodologi yang digunakan, sistematika penulisan serta relevansi penelitian ini terhadap penelitian sejenis atau lebih rumit di masa mendatang.

1.1 Latar Belakang

Indonesia merupakan sebuah negara kepulauan dengan dua pertiga luas lautan lebih besar daripada daratan. Banyak ancaman datang, seperti yang dilakukan negara tetangga yang melakukan illegal fishing ataupun klaim sepihak atas pulau terluar. Oleh karena itu, sudah selayaknya, Indonesia memiliki kekuatan dalam menjaga keamanan serta pertahanan yang memadai. Namun karena melihat kondisi seperti saat ini, TNI AL masih banyak membutuhkan kelengkapan armada berteknologi tinggi untuk lebih meningkatkan pertahanan.

Dalam perkembangan teknologi kini, berkembang pula metode kendali tanpa awak. Untuk beroperasi di perairan, telah dikembangkan *Unmanned Surface Vehicle* (USV) atau kapal tanpa awak yang mampu beroperasi tanpa nahkoda. Dengan berkembangnya teknologi robot kapal permukaan atau yang disebut *Unmanned Surface Vehicle* (USV), maka tugas manusia dapat digantikan dalam melakukan tugas-tugasnya yang berarti dapat mengurangi resiko keselamatan dari personel yang bertugas. Sedangkan untuk dapat beroperasi tanpa nahkoda, dibutuhkan sistem pemanduan yang memiliki kemampuan obstacle avoidance terhadap halangan dinamis supaya *Unmanned Surface Vehicle* (USV) dapat beroperasi dengan optimal. Oleh karena itu, pada penelitian ini dilakukan perancangan sistem pemanduan untuk obstacle avoidance yang dapat diterapkan pada *Unmanned Surface Vehicle* (USV)..

1.2 Perumusan Masalah

Dibutuhkan sistem pemanduan untuk dapat menghindari halangan dan dapat diimplementasikan ke Unmanned Surface Vehicle (USV).

1.3 Batasan Masalah

Batasan masalah dalam penelitian ini adalah menciptakan sistem *obstacle avoidance* yang mampu menghindari halangan statis, dinamis, dan kombinasi statis dan dinamis yang disimulasikan dalam lintasan USV.

1.4 Tujuan

Tujuan dari penelitian ini adalah mampu membuat sistem *Obstacle Avoidance* yang berbasis pada *Fuzzy Logic Control* dan dapat diterapkan pada Unmanned Surface Vehicle (USV) yang memiliki kemampuan menghindari halangan dalam beberapa keadaan yang memungkinkan.

1.5 Metode Penelitian

Penelitian ini akan dilaksanakan dengan melalui beberapa tahap proses yang telah dirancang sebagai berikut :

1. Studi Literatur

Pada tahap studi literatur, akan dipelajari dasar teori dan kajian mengenai konsep pemanduan, *obstacle avoidace*, dan *fuzzy logic control* yang berasal dari sumber yang terpercaya dan relevan untuk mendukung perancangan tugas akhir ini.

2. Perancangan Sistem Pemanduan

Pada tahap ini dilakukan perancangan sistem pemanduan waypoint yang akan digunakan berdasarkan model dan spesifikasi *Unmanned Surface Vehicle* (USV) yang digunakan dalam tugas akhir ini.

3.Perancangan Sistem *Obstacle Avoidance* berbasis *Fuzzy Logic Control*

Pada tahap ini dilakukan perancangan sistem obstacle avoidance yang memiliki algoritma perencanaan jalur Unmanned Surface Vehicle (USV) bebas tabrakan dari halangan statis dan dinamis. Pada tahapan ini terdapat perancangan halangan statis, dinamis, dan kombinasi. Di mana halangan dirancang agar bertabrakan Unmanned Surface Vehicle (USV) pada suatu titik tertentu. *Algoritma obstacle avoidance* akan memilih jalur menghindari halangan. Algoritma yang akan digunakan adalah *Fuzzy Logic Control.*

4.Tahap Simulasi

Pada tahap ini dilakukan pengujian tiap bagian dari sistem maupun keseluruhan sistem, apabila masih terdapat kesalahan yang terdeteksi pada sistem maka akan dilakukan revisi kembali pada sistem yang ada. Pengujian dilakukan pada sistem pemanduan yang sudah dilengkapi dengan kemampuan obstacle avoidance.

5.Analisis Data dan Evaluasi

Pada tahap ini akan dilakukan analisis terhadap data yang sudah didapatkan sehingga mampu mengenali karakteristik dan cara kerja sistem. Analisis data dilakukan pada hasil data yang didapatkan dari hasil simulasi sistem pada *Unmanned Surface Vehicle* (USV), apakah kinerja

sistem sudah sesuai yang diinginkan. Apabila hasil dari kinerja keseluruhan sistem masih belum sesuai hasil yang diinginkan maka akan dilakukan revisi kembali.

6.Kesimpulan

Kesimpulan didapatkan dari hasil analisis data yang didapatkan dari pengujian dan referensi yang terkait. Pada tahapan ini akan ditarik kesimpulan apakah hasil penelitian sudah mampu mengatasi permasalahan yang ada di perumusan masalah.

7.Penyusunan Buku Tugas Akhir

Pada tahap ini akan dilakukan penyusunan buku laporan sesuai hasil proses Tugas Akhir yang telah dilakukan selama proses penelitian. Penyusunan buku tugas akhir dilakukan sebagai bentuk laporan tertulis.

1.6 Sistematika Penulisan

Sistematika penulisan yang diterapkan pada buku tugas akhir ini terbagi menjadi lima bab, yaitu:

BAB 1 PENDAHULUAN

Bab ini membahas tentang latar belakang, permasalahan yang ada, batasan masalah, tujuan pelaksanaan tugas akhir, metodologi pelaksanaan, sistematika penulisan laporan tugas akhir, dan relevansi.

BAB 2 TEORI PENUNJANG

Bab ini membahas tinjauan pustaka yang membantu penelitian, diantaranya konsep dasar dari Un*manned Surface Vehicle* (USV), Dubins *Path Planning*, dan *Fuzzy Logic Control*.

BAB 3 PERANCANGAN SISTEM *OBSTACLE AVOIDANCE*

Bab ini membahas persamaan gerak USV, perancangan sistem *obstacle avoidance* berbasis *Fuzzy Logic Control* dan perancangan sistem estimasi tabrakan.

BAB 4 PENGUJIAN SISTEM *OBSTACLE AVOIDANCE*

Bab ini memuat hasil simulasi sistem *obstacle avoidance* pada USV dalam berbagai kondisi lintasan. Setelah simulasi dijalankan, dilanjutkan dengan tahap menganalisa hasil simulasi dari sistem *obstacle avoidance* yang telah dirancang.

BAB 5 PENUTUP

Bab ini akan dijelaskan suatu kesimpulan dan saran untuk ke depannya terkait hasil yang telah dicapai pada penelitian yang telah dilakukan.

1.7 Relevansi

Hasil yang diperoleh dari Tugas Akhir ini diharapkan menjadi referensi pengembangan teknologi pemanduan dengan *obstacle avoidance* menggunakan *Fuzzy Logic Control* yang dapat diimplementasikan pada *Unmanned Surface Vehicle* (USV) di masa depan.

BAB 2
TEORI PENUNJANG

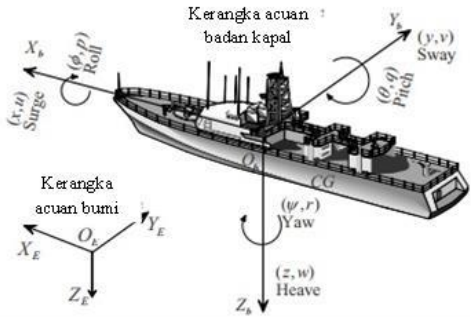
2.1 *Unmanned Surface Vehicle* (USV)

Unmanned surface vehicle atau kapal tanpa awak adalah kapal yang memiliki kemampuan untuk melaju tanpa ada nahkoda pada kapal tersebut. Kapal dapat berjalan secara manual dengan kendali jarak jauh. Kapal dapat melaju semi otomatis dalam pantauan operator. Kapal dapat melaju secara otomatis penuh tanpa pantauan operator .

USV memiliki banyak manfaat untuk manusia, antara lain bisa digunakan untuk mengintai teritorial air musuh, bisa digunakan untuk pemetaan kontur bumi dibawah air, dan masih banyak manfaat yang lain. Dalam kenyataannya, nama USV diklasifikasikan sesuai jumlah lambung yang digunakan. USV Monohull untuk USV yang memiliki satu lambung, Katamaran untuk dua lambung, dan Trimaran untuk USV yang menggunakan 3 lambung kapal.

Sama halnya dengan kapal pada umumnya, USV digerakkan dengan mesin thruster dan kemudi. Mesin thruster digunakan untuk memberi gaya dorong kepada kapal, dan kemudi digunakan untuk mengatur arah kapal melaju.

Pergerakan unmanned surface vehicle umum dinyatakan dalam 6 derajat kebebasan, seperti pada Gambar 1. Tiga koordiant pertama (x, y, z) dan turunan pertamanya untuk menyatakan posisi dan pergerakan translasi USV umum, sedangkan tiga koordinat terakhir (Φ, θ, Ψ) dan turunan pertamanya untuk menyatakan arah dan pergerakan rotasi USV umum. Notasi pada USV umum dapat dilihat pada Tabel 1.



Gambar 2.1 Derajat Kebebasan USV

Tabel 2.1 6 Derajat Kebebasan USV

Derajat Kebebasan	Nama	Gaya dan Momen	Kecepatan Linier dan Kecepatan Sudut	Posisi dan Sudut Euler
1	*Surge*	*X*	*u*	*x*
2	*Sway*	*Y*	*v*	*y*
3	*Heave*	*Z*	*w*	*z*
4	*Roll*	*K*	*p*	Φ
5	*Pitch*	*M*	*q*	θ
6	*Yaw*	*N*	*r*	Ψ

Variabel derajat kebebasan USV umum dapat dinyatakan dengan vektor – vektor berikut :

dimana,

$$\boldsymbol{\eta} = [\boldsymbol{\eta_1}\ \boldsymbol{\eta_2}]^T, \quad \boldsymbol{\eta_1} = [x\ y\ z]^T, \quad \boldsymbol{\eta_2} = [\phi\ \theta\ \psi]^T$$
$$\boldsymbol{v} = [\boldsymbol{v_1}\ \boldsymbol{v_2}]^T, \quad \boldsymbol{v_1} = [u\ v\ w\ ]^T, \quad \boldsymbol{v_2} = [p\ q\ r\ ]^T$$
$$\boldsymbol{\tau} = [\boldsymbol{\tau_1}\ \boldsymbol{\tau_2}]^T, \quad \boldsymbol{\tau_1} = [X\ Y\ Z\ ]^T, \quad \boldsymbol{\tau_2} = [K\ M\ N\ ]^T$$

$\boldsymbol{\eta}$: vektor posisi dan orientasi pada kerangka acuan bumi (earth-fixed)

$\boldsymbol{v}$: vektor kecepatan linier dan kecepatan sudut pada kerangka acuan badan kapal (body-fixed)

$\boldsymbol{\tau}$: gaya dan momen yang bekerja pada USV pada kerangka acuan badan kapal (body-fixed)

2.2 Dubins Path

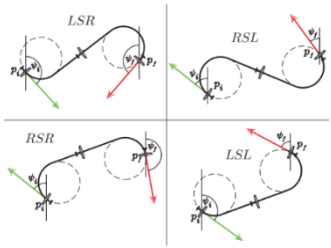
2.2.1 Pengertian Dubins Path

Metode *Dubins path* merupakan salah satu teknik untuk merancang trayektori atau lintasan. Trayektori deterministik yang dihasilkan dari *Dubins path* ini kemudian akan menjadi trayektori yang akan dilalui oleh USV dari sejak titik keadaan awal sampai dengan titik terakhir dari lintasan yang telah diketahui. Pada metode *Dubins path* ini, trayektori yang dibangun terdiri dari gabungan trayektori garis lurus dan busur lingkaran atau *arc*. Lintasan *Dubins* terpendek dapat dicapai ketika radius busur lingkaran minimum dan garis merupakan garis lurus. Dalam geometri *euclidean*, *Dubins path* dihasilkan dari menarik garis singgung atau *tangent* antar 2 busur lingkaran. Garis singgung ini dapat

menghubungkan 2 busur lingkaran secara eksternal maupun internal (diagonal).

Tahap pertama pada metode *Dubins path* adalah menentukan tipe lintasan apa yang akan digunakan.Terdapat empat macam kombinasi lintasan dari *Dubins path* untuk bisa membangun trayektori yang diinginkan, dimana masing-masing kombinasi tersusun dari 2 segmen melingkar dan 1 garis lurus seperti dapat dilihat pada gambar 1.

Dapat dilihat pula dari gambar 1 bahwa 4 macam lintasan *Dubins* adalah LSR, RSL, RSR dan LSL. LSR merupakan lintasan *left straight right* yang sesuai namanya merupakan lintasan berbelok kiri, kemudian lurus, diakhiri dengan belok kanan. RSL adalah lintasan *right straight left* yaitu lintasan kanan, lurus, diakhiri dengan belok kiri. RSR adalah lintasan *right straight right* yaitu lintasan berbelok kanan, lurus, dan kanan. LSL merupakan *left straight left* dimana lintasan yang terbentuk adalah berbelok kiri, kemudian lurus, dan diakhiri belokan ke kiri.



Gambar 2.2 Lintasan Dubins

Setelah menentukan bentuk lintasan yang diinginkan, kemudian mendapatkan konfigurasi awal dan akhir dari objek bergerak, yang dalam penelitian ini merupakan USV. Konfigurasi yang dimaksud adalah posisi awal P_s dan akhir P_f.

2.2.2 Algoritma Dubins Path

Dalam proses menentukan lintasan dubins dibutuhkan langkah-langkah untuk mencari nilai parameter yang digunakan. Terdapat algortima dasar dari *Dubins Path.*

Langkah Pertama ialah menentukan posisi awal P_s dan posisi akhir P_f dalam koordinat (x, y), sudut arah atau *heading* USV (θ), dan radius putar minimum ρ. Secara matematis, konfigurasi awal dan akhir adalah $P_s(x_s, y_s, \theta_s)$ dan $P_f(x_f, y_f, \theta_f)$.

Cara menghitung θ_s

Jika $(X_f - X_s) > 0$ dan $(Y_f - Y_s) \leq 0$ maka $\theta_s = -90° - \arctan\left(\frac{|Y_f - Y_s|}{|X_f - X_s|}\right)$

Jika $(X_f - X_s) > 0$ dan $(Y_f - Y_s) > 0$ maka $\theta_s = -\arctan\left(\frac{|X_f - X_s|}{|Y_f - Y_s|}\right)$

Jika $(X_f - X_s) < 0$ dan $(Y_f - Y_s) > 0$ maka $\theta_s = \arctan\left(\frac{|X_f - X_s|}{|Y_f - Y_s|}\right)$

Jika $(X_f - X_s) < 0$ dan $(Y_f - Y_s) \leq 0$ maka $\theta_s = 90° + \arctan\left(\frac{|Y_f - Y_s|}{|X_f - X_s|}\right)$

Menentukan titik pusat dari lingkaran awal $O_s(x_{cs}, y_{cs})$ dan akhir $O_f(x_{cf}, y_{cf})$ dengan jari-jari minimum yang dapat dibentuk USV yaitu 24 meter. Untuk titik awal dianggap tidak memerlukan lingkaran sehingga hanya dicari titik pusat lingkaran pada waypoint selanjutnya (P_f) sehingga dianggap jari-jari lingkaran 0 meter. Untuk waypoint ketiga dan selanjutnya, X_{cs} dan Y_{cs} sama dengan X_{cf} dan Y_{cf} waypoint sebelumnya. Terdapat dua kondisi untuk menetukan titik pusat lingkaran yaitu:

Jika sudut heading saat ini lebuh besar dari sudut heading setelahnya, maka

$X_{cf} = X_f - \rho \cos(\theta_f + \pi)$

$Y_{cf} = Y_f + \rho \, sin(\theta_f + \pi)$

Jika sudut heading saat ini lebuh kecil dari sudut heading setelahnya, maka

$X_{cf} = X_f - \rho \cos(\theta_s)$

$Y_{cf} = Y_f + \rho \, sin(\theta_s)$

Mencari jarak antar titik pusat lingkahran menggunakan Persamaan:

$$|c| = \sqrt{(x_{cs} - x_{cf})^2 + (y_{cs} - y_{cf})^2}$$

Menghitung sudut kemiringan dari garis antar titik pusat lingkaran menggunakan rumus pada Persamaan :

$$\psi = \arctan\left(\frac{y_{cf} - y_{cs}}{x_{cf} - x_{cs}}\right)$$

Menghitung sudut antara garis $O_s O_f$ dan $O_s T'$ menggunakan Persamaan :

$$\phi_e = \arcsin\left(\frac{\rho_f - \rho_s}{|c|}\right)$$

Mencari sudut ϕ_{ex} yaitu sudut *exit* dan ϕ_{en} adalah sudut *entry*

$$\phi_{ex} = \phi_{es} + \frac{\pi}{2} + \psi$$

$$\phi_{en} = \phi_{ef} + \frac{\pi}{2} + \psi$$

Menentukan titik T_{ex} (*tangent* exit) yang merupakan titik akhir dari lintasan busur lingkaran awal sebelum memasuki lintasan garis lurus serta titik T_{en} (*tangent* entry) yang merupakan titik awal dari lintasan busur lingkaran akhir setelah lintasan garis lurus menggunakan rumus pada Persamaan

$$T_{ex} = (x_{cs} + \rho_s \cos(\phi_{ex}), y_{cs} + \rho_s \sin(\phi_{ex}))$$

$$T_{en} = \left(x_{cf} + \rho_f \cos(\phi_{en}), y_{cf} + \rho_f \sin(\phi_{en})\right)$$

2.3 *Fuzzy Logic Control*

Logika Fuzzy merupakan suatu cara yang tepat untuk memetakan suatu ruang input ke dalam ruang output. Untuk sistem yang sangat rumit, penggunaan logika fuzzy (*fuzzy logic*) adalah salah satu pemecahannya. Sistem tradisional dirancang untuk mengontrol keluaran tunggal yang berasal dari beberapa masukan yang tidak saling berhubungan. Karena ketidaktergantungan ini, penambahan masukan yang baru akan memperumit proses kontrol dan membutuhkan proses perhitungan kembali dari semua fungsi . Kebalikannya, penambahan masukan baru pada sistem fuzzy, yaitu sistem yang bekerja berdasarkan prinsip-prinsip logika fuzzy, hanya membutuhkan penambahan fungsi keanggotaan yang baru dan aturan-aturan yang berhubungan dengannya.

Secara umum, sistem fuzzy sangat cocok untuk penalaran pendekatan terutama untuk sistem yang menangani masalah-masalah yang sulit didefinisikan dengan menggunakan model matematis. Contohnya, nilai masukan dan parameter sebuah sistem bersifat kurang akurat atau kurang jelas, sehingga sulit mendefinisikan model matematikanya.

Sistem fuzzy memiliki beberapa keuntungan bila dibandingkan dengan sistem logika tradisional, salah satunya pada jumlah aturan yang

dipergunakan. Pemrosesan awal sejumlah besar nilai menjadi sebuah nilai derajat keanggotaan pada sistem fuzzy mengurangi jumlah nilai menjadi sebuah nilai derajat keanggotaan pada sistem fuzzy mengurangi jumlah nilai yang harus dipergunakan pengontrol untuk membuat suatu keputusan. Keuntungan lainnya adalah sistem fuzzy mempunyai kemampuan penalaran yang mirip dengan kemampuan penalaran manusia. Hal ini disebabkan karena sistem fuzzy mempunyai kemampuan untuk memberikan respon berdasarkan informasi yang bersifat kualitatif, tidak akurat, dan ambigu.

Logika fuzzy adalah peningkatan dari logika Boolean yang berhadapan dengan konsep kebenaran sebagian. Dimana logika klasik (crisp) menyatakan bahwa segala hal dapat diekspresikan dalam istilah binary (0 atau 1, hitam atau putih, ya atau tidak). Logika fuzzy menggantikan kebenaran Boolean dengan tingkat kebenaran. Logika fuzzy memungkinkan nilai keanggotaan antara 0 dan 1, tingkat keabuan dan juga hitam dan putih, dan dalam bentuk linguistic, konsep tidak pasti seperti "sedikit", "lumayan", dan "sangat".

Sistem fuzzy pertama kali diperkenalkan oleh Prof. L. A. Zadeh dari Barkelay pada tahun 1965. Sistem fuzzy merupakan penduga numerik yang terstruktur dan dinamis. Sistem ini mempunyai kemampuan untuk mengembangkan sistem intelijen dalam lingkungan yang tak pasti. Sistem ini menduga suatu fungsi dengan logika fuzzy. Dalam logika fuzzy terdapat beberapa proses yaitu penentuan himpunan fuzzy, penerapan aturan IF-THEN dan proses inferensi fuzzy (Marimin, 2005:10).

2.3.1 Himpunan *Crisp*

Pada dasarnya, teori himpunan fuzzy merupakan perluasan dari teori himpunan klasik. Pada terori himpunan klasik (*crisp*), keberadaan suatu elemen pada suatu himpunan A hanya akan memiliki 2 kemungkinan keanggotaan, yaitu menjadi anggota A atau tidak menjadi anggota A (Chak, 1998). Suatu nilai yang menunjukkan seberapa besar tingkat keanggotaan suatu elemen (x) dalam suatu himpunan (A), sering dikenal dengan nama nilai keanggotaan, dinotasikan dengan $\mu_A(x)$. Pada himpunan klasik, hanya ada 2 nilai keanggotaan, yaitu $\mu_A(x) = 1$ untuk x menjadi anggota A; dan $\mu_A(x) = 0$ untuk x bukan anggota dari A.

2.3.2 Himpunan Fuzzy

Teori himpunan Fuzzy diperkenalkan oleh Lotfi A. Zadeh pada tahun 1965. Zadeh memberikan definisi tentang himpunan fuzzy dengan mengumpamakan X. Jika X adalah korelasi dari obyek-obyek yang

dinotasikan seara generic oleh x, maka suatu himpunan fuzzy˜, dalam X adalah suatu himpunan pasangan berurutan:

$$\tilde{A} = * \{(x, \mu_{\tilde{A}}(x)) | x \in X\}$$

Dengan $\mu_{\tilde{A}}(x)$ adalah derajat keanggotaan x di $\tilde{A}$ yang memetakan X ke ruang keanggotaan M yang terletak pada rentang [0, 1].

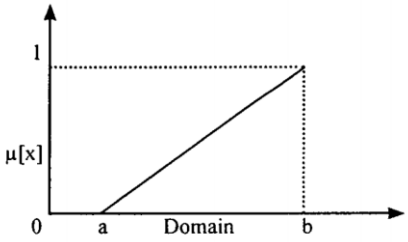
2.4 Fungsi Keanggotaan Fuzzy

Fungsi keanggotaan (membership function) adalah suatu kurva yang menunjukkan pemetaan titik-titik input data ke dalam nilai keanggotaannya. Salah satu cara yang dapat digunakan untuk mendapatkan nilai keanggotaan adalah dengan melalui pendekatan fungsi. Terdapat beberapa jenis yang bisa digunakan untuk menyatakan fungsi keanggotaan pada fuzzy.

2.4.1 Representasi Linier

Pada representasi linear, pemetaan input ke derajat keanggotaannya digambarkan sebagai suatu garis lurus. Bentuk ini paling sederhana dan menjadi pilihan yang baik untuk mendekati suatu konsep yang kurang jelas.

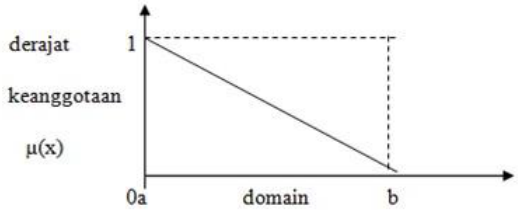
Ada 2 keadaan himpunan fuzzy yang linear. Pertama, kenaikan himpunan dimulai pada nilai domain yang memiliki derajat keanggotaan nol [0] bergerak ke kanan menuju ke nilai domain yang memiliki derajat keanggotaan lebih tinggi seperti yang digambarkan pada Gambar 2.3.



Gambar 2.3 Representasi Linier Kanan ke Kiri

$$\mu(x) = \begin{cases} 0; \\ \dfrac{x-a}{b-a}; \\ 1; \end{cases}$$

Keadaan yang kedua meruapakan kebalikan dari keadaan pertama. Garis lurus dimulai dari nilai domain dengan derajat keanggotaan tertinggi pada sisi kiri, kemudian bergerak menurun ke nilai domain yang memiliki derajat keanggotaan lebih rendah Gambar 2.4,



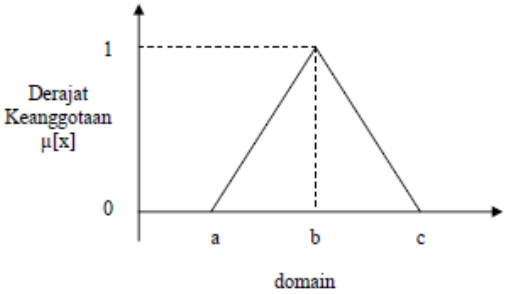
Gambar 2.4 Representasi Linier Kiri ke Kanan

Dengan fungsi keanggotaan:

$$\mu(x) = \begin{cases} \dfrac{b-x}{b-a}; a \le x \le b \\ 0; x \ge b \end{cases}$$

2.4.2 Representasi Segitiga

Kurva segitiga pada dasarnya merupakan gabungan antara 2 garis (linear) seperti terlihat pada Gambar 2.5.



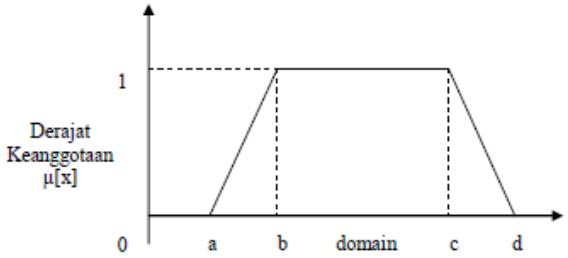
Gambar 2.5 Representasi Segitiga

Walaupun tersusun atas gabungan dua garis linier, representasi kurva segitiga memiliki persamaan yang beda dengan kurva linier. Persamaan fungsi keanggotaan dinyatakan sebagai berikut:

$$\mu(x) = \begin{cases} 0; x \leq a \; atau \; x \geq c \\ \dfrac{x-a}{b-a}; a \leq x \leq b \\ \dfrac{b-x}{c-b}; b \leq x \leq c \end{cases}$$

2.4.3 Representasi Trapesium

Kurva trapesium pada dasarnya seperti bentuk segitiga, hanya saja ada titik yang memiliki nilai keanggotaan 1 seperti yang terlihat pada Gambar 2.6.



Gambar 2.6 Representasi Trapesium

$$(x) = \begin{cases} 0; x \leq a \; atau \; x \geq c \\ \dfrac{x-a}{b-a}; a \leq x \leq b \\ 1; \quad b \leq x \leq c \\ \dfrac{b-x}{c-b}; \quad x \geq d \end{cases}$$

2.5 Metode dalam Fuzzy Logic Control

Terdapat beberapa metode yang dapat digunakan untuk merepresentasikan hasil logika fuzzy yaitu metode Tsukamoto, Sugeno dan Mamdani.

2.5.1 Metode Tsukamoto

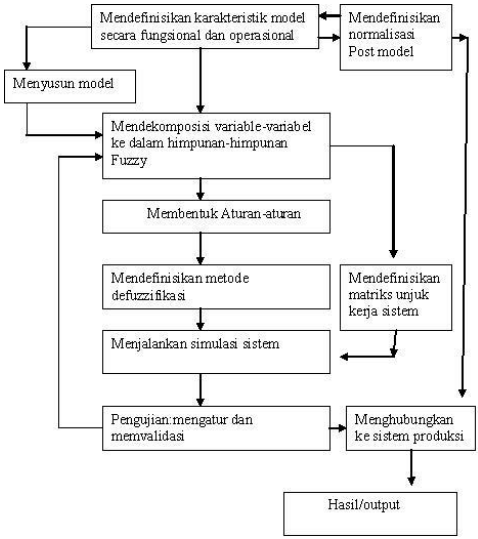
Pada metode Tsukamoto, setiap konsekuen direpresentasikan dengan himpunan fuzzy dengan fungsi keanggotaan monoton. Output hasil inferensi masing-masing aturan adalah z, berupa himpunan biasa (*crisp*) yang ditetapkan berdasarkan -predikatnya. Hasil akhir diperoleh dengan menggunakan rata-rata terbobotnya.(Sri Kusumadewi,2002:108)

Pada metode fuzzy Tsukamoto, proses inferensi dilakukan dengan aturan (rule) berbentuk IF-THEN dan menggunakan operasi AND, dimana akan dipilih nilai yang lebih minimum (MIN) dari dua variabel yang ada.

2.5.2 Metode Mamdani

Pada metode Mamdani atau yang biasa disebut dengan metode MAX-MIN, aplikasi fungsi implikasi menggunakan *MIN*, sedang komposisi aturan menggunakan metode *MAX*. Metode Mamdani dikenal juga dengan metode *MAX-MIN*. Inferensi output yang dihasilkan berupa bilangan fuzzy maka harus ditentukan suatu nilai crisp tertentu sebagai output.

Metode Mamdani sering dikenal dengan nama Metode Min–Max. Metode ini diperkenalkan oleh Ebrahim Mamdani pada tahun 1975. Untuk mendapatkan output, diperlukan 4 tahapan yaitu pembentukan himpunan fuzzy, setelah itu dilakukan aplikasi fungsi implikasi, yang ketiga adalah proses komposisi aturan (*rules*), dan diakhiri dengan proses *defuzzifikasi*. Proses kerja logika fuzzy dengan metode mamdani dapat dilihat pada Gambar 2.7.



Gambar 2.7 Proses Kerja Fuzzy Logic Control Metode Mamdani

Proses diawali dengan pembentukan himpunan fuzzy. Pada metode Mamdani baik variabel input maupun variabel output dibagai menjadi satu atau lebih himpunan fuzzy. Setelah himpunan fuzzy terbentuk maka dilakukan aplikasi fungsi implikasi. Pada metode Mamdani, fungsi implikasi yang digunakan adalah *min*.

Pada tahapan komposisi aturan tidak seperti penalaran monoton, apabila sistem terdiri dari beberapa aturan, maka inferensi diperoleh dari kumpulan dan korelasi antar aturan. Ada 3 metode yang digunakan dalam melakukan inferensi sistem fuzzy yaitu : Max, Additive dan Probabilistik OR.

Pada metode max solusi himpunan fuzzy diperoleh dengan cara mengambil nilai maksimum aturan, kemudian menggunakannya untuk memodifikasi daerah fuzzy dan mengaplikasikan ke output dengan menggunakan operator OR(union). Jika semua proposisi telah dievaluasi, maka output akan beisi suatu himpunan fuzzy yang merefleksikan konstribusi dari tiap-tiap proposisi. Pada metode sum, solusi himpunan fuzzy diperoleh dengan cara melakukan bounded-sum terhadap semua output daerah fuzzy.

Tahapan yang terakhir merupakan defuzzifikasi. Input dari proses Defuzzifikasi adalah suatu himpunan fuzzy yang diperoleh dari komposisi aturan-aturan fuzzy, sedangkan output yang dihasilkan merupakan suatu bilangan pada domain himpunan fuzzy tersebut. Sehingga jika diberikan suatu himpunan fuzzy dalam range tertentu, maka harus dapat diambil suatu nilai crisp tertentu sebagai output. Terdapat beberapa metode yang dipakai dalam proses defuzzifikasi.

Metode pertama adalah metode centroid di mana pada metode ini penetapan nilai crisp dengan cara mengambil titik pusat daerah fuzzy.

Sedangkan metode kedua adalah metode bisektor. Pada metode ini, solusi crisp diperoleh dengan cara mengambil nilai pada domain fuzzy yang memiliki nilai keanggotaan seperti dari jumlah total nilai keanggotaan pada daerah fuzzy.

Metode berikutnya adalah metode *means of maximum* (MOM) di mana pada metode ini, solusi crisp diperoleh dengan cara mengambil nilai rata-rata domain yang memiliki niali keanggotaan maksimum. Sedangkan pada metode *Largest of Maximum* (LOM), solusi crisp diperoleh dengan cara mengambil nilai terbesar dari domain yang memiliki niali keanggotaan maksimum. Yang terakhir adalah metode *Smallest of Maximum* (SOM). Di mana pada metode ini, solusi crisp diperoleh

dengan cara mengambil nilai terkecil dari domain yang memiliki nilai keanggotaan maksimum.

2.5.3 Metode Sugeno

Metode Sugeno memiliki kemiripan dengan metode Mamdani, hanya output (konsekuen) tidak berupa himpunan fuzzy, melainkan berupa konstanta atau persamaan liniar. Ada dua model metode Sugeno yaitu model fuzzy sugeno orde nol dan model fuzzy sugeno orde satu.

BAB 3
PERANCANGAN SISTEM GUIDANCE DAN OBSTACLE AVOIDANCE

Bab ini menjelaskan mengenai perancangan system yang terbagi dalam beberapa tahapan. Tahap pertama adalah perancangan model USV. Tahap kedua adalah perancangan fuzzy logic control untuk menghindari halangan. Tahapan ketiga adalah perencanaan sistem estimasi tabrakan yang mengacu pada tiga lokasi sebelumnya.

3.1 Perancangan Persamaan Gerak USV dan Halangan

Perancangan model USV adalah sebagai berikut dengan duawaku

3.1.1 Inisiasi Parameter USV

Proses pertama perancangan persamaan gerak USv adalah dengan mengisiasi Parameter USV.

USV yang dirancang pada sistem simulasi memiliki kecepatan awal 0 m/s dan setelah itu berjalan dengan kecepatan konstan 6 m/s hingga berhenti di titik akhir. Untuk dimensi ukuran, USV memiliki ukuran 12 meter x 7 meter.

3.1.2 Definisi Posisi dan Kecepatan USV

Untuk Persamaan Gerak USV, digunakan beberapa persamaan Gerak USV. Di mana tiap persamaan menyatakan output yang berbeda:

1) Mendefinisikan perubahan posisi dan kecepatan USV saat ini (pada waktu tertentu):

$\Delta posisi = WP_i - posisi\_usv_t$
$\Delta v = v_{target}^2 - v\_usv_t^2$
$yaw\_target = tan^{-1}(\Delta posisi_y / \Delta posisi_x)$
$Jarak = |\Delta posisi|$
Dimana,
WP_i = Waypoint ke-i
$posisi\_usv_t$ = Posisi USV pada waktu t
v_{target} = Kecepatan target USV yang diinginkan
yaw_target = Sudut *heading* target USV sesuai waypoint

Jika jarak USV antara Waypoint kurang dari toleransi, maka didefinisikan $\Delta posisi$ baru dan yaw_target baru

2) Mendefinisikan vektor percepatan USV

$$percepatan = \Delta v/(2 \times jarak)$$
$$\overrightarrow{a_{usv}} = \begin{bmatrix} a_x \\ a_y \end{bmatrix} = \begin{bmatrix} percepatan \times cos\ (yaw\_target) \\ percepatan \times sin\ (yaw\_target) \end{bmatrix}$$

3) Mendefinisikan vektor kecepatan USV

$$\overrightarrow{v_{usv}} = \begin{bmatrix} v_x \\ v_y \end{bmatrix} = \begin{bmatrix} v\_usv_i \times cos\ (yaw\_target) \\ v\_usv_i \times sin\ (yaw\_target) \end{bmatrix} + \begin{bmatrix} a_x \\ a_y \end{bmatrix} \times dt$$

Dimana,
$dt =$ Sampling Time

4) Mendefinisikan posisi update USV

$$posisi_{usv_{t+dt}} = posisi_{usv_t} + \overrightarrow{v_{usv}} \times dt$$
$$\begin{bmatrix} posisix\_usv_{t+dt} \\ posisiy\_usv_{t+dt} \end{bmatrix} = \begin{bmatrix} posisix\_usv_t \\ posisiy\_usv_t \end{bmatrix} + \begin{bmatrix} v_x \\ v_y \end{bmatrix} \times dt$$

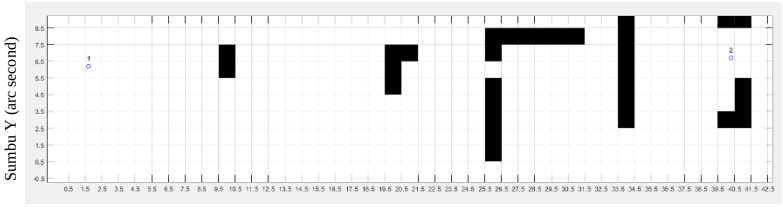
3.2 Perancangan Simulasi Lintasan USV

Pada tugas akhir ini untuk melakukan simulasi sistem dibutuhkan lintasan. Lintasan yang digunakan memiliki dimensi 2 dimensi yang tersusun dalam grid berukuran 43 x 9 di mana satu grid berukuran 1 arc second (1290 m x 270 m). Lintasan disimulasikan dengan software MATLAB dengan ketentuan halangan statis tetap pada lintasan sebagai berikut:

```
MAPS(5:7,20)=1; MAPS(7,21)=1;
MAPS(9,40:41)=1;
MAPS(3,40)=1; MAPS(3:5,41)=1;
MAPS(6:8,60:61) = 1;
MAPS(1:8,26)=1;
MAPS(8,26:31)=1;
MAPS(3:10,34)=1;
MAPS(6,26)=0;
MAPS(6:7,10)=1;
```

Dalam lintasan tersusun atas komponen untuk pengujian sistem *obstacle avoidance* di antaranya titik waypoint, halangan tetap, halangan statis, dan halangan dinamis yang telah disimulasikan sebelumnya. USV digambarkan dengan kapal merah, halangan statis digambarkan dengan kotak hitam dan merah, dan halangan dinamis digambarkan dengan kapal merah yang dilengkapi dengan penomoran pada halangan seperti yang terlihat pada gambar.

Halangan tetap merupakan halangan yang datanya telah dimasukkan ke Lintasan USV untuk dihindari (USV sudah akan merencanakan titik penghindaran sebelum melakukan deteksi pada halangan tetap). Halangan tetap di sini merupakan representasi dari halangan-halangan tetap yang ada di laut (seperti batu karang, pulau-pulau kecil, dan batu laut).



Gambar 3.1 Simulasi Peta Lintasan USV

Halangan statis pada lintasan telah ditempatkan pada beberapa titik-titik di antara waypoint 1 dan 2. Halangan statis digunakan untuk menguji keandalan sistem bahwa sistem bisa menghindari halangan baik statis maupun dinamis. Gambar lintasan yang telah dilengkapi dengan halangan statis dapat dilihat pada Gambar 3.2.

3.3 Perancangan Pemanduan Waypoint

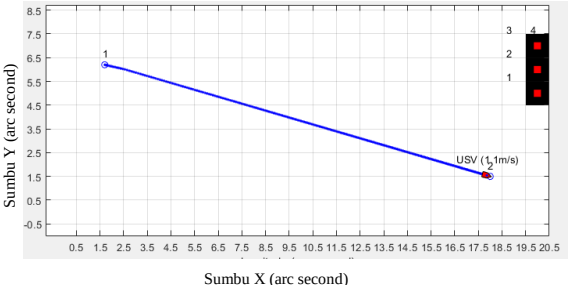
Pada simulasi di tugas akhir ini terdapat dua waypoint yang digunakan dalam tiap lintasan. Dalam proses pengujian USV bergerak dari waypoint 1 ke waypoint 2 dengan ketentuan waypoint tiap lintasan tertulis di Tabel 3.1.

Pada pengujian akan digunakan 2 lintasan dengan sepasang titik waypoint yang berbeda. Diamana lintasan lurus menggunakan 2 *waypoint* yang memiliki jarak berdekatan dan tidak ada halangan statis tetap di lintasan seperti yang terlihat pada Gambar 3.2.

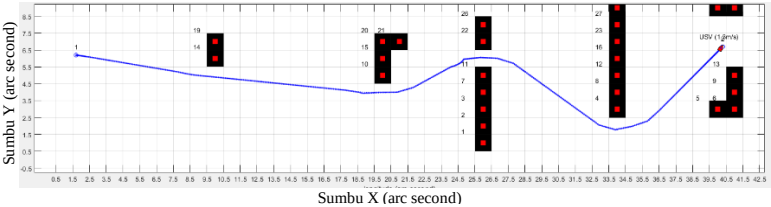
Tabel 3.1 Perancangan Titik Waypoint

Lintasan Lurus	*Waypoint* ke-	Titik Koordinat		Jarak Total
		Sumbu X	Sumbu Y	(meter)
	1	1,7	6,2	516
	2	40,2	6,602	
Lintasan Gabungan	*Waypoint* ke-	Titik Koordinat		Jarak Total
		Sumbu X	Sumbu Y	(meter)
	1	1,7	6,2	1266
	2	18	1,5	

Lintasan *waypoint* didapatkan setelah data halangan tetap diproses oleh USV dan didapatkan titik penghindaran. Setelah didapatkan titik penghindaran, maka akan muncul lintasan *waypoint* yang dapat menghindari halangan tetap tersebut. Seperti yang terlihat pada lintasan lurus di Gambar 3.2. adalah lintasan *waypoint* yang dihasilkan untuk menghubungkan kedua titik. Sedangkan lintasan gabungan memiliki halangan di tengah lintasan yang sehingga membentuk lintasan *waypoint* yang berlika-liku mengikuti halangan.



Gambar 3.2 Lintasan *Waypoint* pada Lintasan Lurus

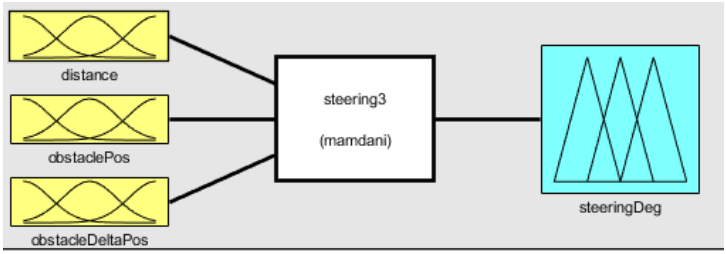


Gambar 3.3 Lintasan *Waypoint* pada Lintasan Gabungan

3.4 Perancangan *Obstacle Avoidance* berbasis Fuzzy Logic Control

Pada tugas akhir ini algoritma yang digunakan untuk *obstacle avoidance* (proses menghindari halangan) adalah algoritma *Fuzzy Logic Control*. Kapal berjalan mengikuti *waypoint* yang telah ditentukan tetapi ketika USV menemui halangan baik statis maupun dinamis akan mengaktifkan *Fuzzy Logic Control* dan peraturan yang telah ditentukan akan aktif dan mengatur berjalannya USV untuk menghindari halangan yang disimulasikan.

Langkah pertama dalam merancang *fuzzy logic control* ialah dengan menentukan input dan output yang diinginkan. Pada tugas akhir ini diambil dua input dengan satu output. Input berupa jarak halangan terhadap USV dan posisi obstacle terhadap USV sedangkan untuk outputnya adalah sudut putar USV (*steering degree*). Input dan output yang digunakan dan telah disimulasikan dalam MATLAB dapat dilihat pada Gambar 3.4.



Gambar 3.4 *Fuzzy Logic Controller*

Langkah kedua dalam menyusun algoritma fuzzy adalah menentukan masing-masing himpunan fungsi anggota fuzzy. Di mana setelah anggota terbentuk maka akan dapat dilakukan proses penyusunan aturan yang akan digunakan.

Aturan yang ada pada *Fuzzy Logic Control* ditulis dalam *Fuzzy Toolbox* yang ada pada *software* MATLAB. Aturan yang mengacu dari parameter dalam fuzzy ditentukan dalam proses *trial and error* hingga keputusan yang sesuai.

3.4.1 Himpunan Fuzzy

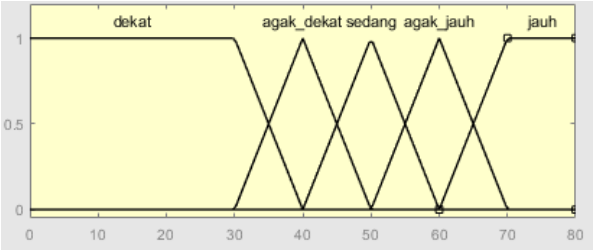
Dalam merancang *fuzzy logic control* dibutuhkan *rules* (peraturan) untuk memproses input yang ada sehingga dapat mendapatkan output yang sesuai dan diinginkan. Himpunan fuzzy disusun berdasarkan logika dari cara kerja USV.

Setiap elemen input dan output dari sistem logika fuzzy memiliki himpunan keanggotaan masing-masing. Fungsi keanggotaan dari masing-masing input dan output berbeda dan setiap komponen memiliki fungsi keanggotaan dengan parameter yang sesuai.

Pada input yang berupa jarak halangan terhadap USV terdapat 5 fungsi keanggotaan. Fungsi keanggotaan tersebut menyatakan tingkatan jarak antara halangan terhadap USV. Fungsi keanggotaan dari jarak halangan USV antara lain adalah dekat, agak dekat, sedang, agak jauh, dan jauh. Parameter dari fungsi keanggotaan ditampilkan dalamt Tabel 3.2. Fungsi keanggotaan dinyatakan dalam bentuk kurva trapesium dan segitiga seperti yang terlihat pada Gambar 3.5.

Tabel 3.2 Himpunan Anggota Jarak Halangan

Fungsi Keanggotaan	Nilai Terkecil (meter)	Nilai Terbesar (meter)	Representasi
Dekat	0	40	Trapesium
Agak dekat	30	50	Segitiga
Sedang	40	60	Segitiga
Agak Jauh	50	70	Segitiga
Jauh	60	80	Trapesium

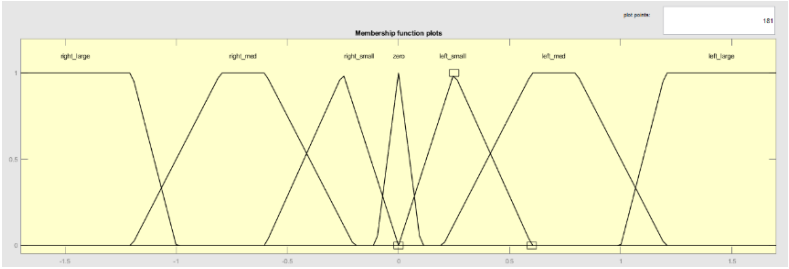


Gambar 3.5 Representasi Anggota Jarak Halangan

Pada input yang berupa letak posisi halangan terhadap USV terdapat 7 fungsi keanggotaan. Fungsi keanggotaan tersebut menyatakan posisi halangan terhadap USV. Fungsi keanggotaan dari jarak halangan USV antara lain *right large, right medium, right small, zero, left small, left medium, left large*. Parameter dari fungsi keanggotaan ditampilkan dalam Tabel 3.3. Fungsi keanggotaan dinyatakan dalam bentuk kurva trapesium dan segitiga seperti yang terlihat pada Gambar 3.6.

Tabel 3.3 Himpunan Anggota Posisi Halangan

Fungsi Keanggotaan	Nilai Terkecil (radian)	Nilai Terbesar (radian)	Representasi
Right large	-1,7	-1	Trapesium
Right medium	-1,2	-0,2	Trapesium
Right small	-0,6	0	Segitiga
Zero	-0,1	0,1	Segitiga
Left small	0	0.6	Segitiga
Left medium	0,2	1,2	Trapesium
Left large	1	1,7	Trapesium

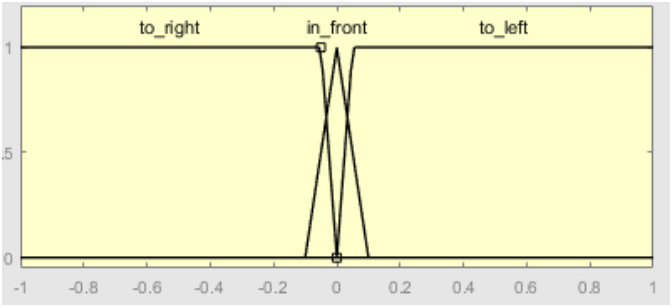


Gambar 3.6 Representasi Anggota Posisi Halangan

Pada input yang berupa perubahan sudut halangan terhadap USV terdapat 3 fungsi keanggotaan. Fungsi keanggotaan tersebut menyatakan perubahan posisi halangan berdasarkan perubahan sudut halangan terhadap USV. Di mana input ini memiliki fungsi keanggotaan *to right*, *in front*, dan *to left*. Parameter dari fungsi keanggotaan ditampilkan dalam Tabel 3.4. dan kurva fungsi keanggotaan dipreresentasikan dalam kurva segitiga dan trapesium seperti yang terlihat pada Gambar 3.7.

Tabel 3.4 Himpunan Anggota Perpindahan Posisi Halangan

Fungsi Keanggotaan	Nilai Terkecil (radian)	Posisi	Nilai Terbesar (radian)	Posisi	Representasi
To right	-1	57,2958 di kanan	0	0	Trapesium
In front	-0.1	5,279 di kanan	0.1	5,279 di kiri	Segitiga
To left	0	0	1	57,2958 di kiri	Trapesium

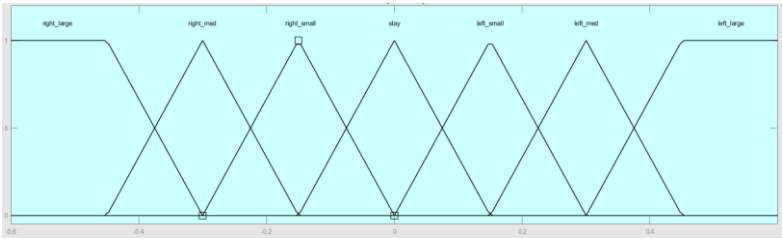


Gambar 3.7 Representasi Anggota Perubahan Posisi Halangan

Pada output yang berupa *steering degree* (sudut belok) USV terdapat 7 fungsi keanggotaan. Fungsi keanggotaan tersebut menyatakan posisi halangan terhadap USV. Fungsi keanggotaan dari *steering degree* (sudut belok) USV antara lain *right large, right medium, right small, stay, left small, left medium, left large.* Parameter dari fungsi keanggotaan ditampilkan dalam Tabel 3.4. Fungsi keanggotaan dinyatakan dalam bentuk kurva trapesium dan segitiga seperti yang terlihat pada Gambar 3.8.

Tabel 3.5 Himpunan Anggota Sudut Belok

Fungsi Keanggotaan	Nilai Terkecil (radian)	Nilai Terbesar (radian)	Representasi
Right large	-0.6	-0.3	Trapesium
Right medium	-0.45	-0.15	Trapesium
Right small	-0.3	0	Segitiga
Stay	-0.15	0.15	Segitiga
Left small	0	0.3	Segitiga
Left medium	0.15	0.45	Trapesium
Left large	0.3	0.6	Trapesium



Gambar 3.8 Representasi Anggota Sudut Belok

3.4.2 Penentuan Fuzzy Rules Base

Proses pengambilan keputusan yang berdasarkan aturan-aturan yang ditentukan pada *rules base* (basis aturan) untuk menghubungkan antar peubah-peubah input fuzzy dan output fuzzy. Aturan-aturan ini dinyatakan dalam bentuk fungsi IF-THEN

Pada tahap ini, hasil dari fuzzifikasi pada setiap aturan akan dilihat kembali. Dalam penulisan *rules* di tugas akhir ini digunakan logika AND yang berarti ketika kedua kondisi sama-sama memenuhi. Terdapat 35 *rules* yang digunakan pada tugas akhir ini dan tertulis di Tabel 3.6 untuk kondisi input perpindahan posisi In_front, Tabel 3.7 untuk perpindahan posisi To_right, dan Tabel 3.7 untuk perpindahan posisi To_left. Dimana tiap 105 *rules* mewakili kemungkinan keadaan yang dihadapi oleh USV terhadap jarak halangan dan posisi halangan lalu akan menghasilkan output berupa sudut belok dari USV.

Tabel 3.6 Fuzzy Rules dengan Kondisi Perpindahan Posisi In_front

In_front	Dekat	Agak dekat	Sedang	Agak jauh	Jauh
Right large	Left medium (7)	Left small (14)	Stay (21)	Stay (28)	Stay (35)
Right medium	Left medium (6)	Left medium (13)	Left small (20)	Stay (27)	Stay (34)
Right small	Left large (5)	Left medium (12)	Left small(19)	Left small (26)	Right small (33)
Zero	Right large(4)	Right medium (11)	Right small (18)	Right small (25)	Right small (32)
Left small	Right large (3)	Right medium (10)	Right small (17)	Right small (24)	Left small (31)
Left medium	Right medium (2)	Right medium (9)	Right small (16)	Stay (23)	Stay (30)
Left large	Right medium (1)	Right small (8)	Stay (15)	Stay (22)	Stay (29)

Tabel 3.7 Fuzzy Rules dengan Kondisi Perpindahan Posisi To_right

To_right	Dekat	Agak dekat	Sedang	Agak jauh	Jauh
Right large	Left small (36)	Stay (43)	Stay (50)	Stay (57)	Stay (64)
Right medium	Left small (37)	Left small (44)l	Stay (51)	Stay (58)	Stay (65)
Right small	Left large (38)	Left small (45)	Left small (52)	Stay (59)	Stay (66)
Zero	Left medium (39)	Left medium (46)	Left small (53)	Left small (60)	Stay (67)
Left small	Left medium (40)	Left medium (47)	Left medium (54)	Left small (61)	Left small (68)
Left medium	Left medium (41)	Left large (48)	Left medium (55)	Left medium (62)	Left small (69)
Left large	Left large (42)	Left large (49)	Left medium (56)	Left medium (63)	Left medium (70)

Tabel 3.8 Fuzzy Rules dengan Kondisi Perpindahan Posisi To_left

To_left	Dekat	Agak dekat	Sedang	Agak jauh	Jauh
Right large	Right large (71)	Right large (78)	Right medium (85)	Right medium (92)	Right medium (99)
Right medium	Right large (72)	Right medium (79)	Right medium (86)	Right medium (93)	Right small (100)
Right small	Right medium (73)	Right medium (80)	Right medium (87)	Right small (94)	Right small (101)
Zero	Right medium (74)	Right medium (81)	Right small (88)	Right small (95)	Stay (102)
Left small	Right medium (75)	Right small (82)	Right small (89)	Stay (96)	Stay (103)
Left medium	Right small (76)	Right small (83)	Stay (90)	Stay (97)	Stay (104)
Left large	Right small (77)	Stay (84)	Stay (91)	Stay (98)	Stay (105)

3.5 Perancangan Sistem Estimasi Tabrakan

Pada dasarnya USV hanya dapat mendeteksi jarak dan sudut antara USV dengan halangan menggunakan sistem deteksi halangan, karenanya diperlukan algoritma untuk menentukan apakah USV perlu menghindari halangan yang terdeteksi. Metode menghindari halangan dinamis ini terbagi menjadi beberapa bagian utama.

Sistem deteksi halangan mampu mendeteksi adanya halangan statis dan dinamis yang dihadapi oleh USV pada simulasi MATLAB. Sistem deteksi halangan dilengkapi dengan kemampuan sistem deteksi tabrakan. Data dari sistem deteksi halangan akan diolah sehingga memiliki keluaran berupa lokasi tabrakan (*collision*) apabila dilakukan proses estimasi akan terjadi tabrakan. Proses estimasi tabrakan ini menggunakan konsep vektor dan persamaan garis lurus dengan asumsi halangan dinamis bergerak dengan vektor kecepatan konstan (nilai dan arah kecepatan konstan. menampilkan metode deteksi halangan dan prediksi tabrakan.

Dalam sistem estimasi tabrakan, besar jarak terhadap waktu berperan sangat krusial. Proses estimasi halangan dilakukan dengan mencari titik minimum dari fungsi jarak halangan ke USV terhadap waktu. Fungsi tersebut didekati dengan persamaan kuadratik.

$$\begin{bmatrix} 1 & 2kT & 4kT^2 \\ 1 & kT & kT^2 \\ 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix} = \begin{bmatrix} D(k) \\ D(k-1) \\ D(k-2) \end{bmatrix}$$

Dimana k menunjukkan sampling time ke k, T menunjukkan waktu sampling, serta a_1, a_2, a_3 menunjukkan persamaan elemen dari persamaan kuadrat jarak terhadap waktu $a_1 + a_2 t + a_3 t^2$.

Waktu ke titik tabrakan diprediksi berdasarkan fungsi kuadratik dari jarak terhadap waktu. Apabila $a_3 = 0$ maka persamaan linier, sehingga dicari waktu ke titik tabrakan $t_{collision} = \frac{d_{safe} - a_1}{a_2}$. Bila nilai dari waktu ke titik tabrakan lebih dari 0 ($t_{collision} > 0$) maka dapat dipastikan kapal akan menabrak halangan.

Apabila $a_2 = 0$ maka persamaan kuadratik dengan titik minimum ada di titik saat ini sehingga kapal dalam posisi terdekat terhadap obstacle dan dianggap berpeluang bertabrakan bila nilai jarak antara halangan dengan USV lebih kecil dari nilai jarak minimum yang telah ditentukan ($D < D_{min}$). Di mana pada tugas akhir ini telah ditentukan nilai jarak aman sebesar 60 meter. Maka didapatkan waktu ke titik tabrakan seperti pada persamaan.

$t_{collision} = sqrt\left(\frac{d_{safe}-a_1}{a_3}\right)$.

Selain kedua kemungkinan diatas, waktu ke titik tabrakan dapat dicari dengan mencari akar-akar terkecil non-negatif dari persamaan kuadratik jarak terhadap waktu, serta jarak minimum dapat dicari dengan mensubstitusikan waktu hasil prediksi ke persamaan kuadratik jarak terhadap waktu.

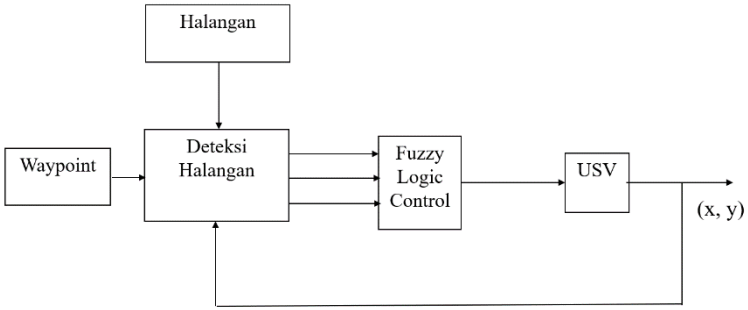
3.6 Perancangan Sistem Pemanduan dengan *Obstacle Avoidance*

Sistem pemanduan pada tugas akhir ini mengacu pada pemanduan *waypoint* di mana USV akan bergerak dari titik *waypoint* 1 ke titik *waypoint* 2 pada lintasan yang telah ditentukan pada simulasi MATLAB. USV disimulasikan dengan kecepatan konstan 6 m/s dan bergerak mengikuti lintasan dari *waypoint*.

USV dilengkapi dengan kemampuan deteksi halangan yang mampu mendeteksi apakah di lintasan yang dilalui terdapat halangan baik statis maupun dinamis. Halangan statis adalah halangan yang memiliki posisi diam dan tidak bergerak sedangkan halangan dinamis adalah halangan yang berpindah posisi sesuai lintasan yang ditentukan seiring dengan perubahan waktu.

Ketika USV berjalan pada lintasan dan sistem deteksi halangan mendeteksi bahwa terdapat halangan di depan USV maka akan dilakukan proses *obstacle avoidance* yaitu proses menghindari halangan. Algoritma yang digunakan untuk menghindari halangan adalah *Fuzzy Logic Control.* Ketika sistem deteksi halangan mendeteksi bahwa terdapat halangan di lintasan USV maka sistem estimasi tabrakan akan aktif dan memetakan apakah berpeluang terjadi tabrakan atau tidak. Jika proses estimasi tabrakan menyatakan bahwa ada peluang terjadi tabrakan (jika nilai jarak minimum halangan ke kapal apabila tidak dihindari lebih kecil dari nilai jarak aman yang ditentukan) maka sisten *Fuzzy Logic Control* akan aktif dan mengambil data jarak dan posisi halangan dari USV yang ada pada sistem deteksi halangan. Ketika sistem mampu mendefinisikan jarak halangan dari USV serta posisi objek dari USV maka kedua parameter tersebut akan diproses oleh *Fuzzy Logic Control* lalu akan dihasilkan output berupa sudut belok USV. Ketika sudut belok terlah terdefinisi maka USV akan membelok mengikuti sudut belok tersebut dan setelah belok USV akan kembali ke lintasan. Jika setelah USV bergerak membelok mengikuti sudut belok yang telah ditentukan dan sistem estimasi tabrakan mendapati terdapat kemungkinan tabrakan kembali, maka *Fuzzy Logic Control* akan bekerja kembali untuk menentukan sudut

belok dari USV. Ketika USV telah kembali ke lintasan dan mendapati ada halangan di lintasan kembali maka akan dilakukan proses *obstacle avoidance* sama seperti sebelumnya hingga USV berhenti ke titik akhir dari lintasan yaitu titik *waypoint* 2. Keseluruhan proses *obstacle avoidance* dari tugas akhir ini dapat dinyatakan dalam diagram blok pada Gambar 3.0.



Gambar 3.9 Diagram Blok Prinsip Kerja Keseluruhan Sistem

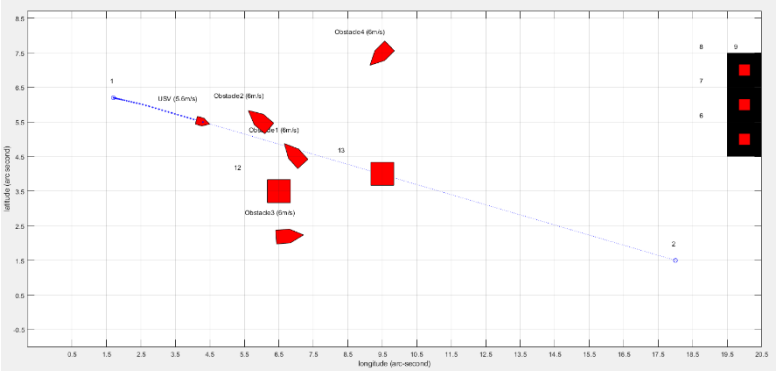
BAB 4
PENGUJIAN SISTEM

Bab ini membahas mengenai hasil dan analisis dari data yang telag diperoleh setelah dilaksanakan proses simulasi sistem *obstacle avoidance* pada aplikasi MATLAB. Terdapat beberapa tahap pengujian yang dilaksanakan.

Pengujian dilakukan pada dua lintasan yang memiliki kondisi halangan yang berbeda-beda di tiap lintasannya. Selain itu juga dilakukan variasi pengujian berdasarkan jarak aman yang telah ditentukan. Jarak aman adalah nilai yang ditentukan pada sistem untuk mengaktifkan kontroler *Fuzzy Logic Control* untuk melakukan penghindaran halangan.

4.1 Pengujian pada Lintasan Lurus

Pada pengujian di lintasan lurus dilakukan pengujian pada USV untuk berjalan di lintasan lurus dengan kondisi beberapa halangan terdapat di lintasan. Halangan yang disimulasikan berupa halangan statis dan dinamis yang telah dirancang untuk berada di tengah-tengah lintasan USV. Lintasan dapat dilihat pada Gambar 4.1.



Gambar 4.1 Lintasan Lurus

Terdapat 2 halangan statis dan 4 halangan dinamis yang disimulasikan pada pengujian di lintasan lurus. Halangan dirancang untuk bergerak dalam posisi yang berdekatan untuk menguji apakah sistem mampu menghindari beberapa halangan dalam radius jarak yang berdekatan atau tidak. Halangan tersebut memiliki ketentuan seperti yang tercantum pada Tabel 4.1.

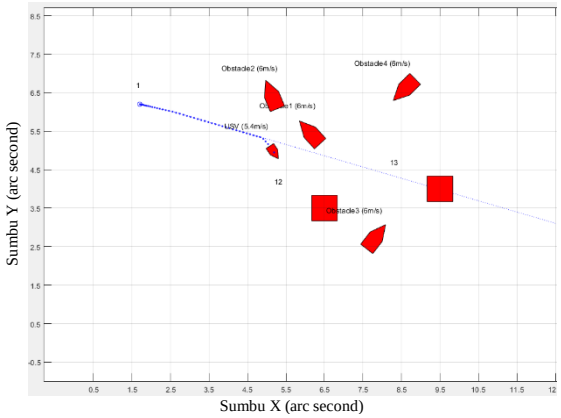
Tabel 4.1 Detail Halangan pada Lintasan Lurus

Jenis Halangan	Nomor Halangan	Posisi Awal		Posisi Akhir	
		Sumbu X	Sumbu Y	Sumbu X	Sumbu Y
Dinamis	1	6	2	11	11
Dinamis	2	5	3	10	10
Dinamis	3	4	4	9	9
Dinamis	4	8,5	10	12,5	0,5
Jenis Halangan	Nomor Halangan	Ukuran Halangan (meter)		Posisi Halangan	
		Panjang	Lebar	Sumbu X	Sumbu Y
Statis	12	20	20	6,5	3,5
Statis	13	20	20	9	4

Pada pengujian ini akan dilakukan percobaan sebanyak tiga kali. Percobaan dilakukan pada kondisi jarak aman yang berbeda-beda pada setiap percobaannya.

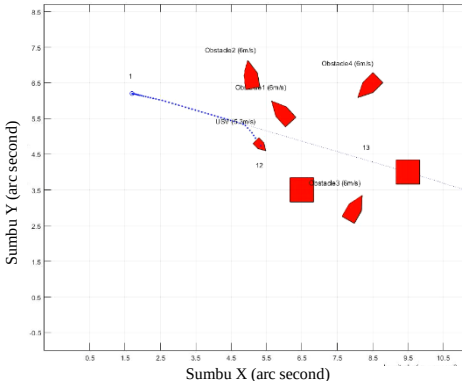
4.1.1 Pengujian dengan Jarak Aman 20 Meter

Pada simulasi pengujian ini, dirancang sistem dengan jarak aman 20 meter untuk mengaktifkan *Fuzzy Logic Control*. Proses penghindaran diawali dengan menghindari sepasang halangan dinamis 1 dan 2 seperti yang terlihat pada Gambar 4.2. *Output fuzzy rules* yang aktif pada proses penghindaran kali ini adalah "*right_small*".



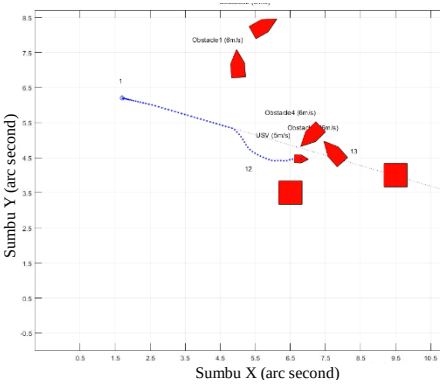
Gambar 4.2 Proses Penghindaran Pertama pada Lintasan Lurus

Setelah berhasil melewati dua halangan pertama, USV kembali berjalan dan melakukan proses penghindaran untuk menghindari halangan statis nomor 12 dengan mengaktifkan *output fuzzy rules* "*left_small"* untuk belok ke kiri seperti yang terlihat di Gambar 4.3.



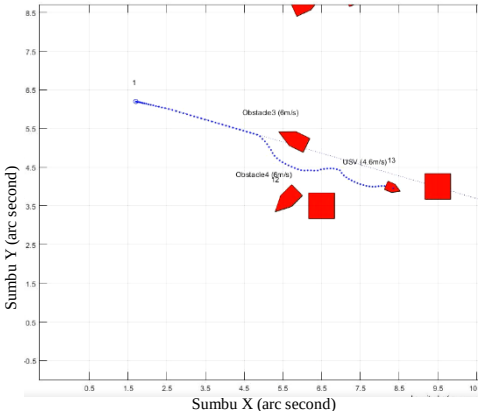
Gambar 4.3 Proses Penghindaran Kedua

Setelah berhasil melakukan penghindaran kedua, USV kembali melakukan penghindaran untuk menghindari halangan dinamis 3 dan 4. *Output fuzzy rules* yang aktif adalah "*right_small"*, di mana USV melakukan proses penghindaran dengan belok ke kanan seperti proses yang terlihat di Gambar 4.4.



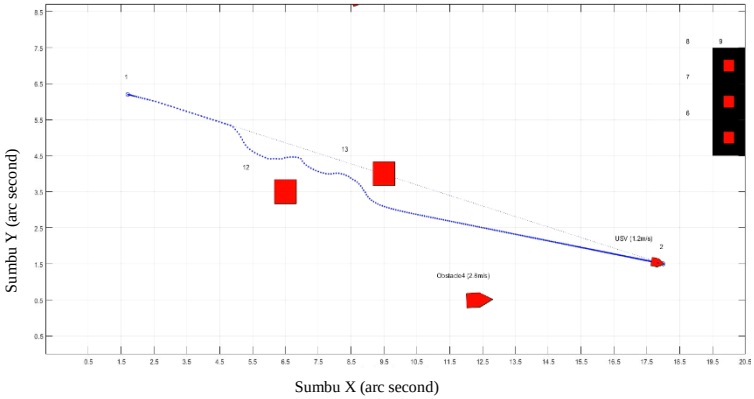
Gambar 4.4 Proses Penghindaran Ketiga pada Lintasan Lurus

Penghindaran keempat dilakukan untuk menghindari halangan statis 13. Penghindaran dilakukan dengan mengaktifkan *output fuzzy rules "right_small"* untuk belok ke kanan. Proses penghindaran dapat dilihat pada Gambar 4.5.



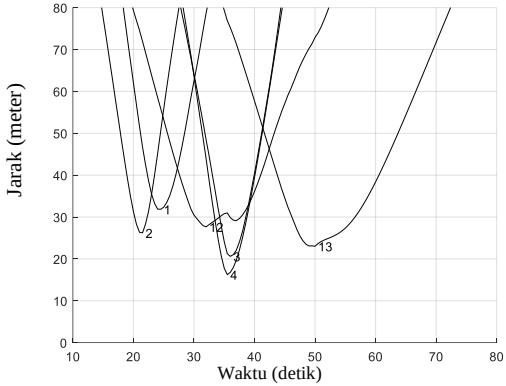
Gambar 4.5 Proses Penghindaran Keempat pada Lintasan Lurus

Setelah melakukan penghindaran, USV kembali ke lintasan dan berusaha untuk menyelesaikan perjalanan dengan menuju titik akhir yaitu titik *waypoint 2.* USV mampu menyelesaikan perjalanan hingga titik akhir seperti yang terlihat pada Gambar 4.6.



Gambar 4.6 USV Berhasil Menyelesaikan Lintasan Lurus

Dari data perjalanan USV yang ditampilkan pada Gambar 4.7, didapatkan jarak paling dekat terdeteksi adalah sebesar 16,24 pada halangan dinamis nomor 4 dan semua halangan berhasil dihindari sehingga tidak terjadi tabrakan. Dari kurva yang ditampilkan dapat dibuat hasil akhir pengujian seperti pada Tabel 4.2. Dalam pengujian ini USV menempuh perjalanan 516 meter.



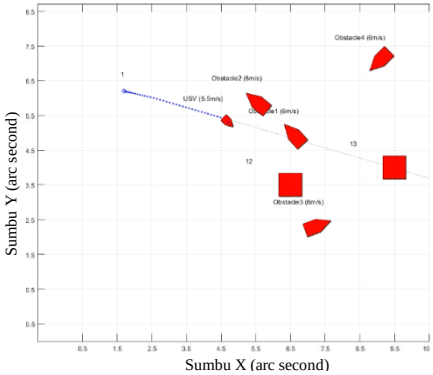
Gambar 4.7 Kurva Deteksi Halangan pada Lintasan Lurus

Tabel 4.2 Output Akhir Pengujian pada Lintasan Lurus

Jenis Halangan	Nomor Halangan	Deteksi Jarak Terdekat (meter)	Status Penghindaran	Fuzzy Rules yang aktif
Dinamis	1	31,85	Mampu dihindari	Right_small (100)
Dinamis	2	26,21	Mampu dihindari	
Dinamis	3	20,61	Mampu dihindari	Right small (77)
Dinamis	4	16,24	Mampu dihindari	
Statis	12	27,66	Mampu dihindari	Left small (53)
Statis	13	24,23	Mampu dihindari	Right small (77)

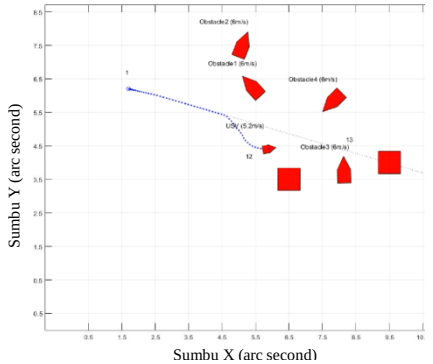
4.1.2 Pengujian dengan Jarak Aman 40 Meter

Pada simulasi pengujian ini, dirancang sistem dengan jarak aman 40 meter untuk mengaktifkan *Fuzzy Logic Control*. Proses penghindaran diawali dengan menghindari sepasang halangan dinamis 1 dan 2 seperti yang terlihat pada Gambar 4.8. *Output fuzzy rules* yang aktif pada proses penghindaran ini adalah "*right_small*".



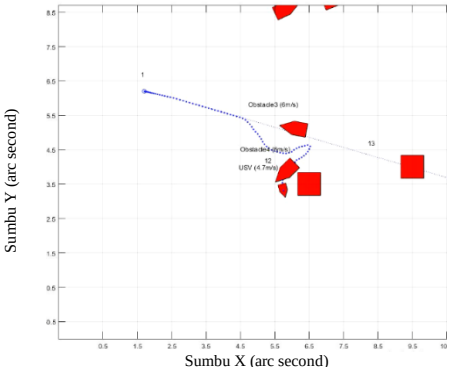
Gambar 4.8 Proses Pertama pada Lintasan Lurus

Setelah berhasil melewati dua halangan pertama, USV kembali berjalan dan melakukan proses penghindaran untuk menghindari halangan statis nomor 12 dengan mengaktifkan *output fuzzy rules* "*left_small"* untuk belok ke kiri seperti yang terlihat di Gambar 4.9.



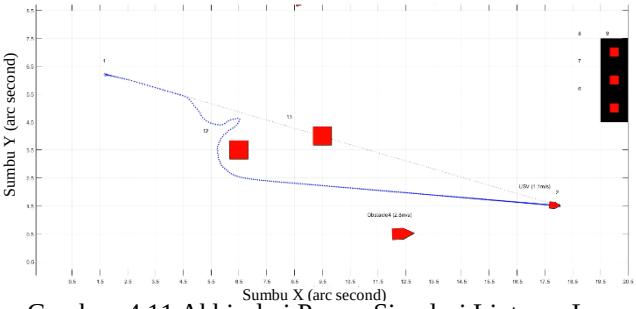
Gambar 4.9. Proses Penghindaran Kedua pada Lintasan Lurus

Setelah USV melakukan penghindaran kedua, USV menuju kembali ke lintasan. Saat menuju kembali ke lintasan, sistem mendeteksi adanya halangan dinamis 3 dan 4 sehingga USV melakukan penghindaran dengan mengaktifkan *output fuzzy rules* "right_small". Setelah menghindari kedua halangan dinamis tersebut, USV menuju ke arah kanan dan kembali mendeteksi adanya halangan statis 12. USV melakukan penghindaran ke arah kanan dengan mengaktifkan *output fuzzy rules* "right_small" dan berhasil menghindari halangan statis 12 seperti yang terlihat di Gambar 4.10.



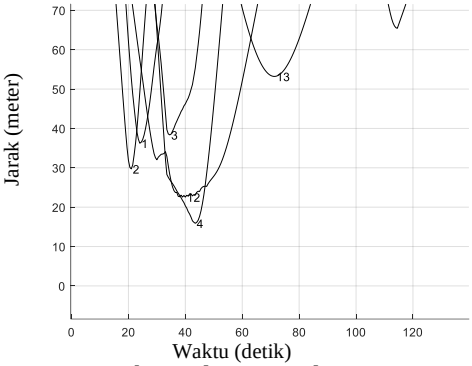
Gambar 4.10 Proses Penghindaran Ketiga dan Keempat pada Lintasan Lurus

Setelah berhasil melakukan penghindaran, USV kembali melanjutkan perjalanan untuk mencapai titik akhir *waypoint* 2. USV berhasil menyelesaikan perjalanan mencapai titik *waypoint* 2 pada Gambar 4.11.



Gambar 4.11 Akhir dari Proses Simulasi Lintasan Lurus

Setelah USV selesai melakukan perjalanan, akan didapatkan data deteksi halangan yang terdeteksi selama pengujian. Data tersebut dinyatakan dalam kurva yang terdapat di Gambar 4.12. Dari kurva tersebut diketahui bahwa jarak halangan terdekat yang terdeteksi oleh sistem sebesar 15,91 meter pada halangan hinamis 4. Dari kurva tersebut diketahui bahwa USV mampu menghindari semua halangan yang ada di lintasan. Dari kurva dapat diambil hasil akhir yang terlihat di Tabel 4.3. Dalam pengujian ini USV menempuh perjalanan 583 meter.



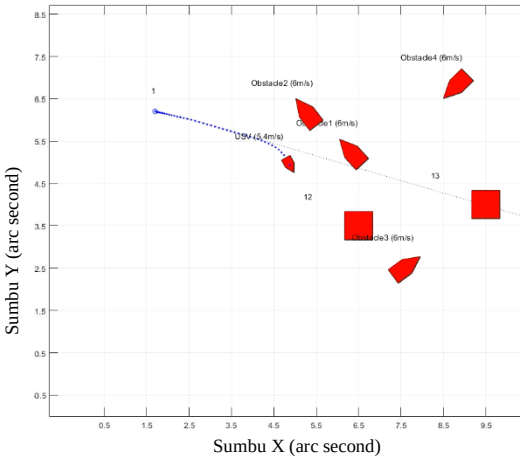
Gambar 4.12 Kurva Deteksi Halangan pada Lintasan Lurus

Tabel 4.3 Hasil Akhir Simulasi pada Lintasan Lurus

Jenis Halangan	Nomor Halangan	Deteksi Jarak Terdekat (meter)	Status Penghindaran	Fuzzy Rules yang aktif
Dinamis	1	36,24	Mampu dihindari	Right_small (32)
Dinamis	2	29,72	Mampu dihindari	
Dinamis	3	38,36	Mampu dihindari	Right small (83)
Dinamis	4	15,91	Mampu dihindari	
Statis	12	22,56	Mampu dihindari	Left_small dan right_small (53 dan 83)
Statis	13	53,22	Mampu dihindari	-

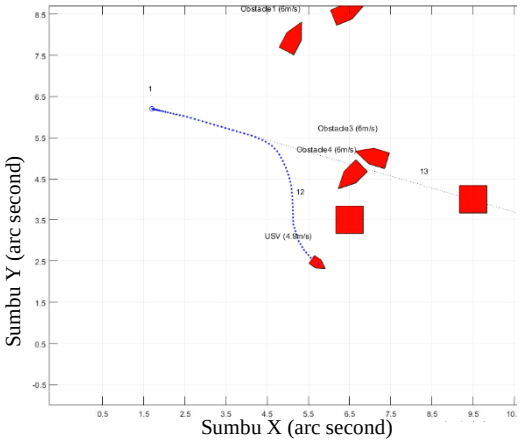
4.1.3 Pengujian dengan Jarak Aman 60 Meter.

Pada simulasi pengujian ini, dirancang sistem dengan jarak aman 60 meter untuk mengaktifkan *Fuzzy Logic Control*. Proses penghindaran diawali dengan menghindari sepasang halangan dinamis 1 dan 2 seperti yang terlihat pada Gambar 4.13. *Output fuzzy rules* yang aktif sebanyak dua kali pada proses penghindaran ini adalah "*right_small*".



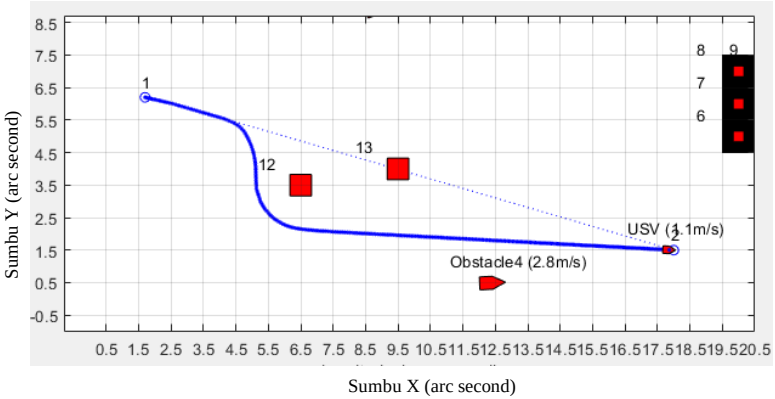
Gambar 4.13 Penghindaran Pertama dan Kedua pada Lintasan Lurus

Setelah mampu menghindari kedua halangan pertama, USV kembali berjalan melanjutkan perjalanan hingga sistem mendeteksi adanya halangan statis 12. USV melakukan penghindaran sebanyak dua kali dengan mengaktifkan dua *output fuzzy rules* yaitu penghindaran pertama dengan "right_small" dan penghindaran kedua dengan "stay". Pada Gambar 4.14 dapat dilihat bahwa USV melakukan mengabaikan halangan dinamis 3 dan 4 karena jarak estimasi yang terdeteksi terlalu jauh sehingga sistem tidak mengaktifkan *Fuzzy Logic Control*.



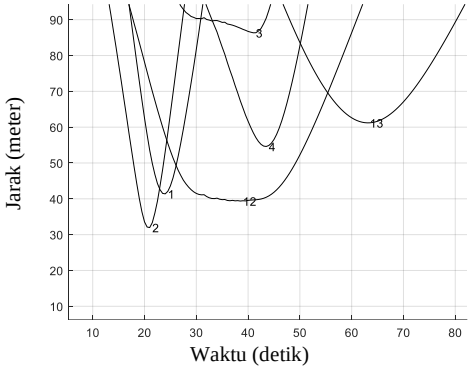
Gambar 4.14 Penghindaran Ketiga pada Lintasan Lurus

Setelah menghindari halangan statis 12, USV berjalan kembali ke lintasan. Dalam perjalanan kembali USV menghindari halangan statis 13 dengan tetap berada di luar lintasan dengan mengaktifkan *output fuzzy rules* "*stay*". Setelah menghindar dari halangan statis 13, USV kembali ke titik akhir yaitu titik *waypoint* 2 seperti hasil akhir lintasan yang terlihat di Gambar 4.15.



Gambar 4.15 Hasil Akhir Lintasan Lurus dengan Jarak Aman 60 Meter

Setelah USV selesai melakukan perjalanan, akan didapatkan data deteksi halangan yang terdeteksi selama pengujian. Data tersebut dinyatakan dalam kurva yang terdapat di Gambar 4.15. Dari kurva tersebut diketahui bahwa jarak halangan terdekat yang terdeteksi oleh sistem sebesar 32,13 meter pada halangan hinamis 2. Dari kurva tersebut diketahui bahwa USV mampu menghindari semua halangan yang ada di lintasan. Dari kurva dapat diambil hasil akhir yang terlihat di Tabel 4.4.



Gambar 4.16 Kurva Deteksi Halangan pada Lintasan Lurus

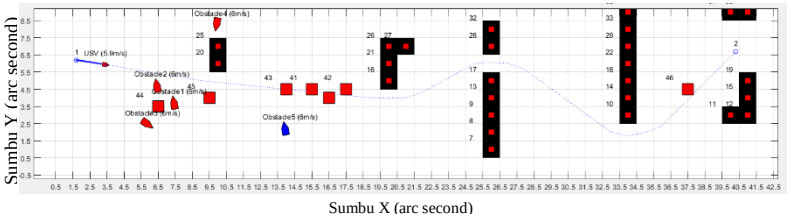
Tabel 4.4 Hasil Akhir Lintasan Lurus dengan Jarak Aman 60 Meter

Jenis Halangan	Nomor Halangan	Deteksi Jarak Terdekat (meter)	Status Penghindaran	Fuzzy Rules yang aktif
Dinamis	1	41,39	Mampu dihindari	Right_small (32)
Dinamis	2	32,13	Mampu dihindari	
Dinamis	3	86,76	Mampu dihindari	Stay (91)
Dinamis	4	54,73	Mampu dihindari	
Statis	12	39,49	Mampu dihindari	Right_small dan stay (90 dan 15)
Statis	13	61,21	Mampu dihindari	Stay (22)

Dari keseluruhan pengujian pada lintasan lurus, terdapat 9 *fuzzy rules* yang aktif dari 105 yang ada dengan output hanya 3 yang aktif. Output yang aktif adalah "*right_small*". "*stay*", dan "*left_small*". Dalam pengujian ini USV menempuh perjalanan 562 meter.

4.2 Pengujian pada Lintasan Gabungan

Pada pengujian di lintasan lurus dilakukan pengujian pada USV untuk berjalan di lintasan lurus dengan kondisi beberapa halangan terdapat di lintasan. Halangan yang disimulasikan berupa halangan statis dan dinamis yang telah dirancang untuk berada di tengah-tengah lintasan USV. Lintasan dapat dilihat pada Gambar 4.17. Lintasan Gabungan dibagi menjadi tiga bagian di mana setiap bagian memiliki kombinasi halangan sendiri

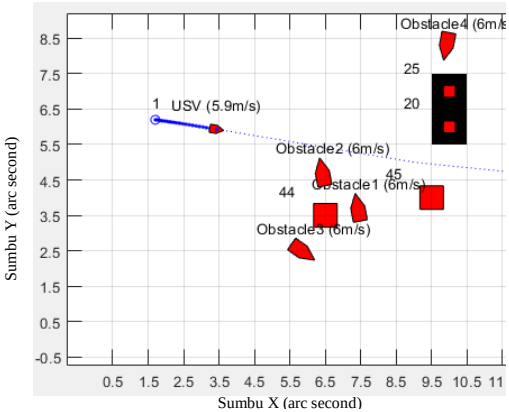


Gambar 4.17 Lintasan Gabungan

Pada Bagian 1 terdapat 5 halangan dinamis, 2 halangan statis baru, dan 1 halangan statis tetap. Rancangan kombinasi halangan disesuaikan dengan perkiraan halangan yang ada di laut. Rancangan halangan tersaji di Tabel 4.5 dan lintasan bagian 1 dapat dilihat di Gambar 4.18.

Tabel 4.5 Rancangan Halangan Bagian 1 Lintasan Gabungan

Jenis Halangan	Nomor Halangan	Posisi Awal		Posisi Akhir	
		Sumbu X	Sumbu Y	Sumbu X	Sumbu Y
Dinamis	1	6	2	11	11
Dinamis	2	5	3	10	10
Dinamis	3	4	4	9	9
Dinamis	4	8,5	10	12,5	0,5
Dinamis	5	12,5	0,5	8,5	10
Jenis Halangan	Nomor Halangan	Ukuran Halangan (meter)		Posisi Halangan	
		Panjang	Lebar	Sumbu X	Sumbu Y
Statis	44	20	20	6,5	3,5
Statis	45	20	20	9	4
Statis Tetap	20	30	60	6	10

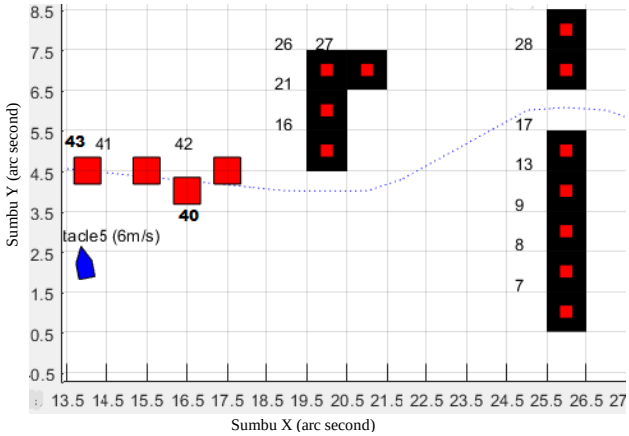


Gambar 4.18 Bagian 1 Lintasan Gabungan

Pada Bagian 2 terdapat 4 halangan statis berdekatan, 1 halangan statis tetap, dan 2 halangan statis tetap yang menjepit lintasan. Rancangan halangan tersaji di Tabel 4.6 dan lintasan bagian 1 dapat dilihat di Gambar 4.19.

Tabel 4.6 Rancangan Halangan Bagian 2 Lintasan Gabungan

Jenis Halangan	Nomor Halangan	Ukuran Halangan (meter)		Posisi Halangan	
		Panjang	Lebar	Sumbu X	Sumbu Y
Statis	40	20	20	16,5	4
Statis	41	20	20	15,5	4,5
Statis	42	20	20	17,5	4,5
Statis	43	20	20	14	4,5
Statis Tetap	16	(30 x 90) + (30 x 30)		20	6
Statis Tetap	17	30	150	25	3
Statis Tetap	28	30	60	25	7,5

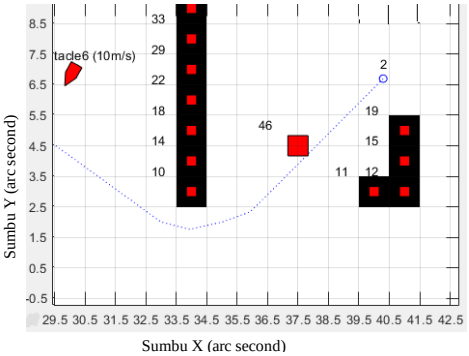


Gambar 4.19 Bagian 2 Lintasan Gabungan

Pada Bagian 3 terdapat 1 halangan dinamis yang memiliki kecepatan lebih besar dari USV, 1 halangan statis tetap, dan 2 halangan statis tetap. Rancangan halangan tersaji di Tabel 4.7 dan lintasan bagian 1 dapat dilihat di Gambar 4. 20.

Tabel 4.7 Rancangan Halangan Bagian 3 Lintasan Gabungan

Jenis Halangan	Nomor Halangan	Posisi Awal		Posisi Akhir	
		Sumbu X	Sumbu Y	Sumbu X	Sumbu Y
Dinamis	6	20	12	26,5	0,5
Jenis Halangan	Nomor Halangan	Ukuran Halangan (meter)		Posisi Halangan	
		Panjang	Lebar	Sumbu X	Sumbu Y
Statis	46	20	20	37,5	4,5
Statis Tetap	45	30	210	34	6
Statis Tetap	20	(30 x 30) + (30 x 90)		40	3

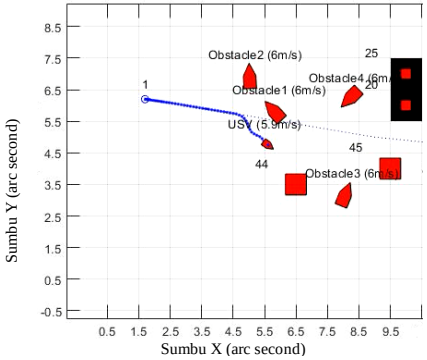


Gambar 4.20 Bagian 3 Lintasan Gabungan

Analisis keseluruhan pengujian akan dilakukan pada setiap bagian dan perngujian dilakukan 3 kali dengan variasi yang diubah adalah jarak aman.

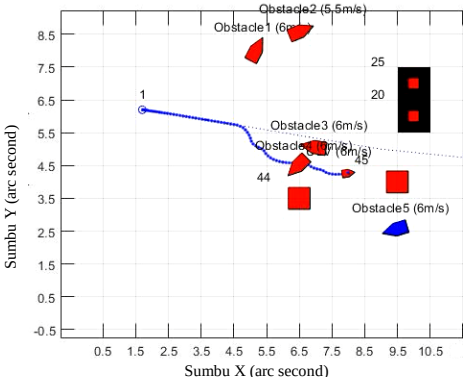
4.2.1 Pengujian Lintasan Gabungan dengan Jarak Aman 20 Meter

Pada bagian 1 dilakukan penghindaran pertama untuk menghindari halangan dinamis 2 dengan mengaktifkan *output fuzzy rules "right_small"* untuk belok ke kanan, setelah itu diikuti penghindaran untuk menghindari halangan dinamis 1 dengan membelok ke arah kanan mengaktifkan *output fuzzy rules "right_small"*. Pada Gambar 4.21 ditampilkan bahwa USV mampu menghindari kedua halangan dinamis tersebut.



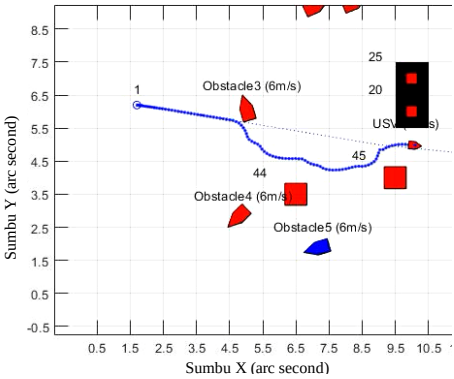
Gambar 4.21 Penghindaran Pertama dan Kedua pada Bagian 1

Setelah menghindari kedua halangan dinamis pertama, USV menghindari halangan 3 dan 4 secara bersamaan dengan mengaktifkan *output fuzzy rules* "*left_small*" untuk belok ke kiri. Terlihat pada Gambar 4.22. bahwa USV mampu menghindari dua halangan dinamis tersebut dan mengabaikan halangan statis 44.



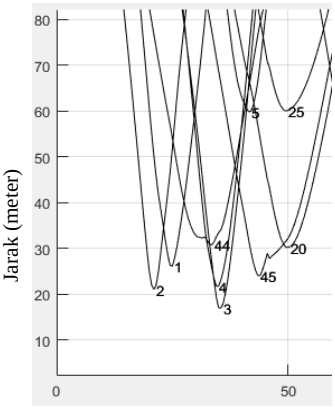
Gambar 4.22 Proses Penghindaran Ketiga pada Bagian 1

Setelah USV menghindari kedua halangan dinamis tersebut, USV menghindari halangan statis 45 dengan belok ke kiri mengaktifkan *output fuzzy rules* "*left_small*". Setelah menghindari halangan statis 45, USV kembali ke lintasan dan mengabaikan halangan statis tetap 20 seperti yang digambarkan dalam Gambar 4.23.



Gambar 4.23 Proses Kembalinya USV ke Lintasan pada Bagian 1

Pada bagian 1 didapatkan bahwa USV mampu menghindari semua halangan di lintasan dengan jarak halangan terdekat ialah halangan dinamis 3 sebesar 17,06 meter seperti yang ditunjukkan kurva dalam Gambar 4.24. Dari kurva tersebut dapat dicari data seperti di Tabel 4.8.

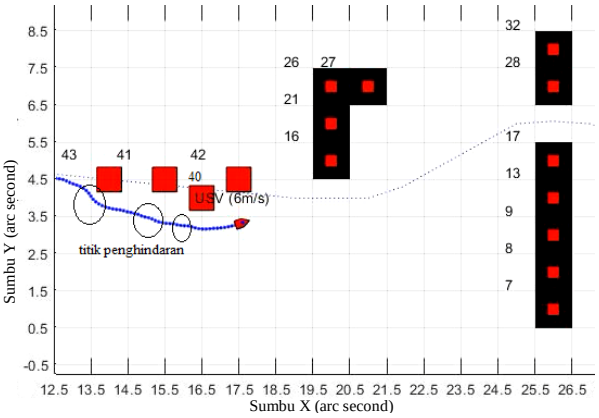


Gambar 4.24 Kurva Deteksi Halangan pada Bagian 1 Lintasan Gabungan

Tabel 4.8 Hasil Akhir Bagian 1 Lintasan Gabungan dengan Jarak Aman 20 Meter

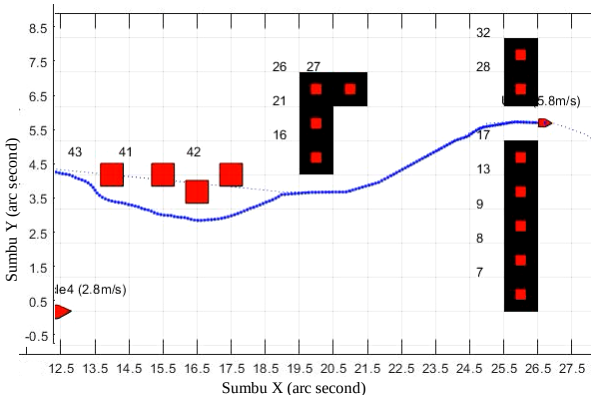
Jenis Halangan	Nomor Halangan	Deteksi Jarak Terdekat (meter)	Status Penghindaran	Fuzzy Rules yang aktif
Dinamis	1	26,36	Mampu dihindari	Right_small (102)
Dinamis	2	21,08	Mampu dihindari	Right_small (16)
Dinamis	3	17,06	Mampu dihindari	Left_small (18)
Dinamis	4	21,77	Mampu dihindari	
Dinamis	5	59,5	Mampu dihindari	-
Statis	44	30,95	Mampu dihindari	-
Statis	45	24,08	Mampu dihindari	Left_small (68)
Statis Tetap	20	30,33	Mampu dihindari	Stay (30)

Pada bagian 2 terdapat 3 penghindaran yaitu untuk menghindari halangan statis 43, 41, dan 40. Pertama-tama diawali dengan belok ke kanan menghindari halangan statis 43 dengan mengaktifkan *output fuzzy rules* "*right_small*" , setelah itu diikuti penghindaran halangan statis 41 dan 40 dengan output yang sama seperti terlihat dalam Gambar 4.24.



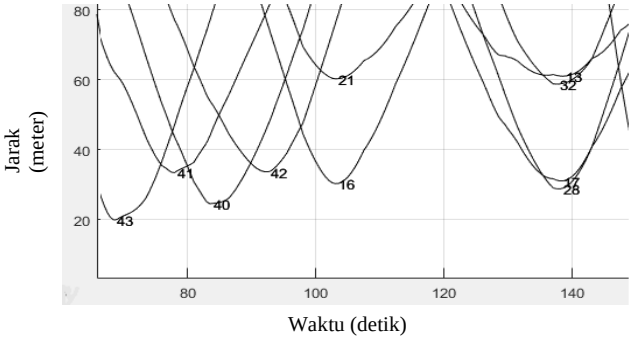
Gambar 4.25 Proses Penghindaran pada Bagian 2 Lintasan Gabungan

Setelah melewati keempat halangan statis 43,41,40, dan 42 USV akan kembali ke lintasan dan mengabaikan ketiga halangan statis 16, 17, dan 28. Halangan tersebut diabaikan karena memenuhi jarak aman dari USV.



Gambar 4.26 USV Menyelesaikan Bagian 2 Lintasan Gabungan

Pada bagian 2 didapatkan bahwa USV mampu menghindari semua halangan di lintasan dengan jarak halangan terdekat ialah halangan statis 43 sebesar 19,96 meter seperti yang ditunjukkan kurva dalam Gambar 4.27. Dari kurva tersebut dapat dicari data seperti di Tabel 4.9.

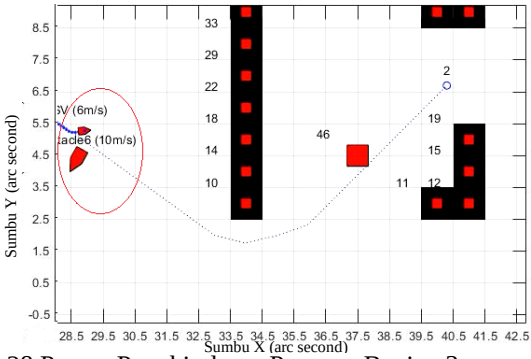


Gambar 4.27 Kurva Deteksi Jarak Bagian 2 Lintasan Gabungan

Tabel 4.9 Hasil Akhir Bagian 2 Lintasan Gabungan dengan Jarak Aman 20 Meter

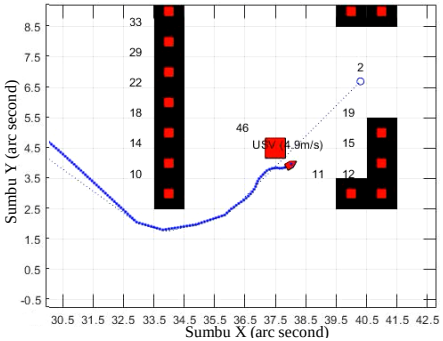
Jenis Halangan	Nomor Halangan	Deteksi Jarak Terdekat (meter)	Status Penghindaran	Fuzzy Rules yang aktif
Statis	43	19,96	Mampu dihindari	Right_small (24 dan 77)
Statis	41	33,5	Mampu dihindari	Right_small (16)
Statis	40	24,63	Mampu dihindari	Right_small, stay (16 dan 23)
Statis	42	33,78	Mampu dihindari	-
Statis Tetap	16	30,63	Mampu dihindari	-
Statis Tetap	17	31,35	Mampu dihindari	-
Statis Tetap	28	28,88	Mampu dihindari	-

Pada bagian 3 dilakukan penghindaran pertama kali untuk menghindari halangan dinamis 6 yang memiliki kecepatan lebih tinggi dari USV. Proses menghindar diawali dengan aktifnya *output fuzzy rules* "*right_small*" yang mengakibatkan USV belok ke kanan untuk menghindari halangan. Proses dapat dilihat di Gambar 4.28.

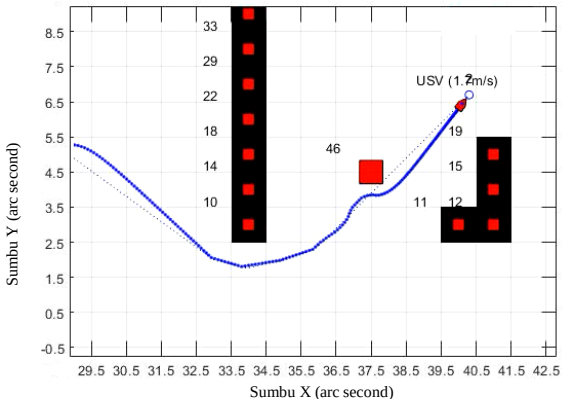


Gambar 4.28 Proses Penghindaran Pertama Bagian 3

Setelah menghindari halangan dinamis 6, USV kembali ke lintasan dan mengabaikan halangan statis tetap 10 lalu melakukan penghindaran untuk menghindari halangan statis 46 seperti pada Gambar 4.29. Penghindaran dilakukan dengan belok ke kanan mengaktifkan *output fuzzy rules* "*right_small*" setelah itu USV kembali ke lintasan dan berhenti di titik *waypoint* 2 yang terlihat di Gambar 4.30.

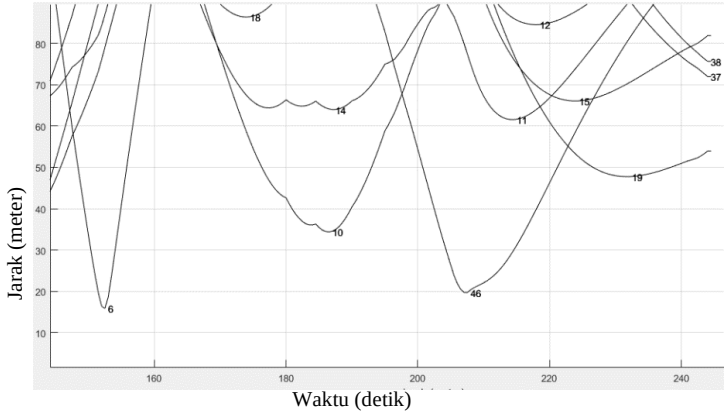


Gambar 4.29 Proses Penghindaran Kedua Bagian 3



Gambar 4.30 Akhir Pengujian pada Lintasan Gabungan dengan Jarak Aman 20 Meter

Pada bagian 3 didapatkan bahwa USV mampu menghindari semua halangan di lintasan dengan jarak halangan terdekat ialah halangan dinamis sejauh 15,84 meter seperti yang ditunjukkan kurva dalam Gambar 4.27. Dari kurva tersebut dapat dicari data seperti di Tabel 4.9. Dalam pengujian ini USV menempuh total perjalanan 1304 meter.



Gambar 4.31 Kurva Deteksi Halangan Bagian 3

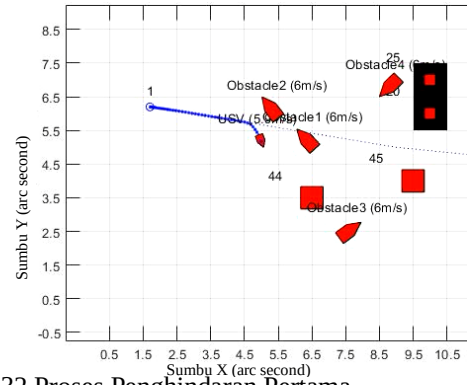
Jenis Halangan	Nomor Halangan	Deteksi Jarak Terdekat (meter)	Status Penghindaran	Fuzzy Rules yang aktif
Dinamis	6	15,84	Mampu dihindari	Right_small (40)
Statis Tetap	10	34,36	Mampu dihindari	-
Statis	46	19,71	Mampu dihindari	Right_small, (76)
Statis Tetap	11	61,56	Mampu dihindari	-

Tabel 4.10 Hasil Akhir Bagian 3 Lintasan Gabungan dengan Jarak Aman 20 Meter

Dari keseluruhan pengujian pada lintasan gabungan dengan jarak aman 20 meter didapatkan bahwa USV mampu menghindari semua halangan yang ada dalam lintasan dengan deteksi jarak halangan terdekat ialah halangan dinamis 6 dengan jarak sebesar 15,64 meter. Terdapat 3 *output fuzzy rules* yang aktif yaitu "*right_small*", "*stay*", dan "*left_small*". Terdapat 10 *fuzzy rules* yang aktif dari 105 *rules* yang ada.

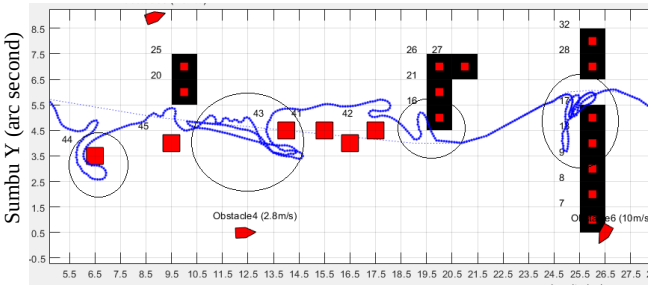
4.2.2 Pengujian Lintasan Gabungan dengan Jarak Aman 40 Meter

Pada pengujian dengan jarak aman sebesar 40 meter, hanya mampu berjalan dengan lancar ketika melewati halangan dinamis 1 dan 2 dengan mengaktifkan *output fuzzy rules* "*right_small*" untuk belok ke kanan seperti pada Gambar 4.32. Setelah penghindaran pertama, USV kesulitan untuk menghindari halangan yang lain karena sistem mendeteksi banyak halangan yang memicu beberapa *fuzzy rules* untuk aktif secara bersamaan sehingga USV kesulitan untuk menghindari halangan dan kembali ke lintasan.



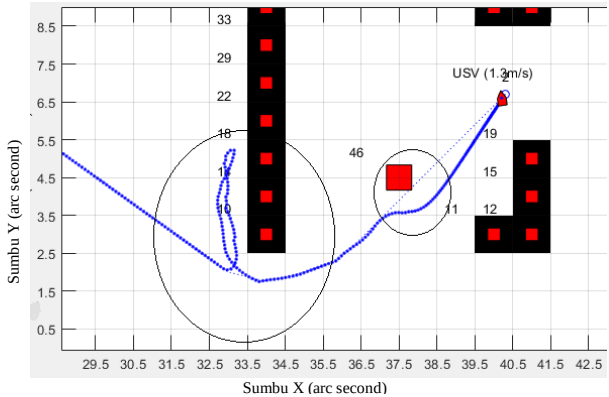
Gambar 4.32 Proses Penghindaran Pertama

Setelah USV menghindari 2 halangan pertama, USV kesulitan untuk menghindar kembali dan melakukan tabrakan pada halangan statis 44 dan setelah itu USV berputar-putar ke arah yang tidak jelas sebanyak 3 kali dan setelah itu mampu menghindari empat halangan berdekatan dengan mengaktifkan *output fuzzy rules* "*left_medium*" untuk belok ke kiri dan setelah itu menabrak halangan statis tetap 16. Setelah itu USV kesulitan menghindari kedua halangan tetap 17 dan 28 yang menjepit lintasan dan berputar-putar selama beberapa lama sampai akhirnya bisa melewati setelah menabrak halangan statis tetap 17. Dapat dilihat pada Gambar 4.33 bahwa USV kerap mengalami kegagalan dalam menghindari halangan. USV berputar-putar disebabkan oleh sistem yang tidak mampu mendefinisikan output yang akan digunakan.



Gambar 4.33 Kondisi Lintasan Gabungan dengan Jarak Aman 40 Meter

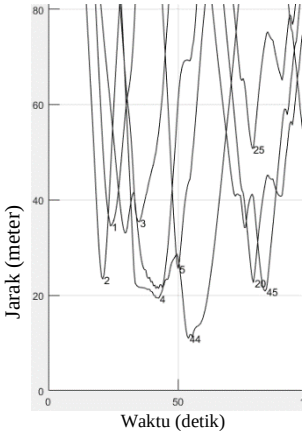
Pada bagian 3 lintasan gabungan, USV kembali kesulitan menghindari halangan statis tetap 10 dan berakhir dapat menghindari setelah berputar beberapa kali. Pada bagian 3, USV hanya mampu menghindari halangan statis 46 dengan mengaktifkan *output fuzzy rules* "*right_small*" dan menyelesaikan perjalanan ke titik *waypoint* 2 seperti pada Gambar 4.34.



Gambar 4.34 Bagian 3 Lintasan Gabungan dengan Jarak Aman 40 Meter

Dari terdeteksinya tiga tabrakan dan USV yang kesulitan kembali ke lintasan dapat diambil kesimpulan bahwa pengujian dengan jarak aman 40 meter gagal dan nilai jarak aman 40 meter bukan merupakan nilai yang ideal untuk sistem karena sistem kesulitan memilih *fuzzy rules* mana yang diaktifkan karena banyaknya estimasi tabrakan dengan halangan yang terdeteksi oleh sistem sehingga fuzzy aktif secara terus menerus dan dalam beberapa keadaan beberapa rules aktif secara bersamaan dan sistem tidak mampu berjalan dengan ideal.

Dari pembacaan kurva yang ada di Gambar 4.35 didapatkan bahwa USV tidak mempu menghindari halangan statis 44 karena nilai jarak terdekatnya bernilai 10,96 meter di mana lebih kecil dari jarak bebas tabrakan yang ditentukan untuk halangan statis (didapatkan dari penjumlahan panjang USV dengan setengah panjang halangan terkait) sebesar 16,5 meter. Detail dari kurva disajikan dalam Tabel 4.11.



Gambar 4.35 Deteksi Halangan pada Bagian 1 Lintasan Gabungan dengan Jarak Aman 40 Meter

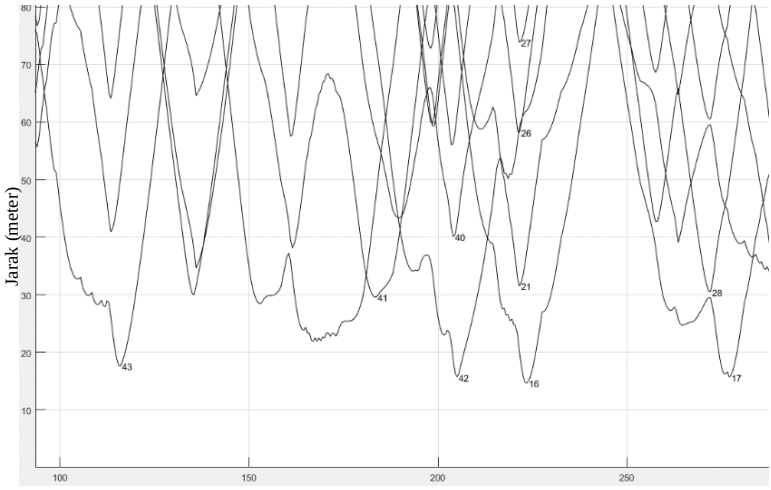
Tabel 4.11 Hasil Akhir Bagian 1 Lintasan Gabungan dengan Jarak Aman 40 Meter

Jenis Halangan	Nomor Halangan	Deteksi Jarak Terdekat (meter)	Status Penghindaran	Fuzzy Rules yang aktif
Dinamis	1	34,5	Mampu dihindari	Right_small (102)
Dinamis	2	23,36	Mampu dihindari	Right_small (16)
Dinamis	3	17,06	Mampu dihindari	Right_small (83)
Dinamis	4	21,77	Mampu dihindari	
Dinamis	5	59,5	Mampu dihindari	Output mengalami *crash*
Statis	44	10,96	Gagal	Output mengalami *crash*
Statis	45	24,08	Mampu dihindari	Output mengalami *crash*
Statis Tetap	20	30,33	Mampu dihindari	Output mengalami *crash*

Kurva pada Gambar 4.35 menyatakan bahwa deteksi jarak halangan pada bagian 1 menunjukkan bahwa secara nilai deteksi jarak terbukti ada tabrakan, tetapi pada kondisi lain beberapa halangan tetap

mampu dihindari walaupun dengan keadaan sistem tidak mampu mendefinisikan output yang sesuai seperti data pada Tabel 4.11.

Pada bagian 2 pembacaan kurva didapati bahwa USV menabrak halangan 16 karena deteksi jarak paling dekat ialah 14,7 meter di mana lebih kecil dari jarak bebas tabrakan yang ditentukan untuk halangan statis (didapatkan dari penjumlahan panjang USV dengan setengah panjang halangan terkait) sebesar 16,5 meter di mana terbukti dalam Gambar 4.33 sebelumnya bahwa USV gagal menghindari halangan statis 16. Sedangkan untuk halangan 43 USV kesulitan untuk menghindari sehingga melakukan perputaran beberapa kali. Sama seperti sebelumnya pada Tabel 4.10, USV juga tidak mampu mendefinisikan output dari *Fuzzy Logic Control* dalam pembacaan yang disajikan dalam Tabel 4.12

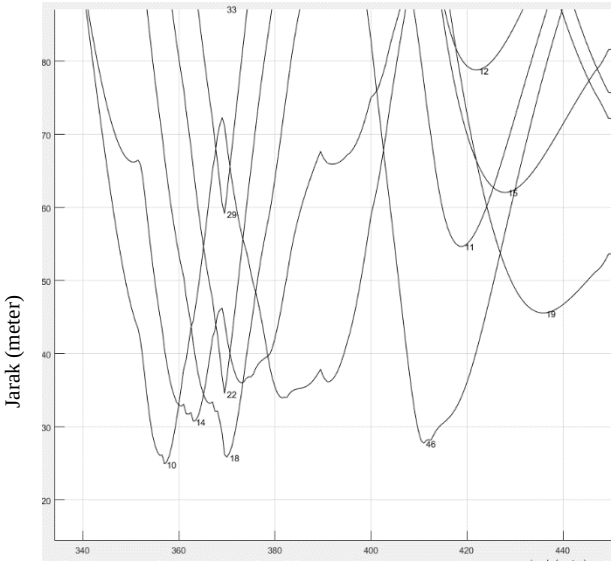


Gambar 4.36 Kurva Deteksi Halangan Bagian 2 Lintasan Gabungan dengan Jarak Aman 40 Meter

Tabel 4.12 Hasil Akhir Bagian 2 Lintasan Gabungan dengan Jarak Aman 40 Meter

Jenis Halangan	Nomor Halangan	Deteksi Jarak Terdekat	Status Penghindaran	Fuzzy Rules yang aktif
		(meter)		
Statis	43	19,96	Mampu dihindari	Left medium (7)
Statis	41	33,5	Mampu dihindari	Left small (16)
Statis	40	24,63	Mampu dihindari	Stay (51)
Statis	42	15,72	Mampu dihindari	Output mengalami *crash*
Statis Tetap	16	14,71	Gagal	Output mengalami *crash*
Statis Tetap	17	17,71	Gagal	Output mengalami *crash*
Statis Tetap	28	30,66	Mampu dihindari	Stay (98)

Pada bagian 3 dapat dikatakan sebagai bagian yang paling lancar dalam proses simulasi ini dikarenakan tidak terjadi tabrakan. Dapat dilihat dari kurva yang ditunjukkan oleh Gambar 4.37 bahwa USV tidak menabrak halangan di bagian 3. USV mengalami kesulitan untuk menghindari halangan statis tetap 10 karena kesulitan untuk mendefinisikan output. Dalam pengujian ini USV menempuh perjalanan 2532 meter.



Gambar 4.37 Hasil Deteksi Halangan Bagian 3 Lintasan Gabungan dengan Jarak Aman 40 Meter

Tabel 4.13 Hasil Akhir Bagian 3 Lintasan Gabungan dengan Jarak Aman 40 Meter

Jenis Halangan	Nomor Halangan	Deteksi Jarak Terdekat (meter)	Status Penghindaran	Fuzzy Rules yang aktif
Dinamis	6	107,5	Mampu dihindari	-
Statis Tetap	10	24,89	Mampu dihindari	Output mengalami crash
Statis	46	28,05	Mampu dihindari	Right_small, (83)
Statis Tetap	11	54,7	Mampu dihindari	-

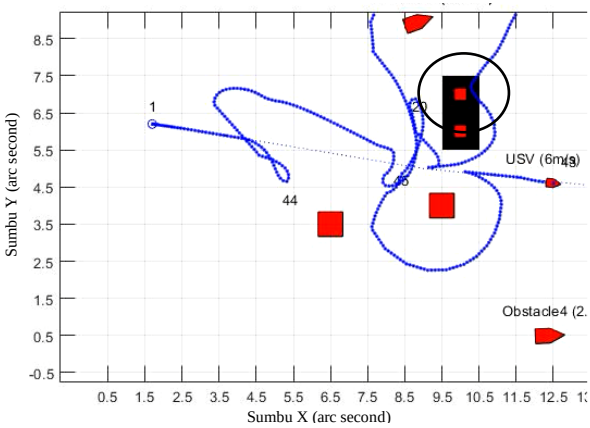
Pengujian ini dinyatakan gagal karena diketahui USV gagal menghindari halangan sebanyak 3 kali dan gagal mendefinisikan output fuzzy rules sebanyak 8 kali.

4.2.3 Pengujian Lintasan Gabungan dengan Jarak Aman 60 Meter

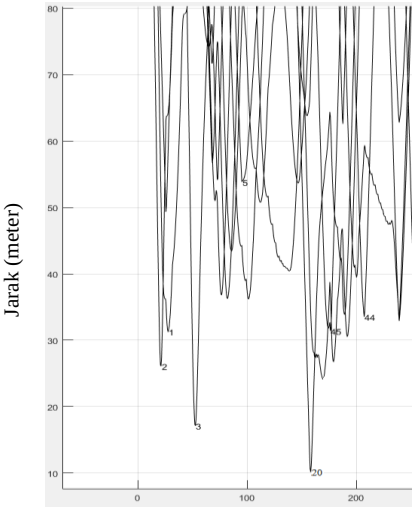
Pada bagian 1 pengujian ini halangan 20 tidak dapat dihindari oleh USV. Namun, semua halangan yang lain mampu dihindari tetapi tidak ada *output fuzzy rules* yang dapat didefinisikan. Dapat dilihat pada Gambar 4.38. bahwa USV berjalan secara tidak beraturan dan proses penghindaran juga tidak beraturan dan memiliki output yang jelas.

Jika kita melihat pada kurva yang terdapat pada Gambar 4.39 maka benar bahwa jarak halangan statis 20 dari USV lebih kecil dari nilai aman maka dapat diambil kesimpulan bahwa benar USV gagal menghindari halangan statis tetap 20.

Walaupun USV mampu menghindari halangan lainnya di bagian 1, proses penghindaran masih belum bisa dikatakan berhasil karena tidak terdapat *output fuzzy rules* yang jelas. *Output fuzzy rules* yang didapatkan saling bertumpukkan sehingga USV kesulitan untuk memproses output yang pasti.



Gambar 4.38 Bagian 1 Lintasan Gabungan dengan Jarak Aman 60 Meter

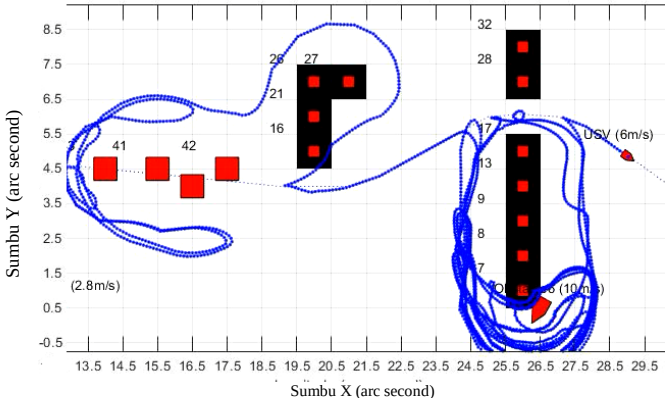


Gambar 4.39 Kurva Deteksi Halangan Bagian 1 Lintasan Gabungan dengan Jarak Aman 60 Meter

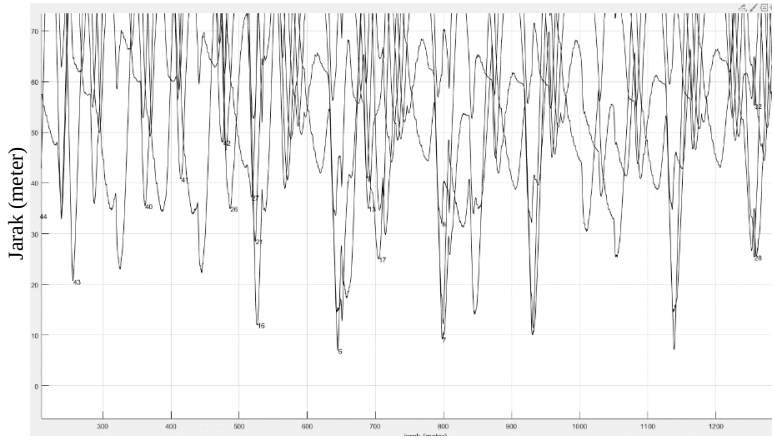
Tabel 4.14 Hasil Akhir Bagian 1 Lintasan Gabungan dengan Jarak Aman 60 Meter

Jenis Halangan	Nomor Halangan	Deteksi Jarak Terdekat (meter)	Status Penghindaran	Fuzzy Rules yang aktif
Dinamis	1	31,25	Mampu dihindari	Output mengalami *crash*
Dinamis	2	26,11	Mampu dihindari	Output mengalami *crash*
Dinamis	3	17,27	Mampu dihindari	Output mengalami *crash*
Dinamis	4	84,86	Mampu dihindari	Output mengalami *crash*
Dinamis	5	53,85	Mampu dihindari	Output mengalami *crash*
Statis	44	33,54	Mampu dihindari	Output mengalami *crash*
Statis	45	26,76	Mampu dihindari	Output mengalami *crash*
Statis Tetap	20	30,33	Gagal	Output mengalami *crash*

Pada bagian 2, USV gagal menghindari halangan tetap 16 dan 17 (halangan 7 termasuk bagian dari halangan 17) seperti yang dapat dilihat pada Gambar 4.40. Pada kondisi halangan lainnya, USV mampu melakukan penghindaran tetapi tidak dapat mendefinisikan *output fuzzy rules*.



Gambar 4.40 Bagian 3 Lintasan Gabungan dengan Jarak Aman 60 Meter



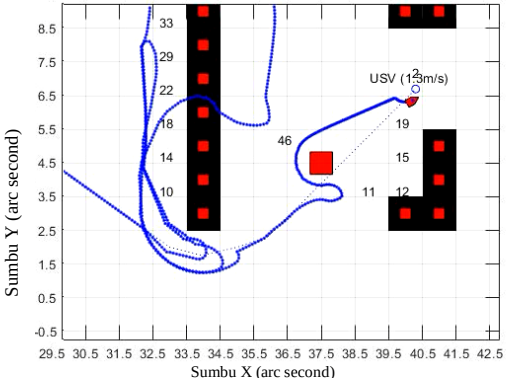
Gambar 4.41 Kurva Bagian 2 Lintasan Gabungan dengan Jarak Aman 60 Meter

Dari kurva yang ada di Gambar 4.40 dapat disusun Tabel 4.15. Bila mengacu pada data di Tabel 4.15, secara pengukuran juga terjadi tabrakan di halangan tetap 16 dan 17. Karena keduanya memiliki nilai lebih kecil dari nilai aman yang telah ditentukan pada halangan terkait

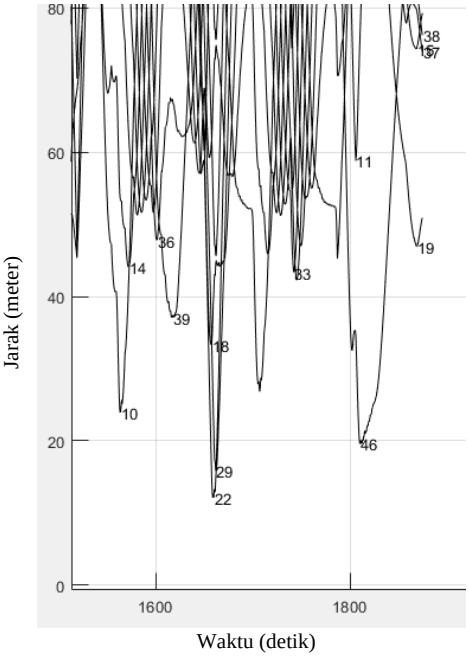
Tabel 4.15 Hasil Akhir Bagian 2 Lintasan Gabungan dengan Jarak Aman 60 Meter

Jenis Halangan	Nomor Halangan	Deteksi Jarak Terdekat (meter)	Status Penghindaran	Fuzzy Rules yang aktif
Statis	43	22,88	Mampu dihindari	Output mengalami *crash*
Statis	41	22,27	Mampu dihindari	Output mengalami *crash*
Statis	40	35,49	Mampu dihindari	Output mengalami *crash*
Statis	42	48,53	Mampu dihindari	Output mengalami *crash*
Statis Tetap	16	12,03	Gagal	Output mengalami *crash*
Statis Tetap	17	7,21	Gagal	Output mengalami *crash*
Statis Tetap	28	25,36	Mampu dihindari	Output mengalami *crash*

Pada bagian 3 USV gagal menghindari halangan statis tetap 10 (halangan 18 dan 22 termasuk bagian dari halangan 10) tetapi mampu menghindari halangan statis 46 dan halangan statis tetap 11 seperti yang terlihat pada Gambar 4.42. Sama seperti sebelumnya, *output fuzzy rules* juga tidak terdefinisi di bagian 3.



Gambar 4.43 Bagian 3 Lintasan Gabungan dengan Jarak Aman 60 Meter



Gambar 4.42 Kurva Deteksi Halangan Bagian 3 Lintasan Gabungan dengan Jarak Aman 60 Meter

Tabel 4.16 Hasil Akhir Bagian 3 Lintasan Gabungan dengan Jarak Aman 60 Meter

Jenis Halangan	Nomor Halangan	Deteksi Jarak Terdekat (meter)	Status Penghindaran	Fuzzy Rules yang aktif
Dinamis	6	7	Gagal	-
Statis Tetap	10	12,81	Gagal	-
Statis	46	19,71	Mampu dihindari	Output mengalami crash
Statis Tetap	11	58,82	Mampu dihindari	Output mengalami crash

Dari pembacaan kurva yang ada pada Gambar 4.43 diketahui bahwa USV gagal menghindari halangan 10, sedangkan pada kurva di Gambar 4.40 diketahui bahwa USV gagal menghindari halangan dinamis 6. USV juga mampu menghindari halangan statis 46 dan statis tetap 11 dengan masing masing dalam keadaan *output fuzzy rules* tak terdefinisi. Dalam perjalanan ini USV menempuh perjalanan 10.289 meter.

4.3 Analisis Fuzzy Rules yang Aktif

Dari seluruh pengujian, didapatkan di dapatkan terdapat 20 *fuzzy rules* yang aktif seperti yang terlihat di Tabel 4.17. Untuk output dari fuzzy sendiri hanya terdapat 4 output *output fuzzy rules* yang aktif , di antaranya *right_small, left_small, stay,* dan *left_medium.*

Jumlah *fuzzy rules* yang aktif berjumlah 20 dari total 105 yang ada, di mana *fuzzy* aktif sebanyak 31 kali untuk melakukan proses penghindaran. *Fuzzy rules* yang paling sering digunakan adalah *rules* nomor 16 dan 83 di mana masing-masing digunakan sebanyak 4 kali.

Tabel 4.17 Daftar *Fuzzy Rules* yang Aktif

Rules	Input			Output	Jumlah Aktif
	Jarak	Posisi	Perubahan Sudut	Steering Degree	
7	dekat	right large	in front	left medium	1
15	sedang	left large	in front	stay	1
16	sedang	left medium	in front	right small	4
18	sedang	right small	in front	left small	1
22	agak jauh	left large	in front	stay	1
23	agak jauh	left medium	in front	stay	1
24	agak jauh	left small	in front	right small	1
30	jauh	left medium	in front	stay	1
32	jauh	zero	in front	right small	2
40	dekat	left small	to right	left medium	1
53	sedang	zero	to right	left small	2
68	jauh	left small	to right	left small	1
76	dekat	left medium	to left	right small	1
77	dekat	left large	to left	right small	3
83	agak dekat	left medium	to left	right small	4
90	sedang	left medium	to left	stay	1
91	sedang	left large	to left	stay	1
98	agak jauh	left large	to left	stay	1
100	jauh	right medium	to left	right small	1
102	jauh	zero	to left	stay	2

Halaman ini sengaja dikosongkan

BAB 5
PENUTUP

Dari perancangan dan proses pengujian sistem yang telah dilakukan, dapat ditulis kesimpulan. Untuk kekurangan dan kendala yang dihadapi, dituliskan dalam bagian saran, untuk membantu penelitian selanjutnya.

5.1 Kesimpulan

Setelah melakukan rangkaian simulasi pengujian untuk menguji sistem, didapatkan beberapa kesimpulan sebagai berikut:

1. Sistem yang dirancang mampu menghindari halangan statis, halangan dinamis, maupun gabungan dari keduanya
2. Didapatkan jarak deteksi halangan paling dekat sebesar 15,84 meter.
3. Didapatkan bahwa jarak aman 20 meter merupakan jarak yang ideal untuk proses menghindari halangan.
4. Jarak aman yang ditentukan berpengaruh terhadap proses arah menghindar dan jarak yang ditempuh

5.2 Saran

Untuk pengembangan selanjutnya pada topik penelitian *obstacle avoidance* untuk USV, terdapat beberapa saran yang diberikan, antara lain:

- Ditambahkan sistem pengendalian kecepatan untuk proses penghindaran halangan
- Ditambahkan simulasi model non linier kapal yang sudah disertai dengan model autopilotnya
- Ditambahkan proses analisis optimasi *fuzzy rules*

Halaman ini sengaja dikosongkan

DAFTAR PUSTAKA

[1] S. Kusumadewi, Aplikasi Logika Fuzzy Untuk Mendukung Keputusan, Yogyakarta: Graha Ilmu, 2004.

[2] Widodo, Sistem Neuro Fuzzy, 2005.

[3] A. F. Setiawan, Dynamic Modelling And Controlling Unmanned Surface Vehicle, 2019.

[4] A. G. Salman, BINUS University, [Online]. Available: https://socs.binus.ac.id/2012/03/02/pemodelan-dasar-sistem-fuzzy/. [Accessed 30 Mei 2020].

[5] Zimmermann, Fuzzy Sets Theory and Its Applications, 1991.

[6] T. J. Ross, Fuzzy Logic with Engineering Applications, 1995.

[7] A. Najmurrokhman, "Mamdani based Fuzzy Logic Controller for A Wheeled Mobile Robot with Obstacle Avoidance Capability," 2019.

[8] C. G. Rusu, "Fuzzy Based Obstacle Avoidance System for Mobile Robot," 2016.

Halaman ini sengaja dikosongkan

LAMPIRAN

a. Perancangan Persamaan Gerak USV

```
    classdef ShipV3 < handle
properties
    % parametricValue
    waypoint
    wpBoxSize
    shipSize
    type
    name
    color

    % calculatedValue
    totalMovementDistance = 0
    plots = []
    currentCoordinate %[x,y,headingDegree]
    currentPolyMap
    currentSpeed
    runningTime = 0
    isFinish = false
    isStatic = false
end

methods

    function this =
ShipV3(shipSize,waypoint,wpBoxSize,type,name,color)
        this.shipSize = shipSize; %[panjang lebar]
        this.wpBoxSize = wpBoxSize;
        this.type = type;
        this.name = name;
        this.color = color;
        if(size(waypoint,1)==1)
            this.isStatic = true;
            this.currentCoordinate = [ waypoint(1:2) 0];
            this.calculatePoly();
            this.draw();
        else
            this.currentCoordinate = waypoint(1,[1 2
4]);
            this.currentSpeed = 0;
            this.waypoint = waypoint(2:end,:);
            deltaPoint = this.waypoint(1,1:2) -
this.currentCoordinate(1:2);
            this.currentCoordinate(3) =
atan2(deltaPoint(2),deltaPoint(1));
        end
    end
```

```
        function move(this,dt)
            tolerance = 5; %meter
            % only do this fn if this.waypoint is not empty
            if size(this.waypoint,1) > 0
                % get data from waypoint
                deltaPoint = this.waypoint(1,1:2) -
this.currentCoordinate(1:2);
                deltaSpeedSq =
(this.waypoint(1,3)/this.wpBoxSize)^2 -
(this.currentSpeed/this.wpBoxSize)^2;

                % if distance between ship and target
waypoint less than
                % some tolerance
                distance = norm(deltaPoint);
                if(distance<(tolerance/this.wpBoxSize))
                    % purge old waypoint
                    this.waypoint = this.waypoint(2:end,:);
                    if ~isempty(this.waypoint)
                        deltaPoint = this.waypoint(1,1:2) -
this.currentCoordinate(1:2);
                        this.currentCoordinate(3) =
atan2(deltaPoint(2),deltaPoint(1));
                    end
                end

                cdeg = this.currentCoordinate(3);

                % find acceleration
                acceleration = deltaSpeedSq/(2*distance);
                accel =
acceleration*[ cos(this.currentCoordinate(3))
sin(this.currentCoordinate(3)) ];

                % find speed vector
                speedVector =
(this.currentSpeed*[cos(this.currentCoordinate(3))
sin(this.currentCoordinate(3))]/this.wpBoxSize) + accel*dt;
                this.currentSpeed =
norm(speedVector)*this.wpBoxSize;

                % find new position
                this.currentCoordinate(1:2) =
this.currentCoordinate(1:2)+speedVector*dt;
                this.totalMovementDistance =
this.totalMovementDistance +
(norm(speedVector*dt)*this.wpBoxSize)
```

```
                % add running time
                this.runningTime = this.runningTime + dt;

                % draw
                this.calculatePoly();
                this.draw();
            else
                this.isFinish = true;
            end
        end


        function calculatePoly(this)
            % calculate USV PolyPosition
            frontX =
this.shipSize(1)/this.wpBoxSize/2*cos(this.currentCoordinate
(3));
            frontY =
this.shipSize(1)/this.wpBoxSize/2*sin(this.currentCoordinate
(3));
            leftX =
this.shipSize(2)/this.wpBoxSize/2*cos(this.currentCoordinate
(3)+(pi/2));
            leftY =
this.shipSize(2)/this.wpBoxSize/2*sin(this.currentCoordinate
(3)+(pi/2));

            if(this.type=="ship")
                this.currentPolyMap(1,:) =
[  this.currentCoordinate(1)+frontX ...

this.currentCoordinate(1)+leftX ...
                            this.currentCoordinate(1)+leftX-
frontX ...
                            this.currentCoordinate(1)-leftX-
frontX ...
                            this.currentCoordinate(1)-
leftX ];

                this.currentPolyMap(2,:) =
[  this.currentCoordinate(2)+frontY ...

this.currentCoordinate(2)+leftY ...
                            this.currentCoordinate(2)+leftY-
frontY ...
                            this.currentCoordinate(2)-leftY-
frontY ...
                            this.currentCoordinate(2)-
leftY ];
            elseif(this.type=="box")
```

```
                this.currentPolyMap(1,:) =
[  this.currentCoordinate(1)+leftX+frontX ...
                            this.currentCoordinate(1)+leftX-
frontX ...
                            this.currentCoordinate(1)-leftX-
frontX ...
                            this.currentCoordinate(1)-
leftX+frontX ];

                this.currentPolyMap(2,:) =
[  this.currentCoordinate(2)+leftY+frontY ...
                            this.currentCoordinate(2)+leftY-
frontY ...
                            this.currentCoordinate(2)-leftY-
frontY ...
                            this.currentCoordinate(2)-
leftY+frontY ];
            end
        end


        function draw(this)
            % draw USV object
            for p=1:size(this.plots,2)
                delete(this.plots(p))
            end
            this.plots = [];
            this.plots(end+1) =
patch(this.currentPolyMap(1,:),this.currentPolyMap(2,:),this
.color);
            if(~this.isStatic)
                this.plots(end+1) =
text(   this.currentCoordinate(1)-1.3, ...
                    this.currentCoordinate(2)+0.7, ...
                    [this.name ' ('
num2str(round(this.currentSpeed,1)) 'm/s)']);
            end
        end


    end
end
```

b. Program Simulasi USV

```
    classdef USV < ShipV3
%UNTITLED Summary of this class goes here
%   Detailed explanation goes here
```

```
    properties
        % lidar
        lidarRay = []

        % obstacle
        obstacle = []

        % last distance recorder
        distanceRecorder = []


    end

    methods


        function this =
USV(shipSize,waypoint,wpBoxSize,name,color)
            this =
this@ShipV3(shipSize,waypoint,wpBoxSize,"ship",name,color);
        end


        function addObstacle(this, obstacle)
            if isempty(this.obstacle)
                this.obstacle = [ obstacle ];
            else
                this.obstacle(end+1) = obstacle;
            end
        end


        function result = calculateAllObstacleDistance(this,
previous)
            result =
zeros(length(this.obstacle),size(previous,2)+1);
            for obs = 1:length(this.obstacle)
                result(obs,end) =
norm(this.currentCoordinate(1:2) ...
                    -
this.obstacle(obs).currentCoordinate(1:2)) ...
                    *this.wpBoxSize;
            end
            if ~isempty(previous)
                result(:,1:end-1) = previous;
            end
        end


        function plotAllObstacleDistance(~, data, dt)
```

```
            color = ['b-'; 'g-'; 'r-'; 'c-'; 'm-'; 'g-'; 'k-
'];
            for nData = 1:size(data,1)
                pl = plot(0:dt:dt*size(data,2)-dt,
data(nData,:), color(nData))
            end
            set( ...
                gca, ... % get current axis
                'XLim', [0 dt*size(data,2)-dt], ... % set X
range
                'YLim', [0 80], ... % set Y range
                'GridLineStyle', '-', ... % set grid style
                'XGrid', 'on', ... % enable grid
                'YGrid', 'on', ...
                'Ydir', 'normal' ...
            );
        end


        function
distanceDetection(this,dt,range,clearance,fov,fuzzyObj)
            rangeNormalize = range/this.wpBoxSize;
%             clearanceNormalize = clearance/this.wpBoxSize;
            % no obstacle flag
            noObstacleNearby = true;
            % for each obstacle, move the obstacle by dt
then calculate distance
            for ob=1:length(this.obstacle)
                if(~this.obstacle(ob).isFinish &&
~this.obstacle(ob).isStatic)
                    this.obstacle(ob).move(dt);
                    this.obstacle(ob).draw();
                end

                % calc distance between usv & obstacle
                % d = distance to obstacle
                % h = heading to obstacle
                d = norm(this.currentCoordinate(1:2) -
this.obstacle(ob).currentCoordinate(1:2));
                h =
atan2(this.obstacle(ob).currentCoordinate(2)-
this.currentCoordinate(2), ...
                    this.obstacle(ob).currentCoordinate(1)-
this.currentCoordinate(1)) - this.currentCoordinate(3);

                 % if distance > range, ignore it, else
calculate minimum fn.
                if ( d <= rangeNormalize && h<=0.5*fov &&
h>=-(0.5*fov))
```

```
                    % find if previous detection on this obj
already done
                    if size(this.distanceRecorder,1)>0
                        dd =
this.distanceRecorder(this.distanceRecorder(:,1)==ob,:);
                    else
                        dd=[];
                    end
                    if size(dd,1)==1

this.distanceRecorder(this.distanceRecorder(:,1)==ob,2:7)
= ...

[ this.distanceRecorder(this.distanceRecorder(:,1)==ob,[4:7]
) d h ];
                    else
                        %if not exist, add new

this.distanceRecorder(end+1,1:7)=[ ob d h d h d h ];
                    end
                else
                    % if obstacle outside detection range,
delete it
                    if size(this.distanceRecorder,1)>0
                        this.distanceRecorder =
this.distanceRecorder(this.distanceRecorder(:,1)~=ob,:);
                    end
                end

                % for each row in distanceDetection,
calculate minimum distance
                TT = [1 2*dt 4*dt^2 ; 1 dt dt^2 ; 1 0 0];
                TS = inv(TT);
                for dc = 1:size(this.distanceRecorder,1)
                    clearanceD =
clearance; %+this.obstacle(this.distanceRecorder(dc,1)).ship
Size(1)/2
                    % calculate distance
                    dist = this.distanceRecorder(dc,[6 4
2])*this.wpBoxSize;
                    a = round(TS*dist',1);
                    if a(3)==0
                        tc = (clearanceD-a(1))/a(2);
                        if tc >= 0
                            dmin = 0;
                        else
                            dmin = Inf;
                        end
                    elseif a(2)==0
                        dmin = a(1);
```

```
                    else
                        dmin = a(1) + a(2)*(-a(2)/a(3)) +
a(3)*((-a(2)/a(3))^2);
                    end

                    % if minimum point less than clearance
                    if dmin <= clearanceD

                        % % % FUZZY
                        % if distance > calculation range &
outside FoV, ignore it,
                        % else do fuzzy.
                        if ( d <= rangeNormalize && h <=
0.5*fov && h >= -(0.5*fov))
                            dReal = d*this.wpBoxSize;
                            %hDeg = rad2deg(h);
                            fisResult =
evalfis(fuzzyObj,[dReal h]);
                            % penghindaran
                            this.currentCoordinate(3) =
this.currentCoordinate(3) + fisResult;
                            noObstacleNearby = false;
                        end % FUZZY

                    end
                end
            end

            if(noObstacleNearby)
                if ~isempty(this.waypoint)
                    deltaPoint = this.waypoint(1,1:2) -
this.currentCoordinate(1:2);
                    deltaDeg =
atan2(deltaPoint(2),deltaPoint(1));
                    this.currentCoordinate(3) = deltaDeg +
(this.currentCoordinate(3) - deltaDeg)*0.8;
                end
            end

            % plot trajectory
            plot(this.currentCoordinate(1),
this.currentCoordinate(2), 'b.')

        end

    end
end
```

c. Program Simulasi Lintasan USV

```
   addpath('./fn/');
clear all;
clc; clf;

% DEKLARASI UKURAN PLANE
CELLSIZE = 30; % 1 arc second

% DEKLARASI SAMPLING TIME
samplingTime = 0.5;

% DEKLARASI GLOBAL MAPS
MAPS = zeros(10,80);

% DEKLARASI POSISI OBSTACLE
MAPS(5:7,20)=1; MAPS(7,21)=1;
MAPS(9,40:41)=1;
MAPS(3,40)=1; MAPS(3:5,41)=1;
MAPS(6:8,60:61) = 1;

MAPS(1:8,26)=1;
MAPS(8,26:31)=1;
MAPS(3:10,34)=1;
MAPS(6,26)=0;

MAPS(6:7,10)=1;

% DEKLARASI POSISI WAYPOINT
Waypoint = [1.7 6.2; 40.3 6.7];
% Waypoint = [1.7 6.2; 40.3 6.7; 24.1 9.8; 17.1
7.3; 1.8 10.1];
% Waypoint = [28 11; 28 6; 39 6; 39 1; 28 1; 28
10];

% FIND NODE, OBSTACLE
node =
[Waypoint;unique(findNode(MAPS,Waypoint),'rows')
];
```

```
obstacle = findObstacle(MAPS,Waypoint);

[nConnection,nDistance] = getConnection( node ,
obstacle , Waypoint );

for nci = 1:size(nConnection,1)
    for ncj = 1:size(nConnection,2)
        if nConnection(nci,ncj) == 1
            % plot
%             drawPlot([node(nci,1)
node(ncj,1)] , [node(nci,2) node(ncj,2)] ,
'b.:');
        end
    end
end



for wpos = 1 : size(Waypoint,1)-1
%     wpos
    dj = Djikstra(nDistance,wpos,wpos+1);
    cost = dj.cost;

    % select node that used in path.
    noderoute = node(dj.route,:);
    noderoute = flip(noderoute,1);

    if wpos==1
        new_WP = noderoute;
    elseif wpos<size(Waypoint,1)
        new_WP = [new_WP ; noderoute(2:end,:)];
    end

end

% dubins calculation, output = x,y,speed
dbn = dubins(MAPS,new_WP(:,1:2),1,deg2rad(0));

figure(1);
```

```
% DRAW MAP
drawData(1,MAPS,Waypoint)
hold on;

% PLOT WAYPOINT WITHOUT DUBINS
% drawPlot(new_WP(:,1),new_WP(:,2),'r.-');

% PLOT WAYPOINT WITH DUBINS
drawPlot(dbn(:,1),dbn(:,2),'b:');

usv = USV([12 7],dbn,CELLSIZE,'USV','red');

% ADD DYNAMIC OBSTACLE
% ob1 = dubins([],[6 2;5 7;11
11],0.2,deg2rad(0));
% usv.addObstacle(ShipV3([24
12],ob1,CELLSIZE,'ship','Obstacle','red'));

% ob1 = dubins([],[5 3; 5 7; 10
10],0.2,deg2rad(0));
% usv.addObstacle(ShipV3([24
12],ob1,CELLSIZE,'ship','Obstacle','red'));
%
% ob1 = dubins([],[4 4; 5 7; 9
9],0.2,deg2rad(0));
% usv.addObstacle(ShipV3([24
12],ob1,CELLSIZE,'ship','Obstacle','red'));

% ob1 = dubins([],[8.5 10; 3.5 1.5; 12.5
0.5],0.2,deg2rad(0));
% usv.addObstacle(ShipV3([24
12],ob1,CELLSIZE,'ship','Obstacle','red'));
%
% ob1 = dubins([],[12.5 0.5; 5.5 1.5; 8.5
10],0.2,deg2rad(0));
% usv.addObstacle(ShipV3([24
12],ob1,CELLSIZE,'ship','Obstacle','blue'));
```

```
% ob1 = dubins([],[14 11; 21 11; 24 4; 21
1],0.2,deg2rad(0));
% usv.addObstacle(ShipV3([24
12],ob1,CELLSIZE,'ship','Obstacle','red'));

% ob1 = dubins([],[20 12; 42 11; 42 8; 34 2; 32
8],0.2,deg2rad(0));
% usv.addObstacle(ShipV3([24
12],ob1,CELLSIZE,'ship','Obstacle','red'));

% ADD STATIC OBSTACLE
% for ob = 1:size(obstacle,1)
%     usv.addObstacle(Ship([24
24],obstacle(ob,:),CELLSIZE,'box','','red'));
% end

% ADD CUSTOM STATIC OBSTACLE
% usv.addObstacle(ShipV3([20 20],[12.5
4.5],CELLSIZE,'box','','red'));
% usv.addObstacle(ShipV3([20 20],[11.5
4],CELLSIZE,'box','','red'));
% usv.addObstacle(ShipV3([20 20],[13.5
4.5],CELLSIZE,'box','','red'));
% % usv.addObstacle(ShipV3([20 20],[14.5
4.5],CELLSIZE,'box','','red'));
% usv.addObstacle(ShipV3([20 20],[16.5
4],CELLSIZE,'box','','red'));
% usv.addObstacle(ShipV3([20 20],[15.5
4.5],CELLSIZE,'box','','red'));
% usv.addObstacle(ShipV3([20 20],[17.5
4.5],CELLSIZE,'box','','red'));
% usv.addObstacle(ShipV3([20 20],[8
1.5],CELLSIZE,'box','','red'));
% usv.addObstacle(ShipV3([20 20],[6.5
3.5],CELLSIZE,'box','','red'));
% usv.addObstacle(ShipV3([20 20],[9.5
4],CELLSIZE,'box','','red'));
```

```
% usv.addObstacle(ShipV3([20 20],[18.5
3.5],CELLSIZE,'box','','red'));
% usv.addObstacle(ShipV3([20 20],[18.5
4],CELLSIZE,'box','','red'));
% usv.addObstacle(ShipV3([20 20],[18.5
2.5],CELLSIZE,'box','','red'));
% VIDEO SAVING CLASS
mv = VideoWriter('statisDinamis.avi');
mv.FrameRate = 10;
open(mv);

fisObj = readfis('steering2');
dist2obst = [];

while(~usv.isFinish)
    usv.move(samplingTime);

usv.distanceDetection(samplingTime,80,60,deg2rad
(184),fisObj);

    dist2obst =
usv.calculateAllObstacleDistance(dist2obst);

    % SET FIGURE 1
    set( ...
        gca, ... % get current axis
        'XTick', floor(min(dbn(:,1)))-
2+1.5:floor(max(dbn(:,1)))+2.5, ...
        'YTick', floor(min(dbn(:,2)))-
2+0.5:floor(max(dbn(:,2)))+2.5, ...
        'XLim', [ min(dbn(:,1))-2.5
max(dbn(:,1))+2.5], ... % set max X
        'YLim', [ min(dbn(:,2))-2.5
max(dbn(:,2))+2.5], ... % set max Y
        'GridLineStyle', '-', ... % set grid
style
        'XGrid', 'on', ... % enable grid
        'YGrid', 'on', ...
        'Ydir', 'normal' ...
```

```
    );

    drawnow;
    writeVideo(mv,getframe(gcf));
end


close(mv);

figure(2); hold on;
usv.plotAllObstacleDistance(dist2obst,
samplingTime)

disp('>> DONE <<')
```

Dubins

```
function result = dubins(ObstacleMap, WP,
softness, initialHeadingDegree)
% convert waypoint to points and heading degree,
feed it to dubins_curve.m
% params : Waypoint[x y], radius, softness

%     radiusSelector = [80/24 9; 64/24 8; 49/24
7; 35/24 6; 24/24 5; 14/24 4; 7/24 3];
    radiusSelector = [ 35/24 6; 24/24 5; 14/24
4; 7/24 3];
    result = [];

    %initial heading direction
    WP(1,3) = initialHeadingDegree;

    for i=1:size(WP,1)-1

        %find heading direction
        WP(i+1,3) = atan2(WP(i+1,2)-
WP(i,2),WP(i+1,1)-WP(i,1));
        calculating = true;
```

```
        selector = 1;

        while(calculating)

            calculating = false;

            % calculate dubin
            dubin =
dubins_curve(WP(i,:),WP(i+1,:),radiusSelector(se
lector,1),softness,false);


            if(length(ObstacleMap)>0)
                % for each dubin wp
                for d=1:size(dubin,1)
                    % get 3x3 obstacle from
dubins point to nearby matrix
                    nearx = round(dubin(d,1));
                    if nearx >=
size(ObstacleMap,2)
                        nearx =
size(ObstacleMap,2)-1;
                    elseif nearx < 2
                        nearx = 2;
                    end
                    neary = round(dubin(d,2));
                    if neary >=
size(ObstacleMap,1)
                        neary =
size(ObstacleMap,1)-1;
                    elseif neary < 2
                        neary = 2;
                    end

    %                 [nearx neary];
    %                 OM = ObstacleMap(neary-
1:neary+1,nearx-1:nearx+1)
```

```
                    for checkx=nearx-1:nearx+1
                        for checky=neary-
1:neary+1
    %                           [checkx checky
ObstacleMap(checkx,checky)]
                                if
ObstacleMap(checky,checkx)==1
                                    % calculate
distance
                                    if norm([checkx-
dubin(d,1) checky-dubin(d,2)]) < 0.9
                                        % if inside
collision area
                                        % hitung
lagi dengan radius yang berubah
                                        calculating
= true;
                                        selector =
selector+1; % ubah selector radius
                                        break;
                                    end
                                end
                            end
                            if calculating
                                break;
                            end
                        end
                        if calculating
                            break;
                        end
                    end
                end
            end


            if selector>size(radiusSelector,1)
                % if collision exist in all
speed
                selector =
size(radiusSelector,1);
```

```
                calculating = false;
                disp('COLLISION DETECTED AT:')
                disp(dubin(d,1:2))
            end
        end

        index = 1;
        lastWP = index;
        dub = dubin(index,:);
        while(index<size(dubin,1)-1)
            index = index+1;
            if abs(dubin(lastWP,3)-
dubin(index,3)) > deg2rad(1)
                dub = [ dub ; dubin(index,:) ];
                lastWP = index;
            end
        end
        result = [ result ; dub
ones(size(dub,1),1)*radiusSelector(selector,2)];
    end
    result(end+1,:) = [ WP(size(WP,1),1:2)
dubin(end,3) radiusSelector(selector,2)];
    result = result(:,[1 2 4 3]);
    result(1,3) = 0;
    result(end,3) = 1;
end
```

Halaman ini sengaja dikosongkan

BIODATA PENULIS

Naufan Hafiyyan, lahir di Semarang pada tanggal 15 Maret 2000. Penulis merupakan putra kedua dari dari dua bersaudara. Penulis menyelesaikan pendidikan di SD Muhammadiyah Program Khusus pada tahun 2011. Kemudian melanjutkan pendidikan di SMPN 9 Surakarta pada tahun 2011. Dan pada tahun 2013 penulis melanjutkan pendidikan di SMAN 1 Surakarta. Pada tahun 2016 penulis memulai jenjang S1 Departemen Teknik Elektro, di Institut Teknologi Sepuluh Nopember (ITS), Surabaya, dengan mengambil Program Studi Teknik Sistem Pengaturan. Selama kuliah, penulis aktif mengikuti kegiatan dari Laboraturium Sistem dan Sibernetika. Untuk menghubungi penulis, dapat melalui surat elektronik pada alamat email naufanhaf@gmail.com

Halaman ini sengaja dikosongkan