



TESIS - EE185401

***TRAJECTORY TRACKING MULTI-ROBOT
MENGUNAKAN PENDEKATAN PENYELESAIAN
PERMASALAHAN SINGULARITY***

ANGGITASARI PUTRI WIDANIS
07111750022001

DOSEN PEMBIMBING
Dr. Trihastuti Agustinah, ST., MT.
Dr. Ir. Ari Santoso, DEA.

PROGRAM MAGISTER
BIDANG KEAHLIAN TEKNIK SISTEM PENGATURAN
DEPARTEMEN TEKNIK ELEKTRO
FAKULTAS TEKNOLOGI ELEKTRO DAN INFORMATIKA CERDAS
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA
2020



TESIS - EE185401

***TRAJECTORY TRACKING MULTI-ROBOT
MENGUNAKAN PENDEKATAN PENYELESAIAN
PERMASALAHAN SINGULARITY***

ANGGITASARI PUTRI WIDANIS
07111750022001

DOSEN PEMBIMBING
Dr. Trihastuti Agustinah, ST., MT.
Dr. Ir. Ari Santoso, DEA.

PROGRAM MAGISTER
BIDANG KEAHLIAN TEKNIK SISTEM PENGATURAN
DEPARTEMEN TEKNIK ELEKTRO
FAKULTAS TEKNOLOGI ELEKTRO DAN INFORMATIKA CERDAS
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA
2020

LEMBAR PENGESAHAN TESIS

Tesis disusun untuk memenuhi salah satu syarat memperoleh gelar

Magister Teknik (MT)

di

Institut Teknologi Sepuluh Nopember

Oleh:

ANGGITASARI PUTRI WIDANIS

NRP: 07111750022001

Tanggal Ujian: 3 Januari 2020

Periode Wisuda: Maret 2020

Disetujui oleh:

Pembimbing:

1. Dr. Trihastuti Agustinah, ST., MT.
NIP: 196808121994032001



2. Dr. Ir. Ari Santoso, DEA.
NIP: 196602181991021001

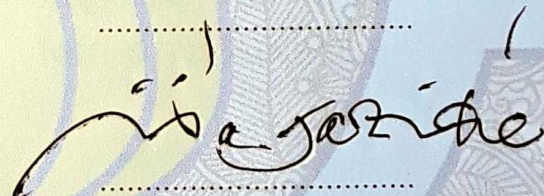


Penguji:


1. Prof.Dr.Ir. Mohammad Nuh, DEA.
NIP: 195906171984031002



2. Prof.Dr.Ir. Achmad Jazidie, M.Eng.
NIP: 195902191986101001



**Kepala Departemen Teknik Elektro
Fakultas Teknologi Elektro dan Informatika Cerdas**



Dedet Candra Riawan, ST., M.Eng., Ph.D.
NIP: 197311192000031001

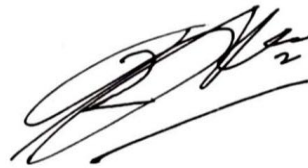
Halaman ini sengaja dikosongkan

PERNYATAAN KEASLIAN TESIS

Dengan ini saya menyatakan bahwa isi keseluruhan Tesis saya dengan judul “**TRAJECTORY TRACKING MULTI-ROBOT MENGGUNAKAN PENDEKATAN PENYELESAIN PERMASALAHAN SINGULARITY**” adalah benar-benar hasil karya intelektual mandiri, diselesaikan tanpa menggunakan bahan-bahan yang tidak diijinkan dan bukan merupakan karya pihak lain yang saya akui sebagai karya sendiri.

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

Surabaya, 3 Januari 2020



Anggitasari Putri Widanis

NRP. 07111750022001

Halaman ini sengaja dikosongkan

TRAJECTORY TRACKING MULTI-ROBOT MENGGUNAKAN PENDEKATAN PENYELESAIAN PERMASALAHAN SINGULARITY

Nama mahasiswa : Anggitasari Putri Widanis
NRP : 07111750022001
Pembimbing : 1. Dr. Trihastuti Agustinah, ST., MT.
2. Dr. Ir. Ari Santoso, DEA.

ABSTRAK

Singularity dapat menyebabkan kontrol *cluster space* pada *multi-robot* tidak dapat bekerja karena adanya tabrakan antar robot. Ada beberapa solusi yang ditawarkan untuk menyelesaikan *singularity* yaitu dengan metode *coupling* (berdasar *angular velocity* roda kanan dan kiri) dan metode kontrol *escaping singularity* (berdasar *angular velocity* 1 roda). Kontrol *escaping singularity* lebih sederhana dari pada metode *coupling* dalam mengatasi *singularity*, karena menggunakan 1 roda pada *angular velocity*. Pada penelitian ini kontrol *escaping singularity* digunakan untuk mengendalikan *multi-robot* pada *cluster space*. *Multi-robot* terdiri dari 1 *leader* dan 2 *followers* (tidak rigid). Robot *leader* mengikuti *trajectory*, sedangkan robot *followers* mengikuti *leader* menggunakan *cluster space* untuk menjaga jarak antar robot. Hasil simulasi menunjukkan bahwa *multi-robot* pada *cluster space* dengan kontrol *escaping singularity* dapat menghindari tabrakan antar robot (*singularity*), meskipun terdapat *disturbance* pada robot *leader*.

Kata kunci: *Trajectory Tracking, Multi-Robot, Singularity, Collision Avoidance, Cluster Space, Kontrol Escaping Singularity.*

Halaman ini sengaja dikosongkan

MULTI-ROBOT TRAJECTORY TRACKING USING THE SINGULARITY APPROACH

By : Anggitasari Putri Widanis
Student Identity Number : 07111750022001
Supervisory : 1. Dr. Trihastuti Agustinah, ST., MT.
2. Dr. Ir. Ari Santoso, DEA.

ABSTRACT

Singularity may cause stoppages in the cluster space controls in a multi-robot system due to collisions between the constituent robots. Solutions to the singularity problem include coupling, using the angular velocity of the right and left wheels, and escaping singularity, using the angular velocity of only one wheel. The escaping singularity method is simpler in implementation than the coupling method due to only requiring one wheel for angular velocity. The escaping singularity method is used to control a multi-robot system in the cluster space. The system consists of one leader and two followers (nonrigid). The leader robot follows a designated trajectory, while the follower robots follow the leader using the cluster space to keep their distance to each other. The simulation results show that the cluster space control for multi-robot system with escaping singularity is able to prevent collisions between robots and incur singularities even it applied disturbances in the leader robot.

Keywords: Trajectory Tracking, Multi-Robot, Singularity, Collision Avoidance, Cluster Space, Escaping Singularity Controller.

Halaman ini sengaja dikosongkan

KATA PENGANTAR

Alhamdulillah, Alhamdulillah dan Alhamdulillah penulis panjatkan kehadiran Allah SWT yang telah melimpahkan segala rahmatNya sehingga penulis dapat menyelesaikan tesis dengan judul “*Trajectory Tracking Multi-Robot Menggunakan Pendekatan Penyelesaian Permasalahan Singularity*” guna memenuhi sebagian persyaratan untuk memperoleh gelar Magister Teknik program studi Teknik Sistem Pengaturan pada Fakultas Teknik Elektro Institut Teknologi Sepuluh Nopember.

Penulis menyadari kelemahan serta keterbatasan yang ada sehingga dalam menyelesaikan tesis ini memperoleh bantuan dari berbagai pihak, dalam kesempatan ini penulis menyampaikan ucapan terimakasih kepada:

1. Kedua orang tua, ARK dan keluarga yang selalu memberi semangat, dukungan, kepercayaan, dan doa-doanya.
2. Ibu Dr. Trihastuti Agustinah, ST., MT dan bapak Dr.Ir. Ari Santoso, DEA atas kepercayaan, waktu, bimbingan, didikan, dan motivasi pada penulis hingga terselesaikannya tesis ini.
3. Para dosen penguji tesis atas saran dan nasihat yang telah diberikan serta dosen-dosen Teknik Sistem Pengaturan atas ilmu bermanfaat yang telah diberikan.
4. Untuk semua pihak yang tidak dapat disebutkan satu persatu yang telah membantu, memberi semangat, dukungan, dan doa-doanya, semoga senantiasa diberikan kelancaran bagi setiap urusanya.

Penulis menyadari bahwa tesis ini masih banyak kekurangan baik isi maupun susunannya. Semoga tesis ini dapat bermanfaat tidak hanya bagi penulis juga bagi para pembaca. “Khusnudzon terhadap Allah SWT”

Surabaya, 3 Januari 2020

Penulis

Halaman ini sengaja dikosongkan

DAFTAR ISI

LEMBAR PENGESAHAN TESIS	iii
PERNYATAAN KEASLIAN TESIS	v
ABSTRAK	vii
ABSTRACT	ix
KATA PENGANTAR	xi
DAFTAR ISI	xiii
DAFTAR TABEL	xvii
BAB 1 PENDAHULUAN	1
1.1 Latar Belakang	1
1.2 Rumusan Masalah	2
1.3 Tujuan	2
1.4 Batasan Masalah	3
1.5 Kontribusi	3
BAB 2 KAJIAN PUSTAKA	5
2.1 Kajian Penelitian Terkait	5
2.1.1 <i>Adaptive Sliding-Mode Cluster Space Control of a Non-holonomic Multi-Robot System with Applications</i> [1]	5
2.1.2 <i>Error Characterization in the Vicinity of Singularities in Multi-Robot Cluster Space Control</i> [2]	8
2.1.3 <i>Solving The Singularity Problem For a Holonomic Mobile Robot</i> [3]	12
2.2 Dasar Teori	13
2.2.1 <i>Model Wheeled Mobile Robot</i>	13
2.2.2 <i>Model Kinematics dan Dynamics pada Non-holonomic Mobile Robot</i>	15
2.2.3 <i>Cluster Control</i>	17
2.2.4 <i>Kontrol Cluster Space</i>	18
2.2.5 <i>Robot space</i>	19
2.2.6 <i>Cluster Space</i>	19
2.2.7 <i>Singularity</i>	20
BAB 3 METODE PENELITIAN	21

3.1	Pemodelan Robot <i>Kinematics</i>	21
3.2	Pemodelan Variabel Robot dan <i>Cluster</i>	22
3.3	Pemodelan Robot <i>Dynamics</i>	25
3.4	Pemodelan <i>Dynamics</i> pada <i>Cluster Space</i>	26
3.5	Pemodelan <i>Tracking Error Cluster Space</i>	28
3.6	<i>Cluster Singularity</i>	29
3.7	Kontrol <i>Escaping Singularity</i>	31
3.8	Blok Diagram Perancangan Kontrol	32
3.9	<i>Flowchart</i> Perancangan	32
BAB 4 HASIL DAN PEMBAHASAN		35
4.1	Pengujian Pembentukan <i>Reference Trajectory</i>	36
4.2	Pengujian <i>Plotting</i> Posisi Awal Robot	36
4.3	Pengujian <i>Plotting</i> Posisi Awal Robot dan <i>Reference Trajectory</i>	37
4.4	Pengujian Simulasi Robot tanpa <i>Collision Avoidance</i>	38
4.5	Pengujian Simulasi Robot Menggunakan <i>Collision Avoidance</i>	41
4.6	Pengujian Simulasi Robot Menggunakan <i>Collision Avoidance</i> dengan Parameter $\beta = 90^\circ$ & $p = 10$	44
4.7	Pengujian Simulasi Robot Menggunakan <i>Collision Avoidance</i> dengan Parameter $\beta = 90^\circ$ & $p = 8$	47
4.8	Pengujian Simulasi Robot Menggunakan <i>Collision Avoidance</i> dengan <i>Disturbance</i> $\beta = 60^\circ$	51
BAB 5 PENUTUP		57
5.1	Kesimpulan	57
5.2	Saran	57
DAFTAR PUSTAKA		59
LAMPIRAN		61
BIOGRAFI PENULIS		65

DAFTAR GAMBAR

Gambar 2.1 <i>Adaptive SMC Diagram</i>	7
Gambar 2.2 <i>Adaptive SMC Control Tracking</i>	7
Gambar 2.3 <i>Comparasion</i> antara SMC dan <i>Adaptive SMC</i> dengan Masukan <i>Disturbance</i> setelah 20s	8
Gambar 2.4 Respon Parameter <i>Adaptive</i>	8
Gambar 2.5 Arsitektur Kontrol <i>Cluster Space</i> untuk Sistem 3-Robot.....	9
Gambar 2.6 Variasi dari Parameter <i>Cluster</i> p, dengan q=2 dan $\beta=60^\circ$	10
Gambar 2.7 Variasi dari Parameter <i>Cluster</i> p, dengan q=2 dan $\beta=15^\circ$	10
Gambar 2.8 Variasi dari Parameter <i>Cluster</i> β , dengan p=q=5	11
Gambar 2.9 Variasi dari Parameter <i>Cluster</i> β , dengan p=q=2	11
Gambar 2.10 <i>Escaping Singularity Block Diagram</i>	12
Gambar 2.11 Hasil <i>Escaping Singularity</i>	13
Gambar 2.12 <i>Coordinate WMR</i>	14
Gambar 2.13 <i>Differential Drive Mobile Robot</i>	16
Gambar 2.14 <i>Velocity</i> dari Pusat Robot	16
Gambar 2.15 <i>Cluster Definition</i>	18
Gambar 2.16 <i>Cluster Controller for n Robots</i>	18
Gambar 2.17 <i>2-Link Near Origin</i>	20
Gambar 3.1 Model WMR	22
Gambar 3.2 <i>Cluster Space and Robot Space Variables</i>	25
Gambar 3.3 Blok Diagram Perancangan Kontrol	31
Gambar 3.4 <i>Flowchat</i> Perancangan	33
Gambar 4.1 Blok Pemodelan Kontrol	35
Gambar 4.2 <i>Plotting Reference Trajectory</i>	36
Gambar 4.3 Inisialisasi Posisi Awal Robot.....	37
Gambar 4.4 <i>Plotting</i> Posisi Awal Robot dan <i>Reference Trajectory</i>	37
Gambar 4.5 Simulasi Terjadinya Tabrakan antara Robot 1 dan Robot	38
Gambar 4.6 <i>Tracking Error Multi-Robot</i>	39
Gambar 4.7 Perubahan Sudut Orientasi <i>Multi-Robot</i>	39
Gambar 4.8 Perubahan <i>Linear Velocity Multi-Robot</i>	40
Gambar 4.9 Perubahan <i>Angular Velocity Multi-Robot</i>	40
Gambar 4.10 Simulasi Potensi Terjadinya Tabrakan antara Robot 1 dan Robot	341
Gambar 4.11 Simulasi Robot 1 Menghindari Tabrakan dengan Robot 3	41
Gambar 4.12 Simulasi <i>Collision Avoidance</i> Robot 1 dan Robot 3	42
Gambar 4.13 <i>Tracking Error Multi-Robot</i>	42
Gambar 4.14 Perubahan Sudut Orientasi <i>Multi-Robot</i>	43
Gambar 4.15 Perubahan <i>Linear Velocity Multi-Robot</i>	43
Gambar 4.16 Perubahan <i>Angular Velocity Multi-Robot</i>	44
Gambar 4.17 Simulasi <i>Multi-Robot</i> $\beta=90^\circ$ & p=10	45

Gambar 4.18 <i>Tracking Error Multi-Robot</i>	45
Gambar 4.19 Perubahan Sudut Orientasi <i>Multi-Robot</i>	46
Gambar 4.20 Perubahan <i>Linear Velocity Multi-Robot</i>	46
Gambar 4.21 Perubahan <i>Angular Velocity Multi-Robot</i>	47
Gambar 4.22 Simulasi <i>Multi-Robot</i> $\beta=90^\circ$ & $p=8$	48
Gambar 4.23 <i>Tracking Error Multi-Robot</i>	48
Gambar 4.24 Perubahan Sudut Orientasi <i>Multi-Robot</i>	49
Gambar 4.25 Perubahan <i>Linear Velocity Multi-Robot</i>	49
Gambar 4.26 Perubahan <i>Angular Velocity Multi-Robot</i>	50
Gambar 4.27 Rata-Rata <i>Tracking Error</i>	51
Gambar 4.28 Ketika diberi <i>Disturbance</i>	51
Gambar 4.29 <i>Tracking Error</i> tanpa <i>Disturbance</i>	52
Gambar 4.30 <i>Tracking Error</i> dengan <i>Disturbance</i>	52
Gambar 4.31 <i>Linear Velocity</i> tanpa <i>Disturbance</i>	53
Gambar 4.32 <i>Linear Velocity</i> dengan <i>Disturbance</i>	53
Gambar 4.33 <i>Angular Velocity</i> tanpa <i>Disturbance</i>	54
Gambar 4.34 <i>Angular Velocity</i> dengan <i>Disturbance</i>	54

DAFTAR TABEL

Tabel 2.1 Variabel <i>Robot Space</i>	19
Tabel 2.2 Variabel <i>Cluster Space</i>	19
Tabel 4.1 Rata-Rata <i>Tracking Error</i>	50
Tabel 4.2 Perbandingan <i>Tracking Error</i> tanpa <i>Disturbance</i> dan dengan <i>Disturbance</i> $\beta=60^\circ$	55

Halaman ini sengaja dikosongkan

BAB 1

PENDAHULUAN

1.1 Latar Belakang

Mobile robot merupakan salah satu jenis robot yang mampu melakukan perpindahan dari satu tempat ketempat lain menggunakan roda, sehingga diperlukan sistem kontrol yang berfungsi untuk mengatur *velocity* pada setiap roda agar mencapai posisi yang diinginkan. Salah satu perkembangan dari *mobile robot* saat ini adalah sistem *multi-robot*, dimana terdapat lebih dari satu *mobile robot* yang saling berkaitan. Hal ini dikarenakan sistem *multi-robot* dapat meningkatkan efisiensi, keandalan, dan fleksibilitas sistem. Beberapa aplikasi sistem *multi-robot* terdapat dalam pemanfaatan robot pada kerja-kerja *surveilans*, pencarian dan penyelamatan (SAR), sistem pengamanan dan pengamatan, eksplorasi daerah tak dikenal atau berbahaya.

Dalam sistem *mobile robot*, diperlukan suatu kontrol agar tidak terjadi tabrakan antara *mobile robot* satu dengan *mobile robot* yang lainnya yang berjalan pada *trajectory* yang telah ditentukan. Terdapat beberapa penelitian yang membahas tentang kontrol *multi-robot*, salah satunya yaitu dengan metode kontrol *cluster space*, dimana *multi-robot* dianggap sebagai satu kesatuan yang disebut *cluster* [1]. *Cluster* memiliki *state dynamics* yang dinamakan *cluster space* dan *cluster state* merupakan fungsi dari *state* robot yang disebut *robot space*. *Cluster* menghasilkan kontroler yang berfungsi untuk mengatur jarak antar robot. Kontrol *cluster* selanjutnya diubah dengan *inverse dynamics* dan matriks *jacobian* agar dapat digunakan sebagai masukan pada *robot space*. Sehingga masing-masing robot memiliki kontrol dari *cluster space* untuk menjaga jarak antar robot.

Pada kontrol *cluster space* dapat terjadi kondisi *singularity* yaitu ketika terjadi tabrakan antar robot, sehingga sistem kontrol tersebut tidak dapat menjalankan fungsinya karena matriks singular tidak memiliki *inverse*. Permasalahan *singularity* pada sistem *multi-robot* telah dibahas pada [2], yaitu dengan teknik *collision avoidance* untuk menghindari tabrakan antar robot,

sehingga tidak terjadi *singularity*. Teknik *collision avoidance* menggunakan nilai parameter *cluster space* yang didapat dengan sistem *trial and error*.

Permasalahan *singularity* juga dapat diselesaikan dengan metode kontrol *escaping singularity* [3] dengan mendeteksi nilai parameter *cluster space* yang menyebabkan kondisi *singularity*. Pada saat kondisi *singularity* terdeteksi, sistem kontrol *escaping singularity* mengatur *velocity* pada setiap robot sehingga dapat menghindari kondisi *singularity*. Namun, metode kontrol *escaping singularity* pada penelitian [3] menggunakan *mobile robot* jenis *holonomic*, belum dibuktikan bahwa metode tersebut dapat digunakan pada jenis *non-holonomic*. Jenis robot *non-holonomic* memiliki gerakan yang terbatas.

Berdasarkan beberapa penelitian diatas, ide yang dihasilkan untuk tesis ini adalah merancang kontrol *multi-robot* yang terdiri dari 3 robot *non-holonomic* (1 *leader* dan 2 *followers*) menggunakan metode *cluster space* yang dilengkapi metode kontrol *escaping singularity* untuk menghindari adanya tabrakan antar robot sehingga kondisi *singularity* bisa dihindari dan *multi-robot* dapat melakukan misi perjalanan dengan mengikuti *trajectory* yang berbentuk seperti angka 8.

1.2 Rumusan Masalah

Permasalahan yang akan dibahas pada penelitian ini adalah bagaimana merancang kontrol *multi-robot* yang terdiri dari 3 robot *non-holonomic* (1 *leader* dan 2 *followers*) pada *cluster space* yang mengikuti *trajectory* berbentuk angka 8 tanpa terjadi tabrakan antar robot *non-holonomic* atau *singularity* meskipun terdapat *disturbance* pada *leader*.

1.3 Tujuan

Tujuan dari penelitian ini adalah menghasilkan kontroler *multi-robot* yang terdiri dari 3 robot *non-holonomic* (1 *leader* dan 2 *followers*) pada *cluster space* menggunakan kontrol *escaping singularity* agar tidak terjadi tabrakan antar robot *non-holonomic* yang mengikuti *trajectory* berbentuk angka 8 meskipun terdapat *disturbance* pada *leader*.

1.4 Batasan Masalah

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

1. Pemodelan robot tanpa slip
2. Sudut miring pada robot *leader* (β) yang digunakan adalah 60° dan 90° .

1.5 Kontribusi

Penelitian ini diharapkan memberikan kontribusi ilmiah yaitu perancangan kontroler untuk *multi-robot* (1 *leader* dan 2 *followers*) jenis *non-holonomic* pada *cluster space* yang mengikuti *trajectory* berbentuk angka 8 dengan kontrol *escaping singularity* untuk menghindari kondisi *singularity* meskipun terdapat *disturbance* pada robot *leader*.

Halaman ini sengaja dikosongkan

BAB 2

KAJIAN PUSTAKA

Pada bab ini penulis akan membahas tentang kajian terhadap penelitian-penelitian terkait yang telah ada. Berbagai metode yang digunakan pada setiap kajian pustaka akan dijelaskan sehingga diperoleh ide dasar untuk penyusunan tesis. Pada bab ini juga terdapat penjelasan teori-teori dasar yang mendukung proses perancangan tesis ini.

2.1 Kajian Penelitian Terkait

2.1.1 *Adaptive Sliding-Mode Cluster Space Control of a Non-holonomic Multi-Robot System with Applications* [1]

Paper ini membahas tentang *multi-robot* yang mengikuti *trajectory* menggunakan kontroler *adaptive SMC (Sliding-Mode Control)* untuk mengatur gerak *multi-robot non-holonomic* pada *cluster space*. *Cluster* terdiri dari *cluster space dynamics*, yaitu:

$$\Lambda(c)\ddot{c} + \mu(c, \dot{c}) + p(c) + \tau_a = \beta(c)\bar{v} - \alpha^T(c)\bar{\lambda} \quad (2.1)$$

dengan

$$\begin{aligned} \Lambda(c) &= J^{-T}(r)\bar{M}(r)J^{-1}(r) \\ \mu(c, \dot{c}) &= J^{-T}(r)\bar{b}(r, \dot{r}) - \Lambda(c)\dot{J}(r, \dot{r})\dot{r} \\ p(c) &= J^{-T}(r)g(r) \\ \beta(c) &= J^{-T}(r)\bar{B}(r) \\ \alpha^T(c) &= J^{-T}(r)\bar{A}^T(r) \end{aligned}$$

dimana:

$$\bar{M}(r) : \begin{bmatrix} M_1 & 0^{3 \times 3} & \dots & 0^{3 \times 3} \\ \cdot & M_2 & 0^{3 \times 3} & \vdots \\ \cdot & \cdot & \ddots & 0^{3 \times 3} \\ \cdot & \cdot & \cdot & M_n \end{bmatrix}$$

$$\begin{aligned}
\bar{b}(r, \dot{r}) &: \begin{bmatrix} b_1(r_1, \dot{r}_1) \\ \vdots \\ b_n(r_n, \dot{r}_n) \end{bmatrix} \\
g &: 0 \\
\bar{B}(r) &: \begin{bmatrix} B_1 & 0^{3 \times 2} & \dots & 0^{3 \times 2} \\ \cdot & B_2 & 0^{3 \times 2} & \vdots \\ \cdot & \cdot & \ddots & 0^{3 \times 2} \\ \cdot & \cdot & \cdot & B_n \end{bmatrix} \\
\bar{A}^T(r) &: \begin{bmatrix} A_1^T & 0^{3 \times 1} & \dots & 0^{3 \times 1} \\ \cdot & A_2^T & 0^{3 \times 1} & \vdots \\ \cdot & \cdot & \ddots & 0^{3 \times 1} \\ \cdot & \cdot & \cdot & A_n^T \end{bmatrix} \\
\bar{\lambda} &: [\lambda_1 \dots \lambda_n]^T \\
\bar{\tau} &: [\tau_1 \dots \tau_n]^T
\end{aligned}$$

dan *robot space*, yaitu transformasi dari signal *error cluster space* yang dikalikan dengan *inverse* matriks Jacobian sebagai berikut:

$$\bar{\Delta r} = J^{-1} \Delta c \quad (2.2)$$

dengan

$$\begin{aligned}
\bar{\Delta r} &= [\Delta x_1, \Delta y_1, \Delta \theta_1, \Delta x_2, \Delta y_2, \Delta \theta_2]^T \\
\bar{\Delta r}_i &= [\Delta x_i, \Delta y_i, \Delta \theta_i]^T
\end{aligned}$$

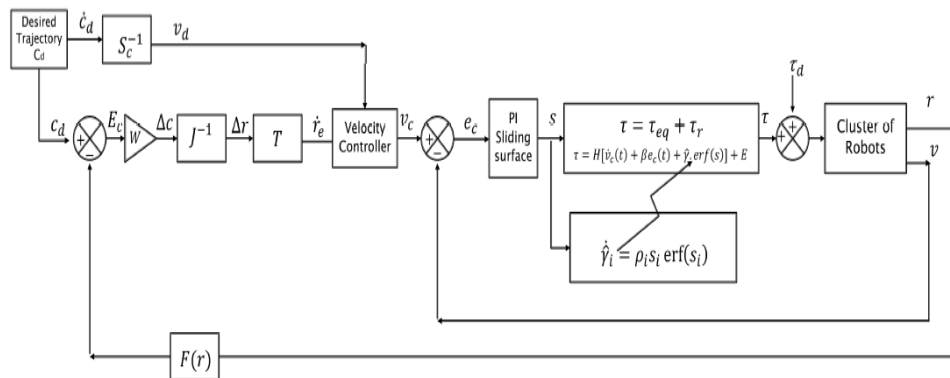
dimana:

Δx : Selisih antara nilai *x actual* dan *reference*

Δy : Selisih antara nilai *y actual* dan *reference*

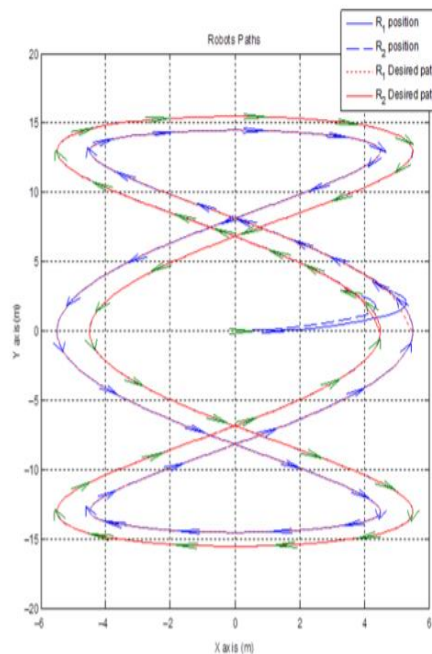
$\Delta \theta$: Selisih antara nilai θ *actual* dan *reference*

Desain kontroler pada *adaptive SMC* digunakan untuk mengatur *cluster* agar dapat mengikuti *trajectory* yang diinginkan. Blok diagram kontrol *adaptive SMC* dapat dilihat pada Gambar 2.1.

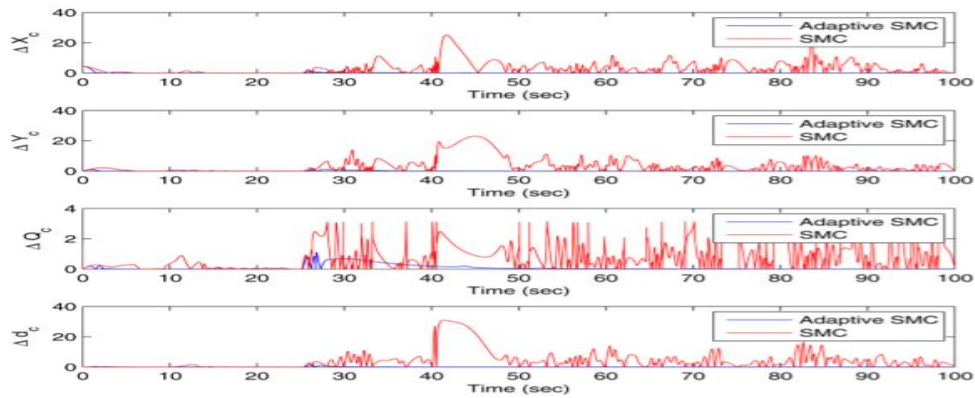


Gambar 2.1 Adaptive SMC Diagram

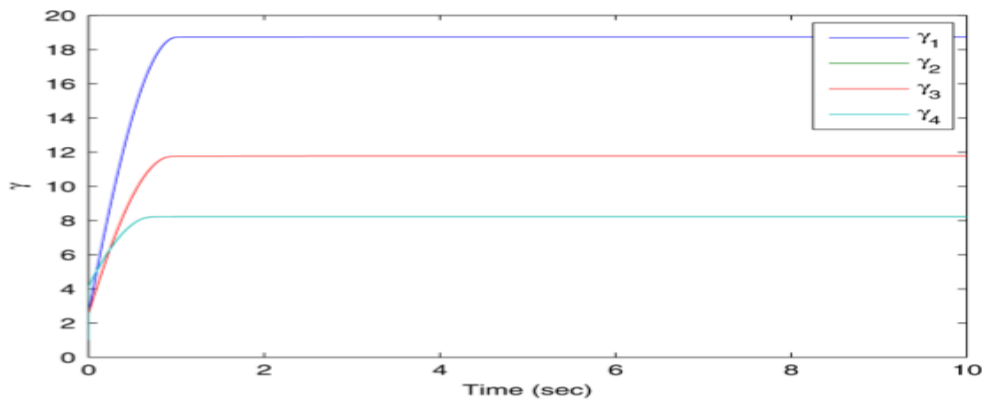
Hasil simulasi pada Gambar 2.2 menunjukkan bahwa *multi-robot* (2 robot) dapat mengikuti *trajectory* fungsi sinusoidal. Pada Gambar 2.3 menunjukkan *disturbance* diberikan setelah 20 detik dan *adaptive SMC* memberikan hasil yang lebih baik dibandingkan dengan SMC. Sedangkan Gambar 2.4 menunjukkan respon parameter *adaptive*, yaitu berubah pada zona transien.



Gambar 2.2 Adaptive SMC Control Tracking



Gambar 2.3 *Comparasion* antara SMC dan *Adaptive SMC* dengan Masukan *Disturbance* setelah 20s

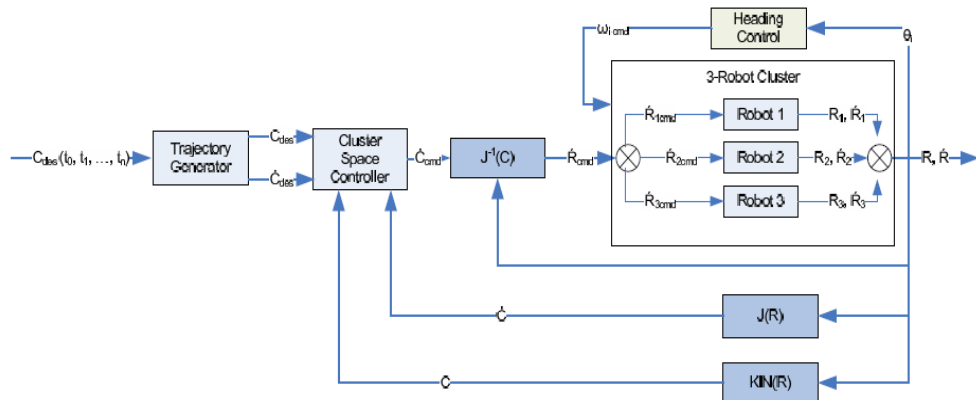


Gambar 2.4 Respon Parameter *Adaptive*

Pada hasil keseluruhan *paper* ini, kontroler *adaptive SMC* berhasil mengatur *cluster* mengikuti *trajectory* sinyal sinusoidal. *Adaptive SMC* juga *adaptive* terhadap *disturbance*. Namun, pada *paper* ini belum terdapat penyelesaian mengenai permasalahan *singularity* yang dapat terjadi pada *cluster space*.

2.1.2 *Error Characterization in the Vicinity of Singularities in Multi-Robot Cluster Space Control* [2]

Singularity dapat menyebabkan kontrol *cluster space* tidak dapat berfungsi karena matriks yang bersifat singular tidak memiliki *inverse*. Sehingga *robot space* tidak dapat terbentuk. *Paper* ini melakukan percobaan terhadap parameter variabel *cluster space* yang menyebabkan *singularity* dengan *trial and error*.

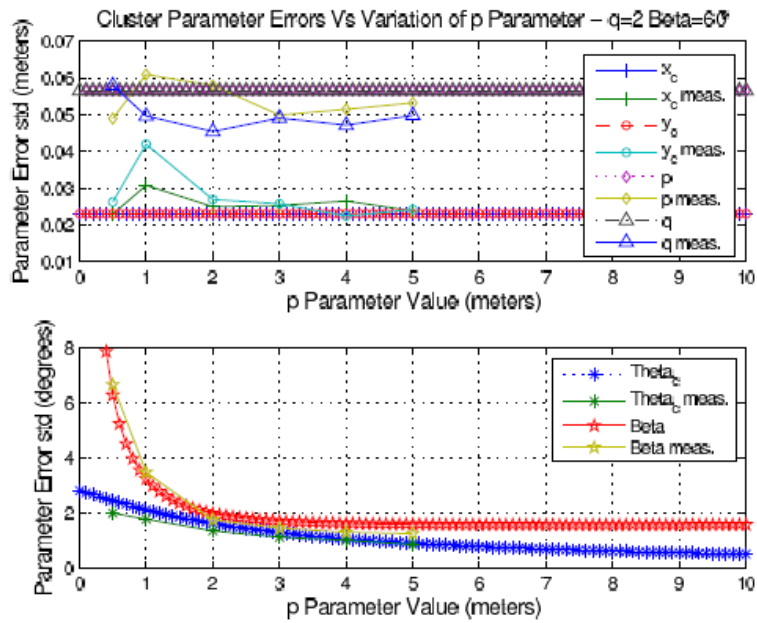


Gambar 2.5 Arsitektur Kontrol *Cluster Space* untuk Sistem 3-Robot

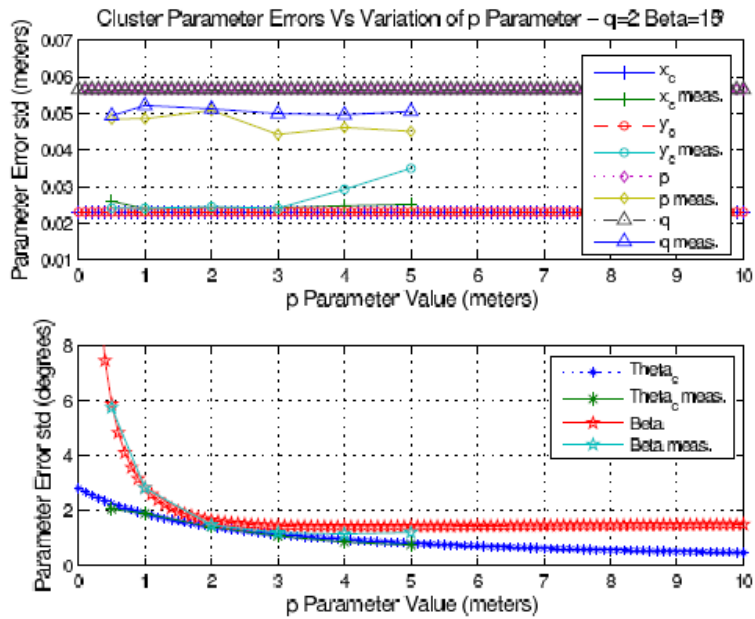
Pada Gambar 2.5 menjelaskan tentang arsitektur kontrol untuk *trajectory tracking* berdasarkan kontrol *cluster space* dari sistem 3-robot. Kontrol kecepatan untuk mengikuti *trajectory* yang diinginkan menggunakan kontrol *cluster space*, yang mana kecepatan tersebut menjadi kecepatan masing-masing robot melalui *inverse Jacobian*. Data dari robot dikonversi menjadi informasi *cluster space* melalui *forward kinematics* dan Jacobian dan *feedback* ke kontroler. *State robot* memiliki 9-variabel (3-robot dengan 3-DOF per-robot) digambarkan menjadi 9-variabel *cluster space* untuk 9-*cluster* DOF.

Kendala *non-holonomic* yang diberikan oleh gerak diferensial robot secara efektif mengurangi jumlah *cluster* yang ditetapkan secara independen menjadi enam variabel. Sehingga pada *paper* ini hanya ada 6-variabel *state* $x_c, y_c, \theta_c, p, q, \beta$, yang dianalisa 3-variabel sisa ϕ_1, ϕ_2, ϕ_3 tidak dianalisa karena *uncertainties* dari parameter tersebut.

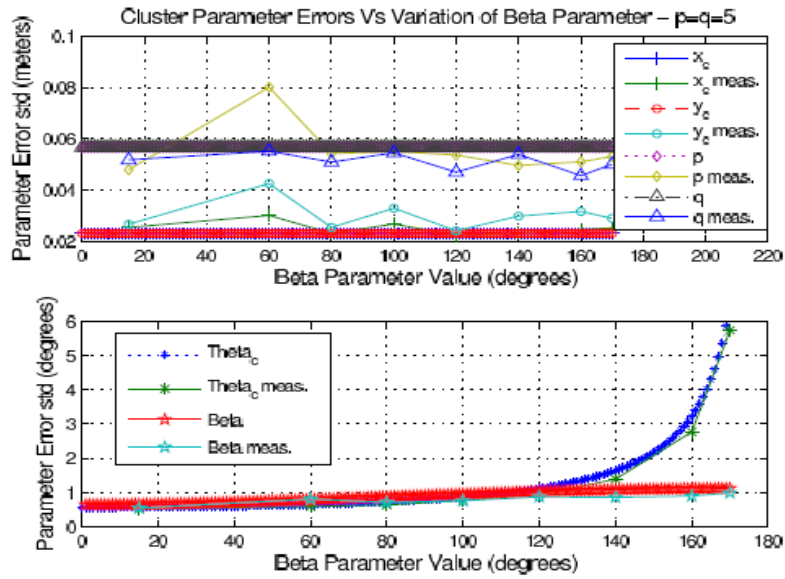
Simulasi dilakukan dengan memposisikan 3-robot pada konfigurasi *cluster* yang diinginkan dan *gain* pada kontroler *closed-loop* diatur menjadi *zero* untuk mencegah robot dari perpindahan. Hasil simulasi pada Gambar 2.6, 2.7, 2.8 dan 2.9 menunjukkan nilai parameter variabel *cluster space* yang tidak menyebabkan *singularity*.



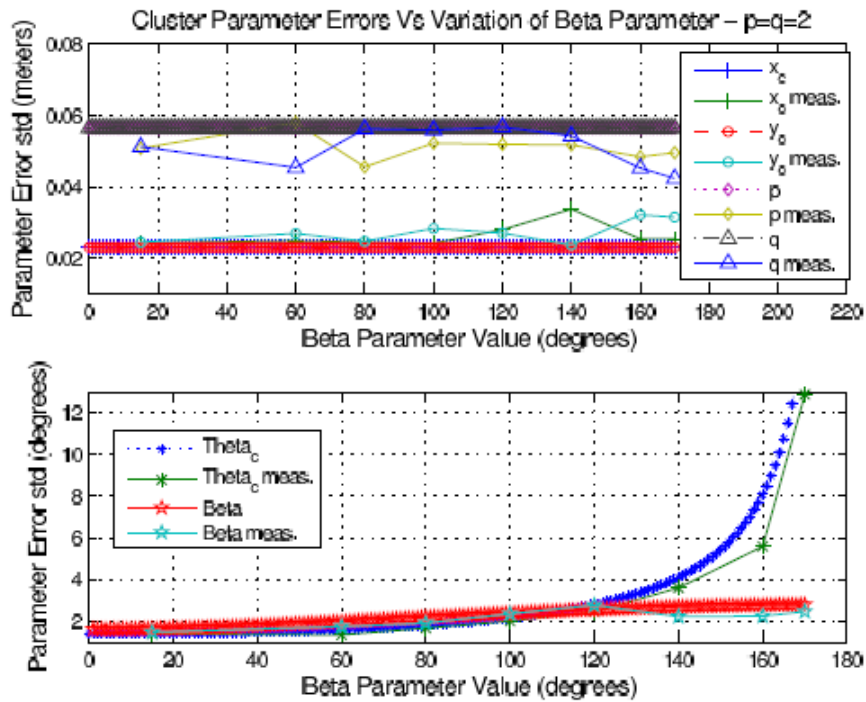
Gambar 2.6 Variasi dari Parameter *Cluster* p, dengan q=2 dan $\beta=60^\circ$



Gambar 2.7 Variasi dari Parameter *Cluster* p, dengan q=2 dan $\beta=15^\circ$



Gambar 2.8 Variasi dari Parameter *Cluster* β , dengan $p=q=5$



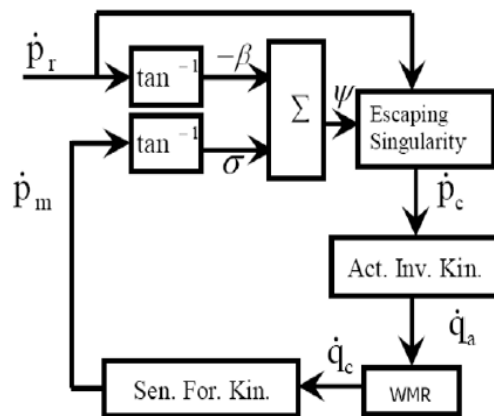
Gambar 2.9 Variasi dari Parameter *Cluster* β , dengan $p=q=2$

Paper telah menyebutkan parameter variabel *cluster space* yang tidak menyebabkan *singularity*. Namun, parameter tersebut masih ditentukan dengan *trial and error* sehingga tidak ada analisa untuk nilai parameter yang lain.

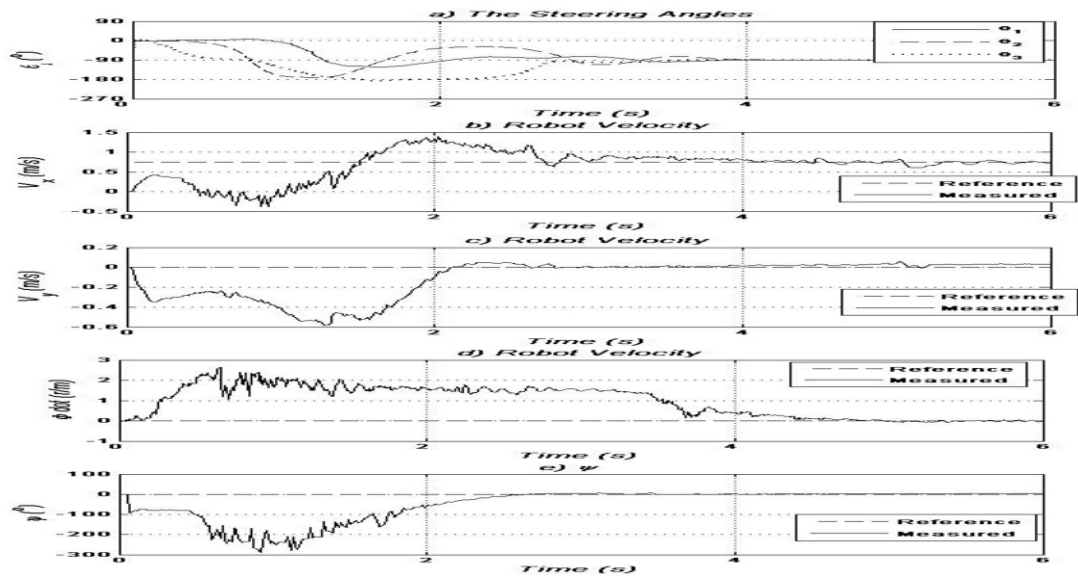
2.1.3 Solving The Singularity Problem For a Holonomic Mobile Robot [3]

Paper ini menjelaskan tentang penyelesaian permasalahan *singularity* untuk *multi-robot holonomic*. Solusi *inverse* menghasilkan *singularity* untuk beberapa konfigurasi roda yang sama. Dalam penelitian ini pendekatan yang diusulkan untuk menyelesaikan masalah *singularity* adalah pendekatan kontrol *escaping singularity*. Pendekatan kontrol *escaping singularity* aktif pada kontrol kecepatan, kecepatan tiap robot akan berubah saat kondisi *singularity* terdeteksi. *Singularity* roda terdeteksi saat $\psi = 90^\circ$. Untuk menghindari kesalahan posisi selama fase *escaping*, *linear velocity* robot harus sekecil mungkin dan *rotation velocity* harus bertambah dengan ψ , sehingga robot berputar dengan *translation* yang minimum seperti yang dijelaskan pada Gambar 2.10.

Pada Gambar 2.11 menunjukkan bahwa *multi-robot holonomic* dapat mengikuti *reference* dan dapat menghindari kondisi *singularity*. Namun, metode kontrol *escaping singularity* ini belum dibuktikan apakah dapat digunakan pada jenis robot *non-holonomic* yang terbatas pada gerakannya.



Gambar 2.10 *Escaping Singularity Block Diagram*



Gambar 2.11 Hasil *Escaping Singularity*

2.2 Dasar Teori

Pada sub-bab ini akan dibahas mengenai dasar teori yang digunakan pada penelitian ini. Adapun beberapa dasar teori tersebut adalah sebagai berikut:

2.2.1 Model *Wheeled Mobile Robot*

Wheeled Mobile Robot (WMR) merupakan robot beroda yang terdiri dari dua roda penggerak dan sumbu penggerak di tengah roda (Gambar 2.12). Dinotasikan untuk posisi saat ini (x_c, y_c) dan posisi *heading* θ_c yang merupakan titik tengah robot dan sudut antara *heading* dan sumbu x , untuk menggambarkan posisi robot saat ini [5]. Gambar 2.12 menggambarkan *mobile robot* beroda saat ini di *coordinate cartesius* dengan menggunakan variabel yang ditentukan. *Constrain non-holonomic* WMR sebagai berikut:

$$\dot{x}_c \sin \theta_c - \dot{y}_c \cos \theta_c = 0 \quad (2.3)$$

Dinotasikan untuk *linear velocity* v dan *angular velocity* ω [6]. Didapatkan model *kinematics* sebagai berikut:

WMR dalam *coordinate cartesius* dituliskan sebagai berikut:

$$\dot{q} = \begin{bmatrix} \dot{x}_c \\ \dot{y}_c \\ \dot{\theta}_c \end{bmatrix} = \begin{bmatrix} v_c \cos \theta_c \\ v_c \sin \theta_c \\ \omega_c \end{bmatrix} = \begin{bmatrix} \cos \theta_c & 0 \\ \sin \theta_c & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v_c \\ \omega_c \end{bmatrix} \quad (2.4)$$

Asumsi 1: jika ada *bounded disturbance* $f = [f_1 f_2 f_3]^T$, representasi persamaan menjadi:

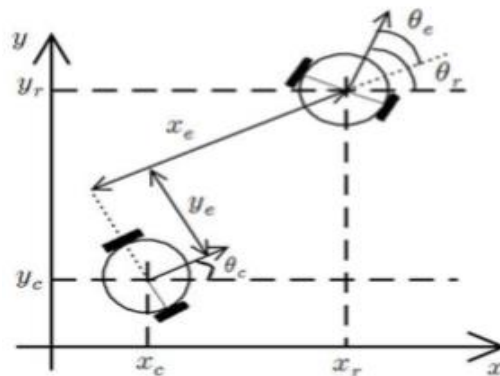
$$\dot{q} = \begin{bmatrix} \dot{x}_c \\ \dot{y}_c \\ \dot{\theta}_c \end{bmatrix} = \begin{bmatrix} \cos \theta_c & 0 \\ \sin \theta_c & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v_c \\ \omega_c \end{bmatrix} + \begin{bmatrix} f_1 \\ f_2 \\ f_3 \end{bmatrix} \quad (2.5)$$

Dimana $|f_i| \leq f_m, i = 1,2,3$. Untuk posisi *reference* $q_r = [x_r y_r \theta_r]^T$, didapat posisi *error* $q_e = [x_e y_e \theta_e]^T$ yang merupakan perbedaan antara posisi *desired* dan posisi *reference* sebagai berikut:

$$q_e = \begin{bmatrix} x_e \\ y_e \\ \theta_e \end{bmatrix} = \begin{bmatrix} \cos \theta_c & \sin \theta_c & 0 \\ -\sin \theta_c & \cos \theta_c & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_r - x_c \\ y_r - y_c \\ \theta_r - \theta_c \end{bmatrix} \quad (2.6)$$

dari representasi persamaan yang berisi *linear velocity* dan *angular velocity* sebagai berikut:

$$\dot{q}_e = \begin{bmatrix} \dot{x}_e \\ \dot{y}_e \\ \dot{\theta}_e \end{bmatrix} = \begin{bmatrix} y_e \omega_c - v_c + v_r \cos \theta_e + y_e f_3 \\ -x_e \omega_c + v_r \sin \theta_e - x_e f_3 \\ \omega_r - \omega_c \end{bmatrix} - \begin{bmatrix} \cos \theta_c & \sin \theta_c & 0 \\ -\sin \theta_c & \cos \theta_c & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} f_1 \\ f_2 \\ f_3 \end{bmatrix} \quad (2.7)$$



Gambar 2.12 *Coordinate WMR*

2.2.2 Model *Kinematics* dan *Dynamics* pada *Non-holonomic Mobile Robot*

Persamaan umum *kinematics* dan *dynamics* pada *mobile robot* dapat dituliskan sebagai berikut untuk $q = [x \ y \ \theta]$ [6]:

$$M(r)\ddot{r} + b(r, \dot{r}) + g(r) + \tau_d = B(r)\tau - A^T(r)\lambda \quad (2.8)$$

dimana:

$M(r)\ddot{r}$: Matriks inersia simetris definit positif

$b(r, \dot{r})$: Gaya *coriolis*, *centripetal* dan friksi

$g(r)$: Gaya gravitasi

τ_d : *Disturbance*

$B(r)$: Matriks transformasi *input*

τ : Torsi

$A^T(r)$: Matriks *constraint*

λ : Pengali *lagrange*

Persamaan *dynamics* didapatkan dari energi kinetik dan energi potensial dari sistem [7]. Energi potensial dari sistem adalah nol karena gerakan dibatasi ke tanah, sedangkan energi kinetik robot dapat diturunkan dari kecepatan pusat robot, untuk mencari *velocity* antara point A dan C yang dijelaskan pada Gambar 2.13, diperoleh persamaan sebagai berikut:

$$\begin{aligned} x_c &= x_A + a \cos \theta \\ y_c &= y_A + a \sin \theta \\ x_c &= y_A - a\theta \sin \theta \\ y_c &= y_A + a\theta \cos \theta \end{aligned} \quad (2.9)$$

dimana:

x_A : *Coordinate x point A*

y_A : *Coordinate y point A*

x_c : *Coordinate x point C*

y_c : Coordinate y point C

θ : Sudut orientasi robot terhadap titik pusa

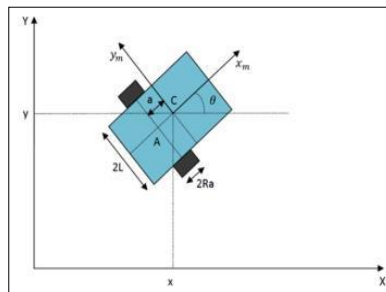
Untuk mencari *velocity* dari pusat robot terhadap point A yang dijelaskan pada Gambar 2.14, diperoleh persamaan sebagai berikut:

Velocity dari pusat robot terhadap titik A adalah:

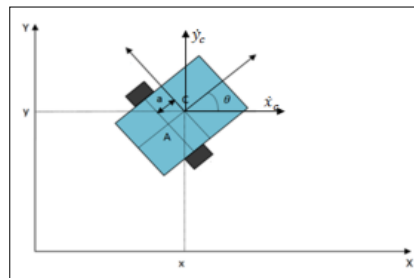
$$\begin{aligned} v_A &= \vec{x}_{cI} + \vec{y}_{cJ} + a\theta \sin \vec{\theta}_I - a\theta \cos \vec{\theta}_J \pi r^2 \\ v_A &= \vec{x}_{cI} + \vec{y}_{cJ} + a\theta \sin \vec{\theta}_I - a\theta \cos \vec{\theta}_J \\ v_A &= (x_c + a\theta \sin \theta)\vec{i} + (y_c - a\theta \cos \theta)\vec{j} \end{aligned} \tag{2.10}$$

dan untuk energi kinetik robot yaitu:

$$\begin{aligned} T &= \left(\frac{1}{2}\right)mv_{A^2} + \left(\frac{1}{2}\right)I_{A^2}\theta^2 \\ T &= \left(\frac{1}{2}\right)m[(x_c + a\theta \sin \theta)^2 \\ &\quad + (y_c - a\theta \cos \theta)^2] \end{aligned} \tag{2.11}$$



Gambar 2.13 *Differential Drive Mobile Robot*



Gambar 2.14 *Velocity* dari Pusat Robot

2.2.3 Cluster Control

Cluster menggunakan dua *state space* yaitu *robot space* dan *cluster space*. Variabel *robot space state* adalah posisi dan *velocity* yang digunakan untuk menggambarkan keadaan gerak robot s dengan *global frame*, dimana untuk *cluster multi-robot* yang ditunjukkan pada Gambar 2.15 [8].

Robot space pose vector didefinisikan sebagai berikut:

$$\vec{R} = (x_1, y_1, \theta_1, x_2, y_2, \theta_2, x_3, y_3, \theta_3)^T \quad (2.12)$$

dimana:

$(x_i, y_i, \theta_i)^T$: Posisi dan *heading* dari robot i.

Cluster space pose vector yaitu:

$$\vec{C} = (x_c, y_c, \theta_c, \phi_1, \phi_2, \phi_3, p, q, \beta)^T \quad (2.13)$$

dimana:

$(x_c, y_c, \theta_c)^T$: Posisi dan *heading cluster*

ϕ_i : *Heading yaw* dari robot i berdasarkan *cluster*

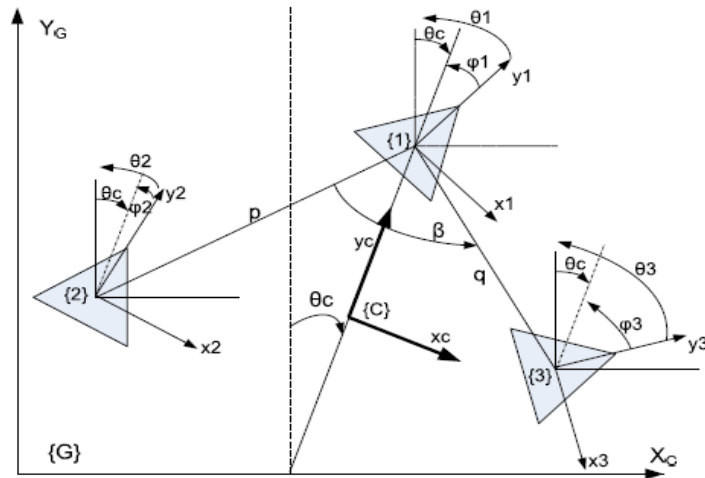
p dan q : Jarak dari robot 1 ke robot 2 dan 3, masing-masing

β : Sudut miring dengan *vertex* pada robot 1.

Dari *state pose vector* didapatkan posisi *pose vector* \vec{R} dan \vec{C} , sedangkan *pose vector velocity* adalah $\dot{\vec{R}}$ dan $\dot{\vec{C}}$. Persamaan ini dapat diselesaikan untuk posisi *robot space* yang menghasilkan persamaan posisi *inverse kinematics*, sehingga posisi *robot space* bisa dihasilkan berdasarkan posisi *cluster space*. Transformasi jacobian dan *inverse jacobian* digunakan untuk mentransformasi $\dot{\vec{R}}$ menjadi $\dot{\vec{C}}$ dan sebaliknya.

$$\dot{\vec{C}} = J(\vec{R}) * \dot{\vec{R}} \quad (2.14)$$

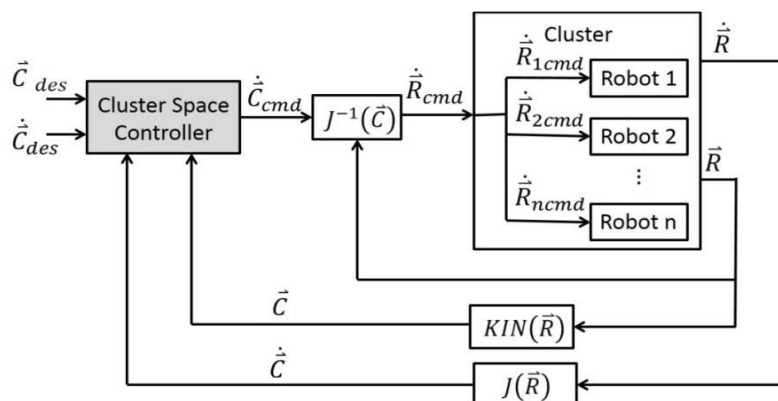
$$\dot{\vec{R}} = J^{-1}(\vec{C}) * \dot{\vec{C}} \quad (2.15)$$



Gambar 2.15 Cluster Definition

2.2.4 Kontrol Cluster Space

Kontrol untuk pengontrol *cluster space* ditunjukkan pada gambar 2.16. Dalam arsitektur ini, kontroler menerima spesifikasi kontrol sebagai variabel *cluster space* [9]. Kontrol kompensasi juga dihitung dalam *cluster space*, yang umumnya mengarah ke gerakan *cluster space* sesuai *trajectory* meskipun gerakan setiap robot cukup kompleks karena sifat *non-linier* dari hubungan *kinematics*. Sebagai robot yang mengeksekusi perintah individu *velocity*, posisi dan *velocity* yang secara kolektif dikonversi ke *cluster space* untuk digunakan oleh kontroler *cluster*.



Gambar 2.16 Cluster Controller for n Robots

2.2.5 Robot space

Robot space didefinisikan dengan cara yang sama untuk *robot cluster*, dengan penambahan robot tunggal yang memiliki x, y, θ pada setiap robot [11]. Tabel 2.1 adalah variabel pada *robot space*.

2.2.6 Cluster Space

Multi-robot dianggap sebagai satu kesatuan, yang disebut *cluster*. *Cluster* ini memiliki *dynamics* sendiri (disebut *cluster space*). Setiap posisi robot didapatkan dari *center cluster* [12]. *Cluster space* masih merupakan *frame* yang ideal untuk memberikan perintah kepada *cluster* [13]. Tabel 2.2 menunjukkan variabel dari *cluster space*.

Tabel 2.1 Variabel *Robot Space*

Variabel	Keterangan
x_1	Jarak dari robot 1 pada sumbu origin x (m)
y_1	Jarak dari robot 1 pada sumbu origin y (m)
θ_1	<i>Heading yaw</i> dari robot 1 pada θ -axis (rad)
x_2	Jarak dari robot 2 pada sumbu origin x (m)
y_2	Jarak dari robot 2 pada sumbu origin y (m)
θ_2	<i>Heading yaw</i> dari robot 2 pada θ -axis (rad)
x_3	Jarak dari robot 3 pada sumbu origin x (m)
y_3	Jarak dari robot 3 pada sumbu origin y (m)
θ_3	<i>Heading yaw</i> dari robot 3 pada θ -axis (rad)

Tabel 2.2 Variabel *Cluster Space*

Variabel	Keterangan
x	Jarak dari <i>cluster center</i> pada sumbu origin x (m)
y	Jarak dari <i>cluster center</i> pada sumbu origin y (m)
θ	Jarak dari <i>cluster center</i> pada sumbu origin θ (m)
ϕ_1	<i>Heading</i> robot 1 dengan hubungan <i>cluster x-axis</i> (rad)
ϕ_2	<i>Heading</i> robot 2 dengan hubungan <i>cluster x-axis</i> (rad)
ϕ_3	<i>Heading</i> robot 3 dengan hubungan <i>cluster x-axis</i> (rad)
p	Jarak <i>center</i> dari robot 1 & 2 (m)
q	Jarak <i>center</i> dari robot 1 & 3 (m)
β	Sudut antar p dan q (rad)

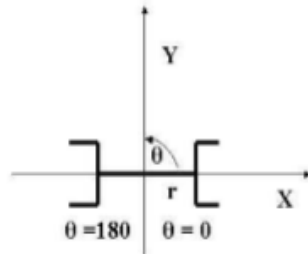
2.2.7 Singularity

Singularity terjadi ketika *joint velocity* di *joint space* menjadi tidak terbatas untuk mempertahankan *cartesian velocity* [14]. Dengan kata lain kontinuitas dalam *joint space* rusak bersamaan dengan *cartesian space*. *Singularity* saat Jacobian adalah 0 (artinya tidak bisa di *inverse*). Matriks Jacobian terkait dikatakan singular apabila:

$$\det(J) = 0 \quad (2.16)$$

Penyelesaian *singularity* dengan $\det(J) = -r$. Determinannya adalah 0 ketika $r = 0$. Karena r adalah jari-jari, robot memiliki *singularity* untuk bergerak dari *origin* dalam *cartesian space*. *Joint space velocity* dari *joint 1* menjadi *infinite* untuk mencapai vektor *cartesian velocity* dijelaskan pada Gambar 2.17.

Gripper di sepanjang sumbu x positif ke sumbu x negatif dengan mengurangi r , *joint 1* (θ) harus berputar dari 0 derajat ke 180 derajat ketika *gripper* melewati *origin*. Rotasi akan menyebabkan *joint* memiliki kecepatan tak terbatas ketika konfigurasi berubah dari sumbu x positif ke negatif.



Gambar 2.17 2-Link Near Origin

BAB 3

METODE PENELITIAN

Pada bab ini dibahas tentang penyelesaian permasalahan *singularity* untuk perancangan kontroler bisa mempertahankan *trajectory* dengan mengukur nilai *error* untuk *trajectory tracking* pada *Non-holonomic Multi-Robot* dengan *cluster space* untuk menghindari *singularity*.

3.1 Pemodelan Robot *Kinematics*

Robot *non-holonomic* yang digunakan adalah *Wheeled Mobile Robot* (WMR) dimana memiliki 2-roda yang harus diatur dan 1-roda pasif, yang ditunjukkan pada Gambar 3.1.

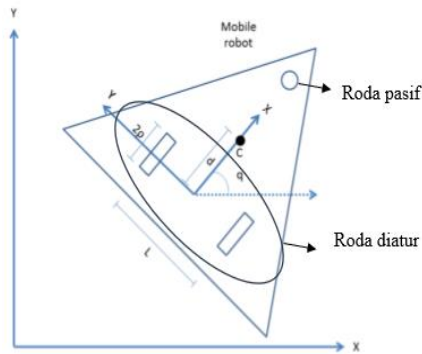
Pengaturan arah pergerakan sistem didapat dari perbedaan *velocity* antara roda kiri dan roda kanan. Persamaan untuk *linear velocity* dan *angular velocity* yang diperoleh dari hasil pengukuran *velocity* roda kanan dan kiri sebagai berikut [5]:

$$\begin{aligned} v &= \frac{v_R + v_L}{2} = R \frac{(w_R + w_L)}{2} \\ w &= \frac{v_R - v_L}{2} = R \frac{(w_R - w_L)}{2} \end{aligned} \quad (3.1)$$

Model *kinematics* untuk sumbu x dan y ditunjukkan oleh persamaan berikut:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \cos(\theta) & 0 \\ \sin(\theta) & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v \\ w \end{bmatrix} - \dot{x} \sin(\theta) + \dot{y} \cos(\theta) = 0 \quad (3.2)$$

dimana: (x, y) posisi, (θ) sudut *heading* dan *linear velocity* dan *angular velocity* (v dan w).



Gambar 3.1 Model WMR

3.2 Pemodelan Variabel Robot dan Cluster

Pada Gambar 3.2 dapat dilihat lokasi 3-robot pada *cluster frame*. Berdasarkan hal tersebut 9-variabel *state space robot* (3-robot dengan 3-DOF per robot) digambarkan menjadi 9-variabel *cluster space* untuk 9-*cluster* DOF. Robot *space pose vector* yaitu [2]:

$$\vec{R} = (x_1, y_1, \theta_1, x_2, y_2, \theta_2, x_3, y_3, \theta_3)^T \quad (3.3)$$

dimana:

$(x_i, y_i, \theta_i)^T$: posisi dan *heading* dari robot i.

Sedangkan *cluster space pose vector* didefinisikan sebagai:

$$\vec{C} = (x_c, y_c, \theta_c, \phi_1, \phi_2, \phi_3, p, q, \beta)^T \quad (3.4)$$

Hubungan antara *cluster space* dan variabel *robot space*, yang dapat ditunjukkan sebagai berikut:

$$x_c = \frac{x_1 + x_2 + x_3}{3}$$

$$y_c = \frac{y_1 + y_2 + y_3}{3}$$

$$\begin{aligned}
\theta_c & \text{atan2} \frac{2/3(x_1) - 1/3(x_2 + x_3)}{2/3(y_1) - 1/3(y_2 + y_3)} \\
\phi_1 & = \theta_1 + \theta_c \\
\phi_2 & = \theta_2 + \theta_c \\
\phi_3 & = \theta_3 + \theta_c \\
p & = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2} \\
q & = \sqrt{(x_3 - x_1)^2 + (y_1 - y_3)^2} \\
\beta & = \text{acos} \frac{p^2 + q^2 - (x_3 - x_1)^2 - (y_3 - y_2)^2}{2pq}
\end{aligned} \tag{3.5}$$

dimana:

$(x_c, y_c, \theta_c)^T$: Posisi dan *heading cluster*

ϕ_i : *Heading yaw* dari robot i berdasarkan *cluster*

p dan q : Jarak dari robot 1 ke robot 2 dan 3, masing-masing

β : Sudut miring dengan *vertex* pada robot 1.

Dengan adanya pemilihan variabel *cluster space state* yang disebutkan di atas, dimungkinkan untuk merepresentasikan *forward position kinematics* sebagai $F(r)$ dan *inverse* dari sistem 3-robot. Maka *kinematics* dari sistem 3-robot dibagi menjadi 2, yaitu *forward position kinematics* dan *inverse kinematics*. Persamaan *inverse position kinematics* sebagai berikut:

$$\begin{aligned}
x_1 & = x_c + \left(\frac{1}{3}\right) r \sin \theta_c \\
y_1 & = y_c + \left(\frac{1}{3}\right) r \cos \theta_c \\
\theta_1 & = \phi_1 - \theta_c \\
x_2 & = x_c + \left(\frac{1}{3}\right) r \sin \theta_c - p \sin \left(\frac{\beta}{2} + \theta_c\right) \\
y_2 & = y_c + \left(\frac{1}{3}\right) r \cos \theta_c - p \cos \left(\frac{\beta}{2} + \theta_c\right) \\
\theta_2 & = \phi_2 - \theta_c \\
x_3 & = x_c + \left(\frac{1}{3}\right) r \sin \theta_c + p \sin \left(\frac{\beta}{2} + \theta_c\right) \\
y_3 & = y_c + \left(\frac{1}{3}\right) r \cos \theta_c - q \cos \left(\frac{\beta}{2} + \theta_c\right)
\end{aligned} \tag{3.6}$$

$$\theta_3 = \phi_3 - \theta_c$$

dimana:

$$r: \sqrt{(q + p \cos \beta)^2 + (p \sin \beta)^2}$$

Persamaan matriks Jacobian dan *inverse* Jacobian adalah:

$$\dot{\vec{C}} = J(\vec{R}) * \dot{\vec{R}} \quad (3.7)$$

dengan

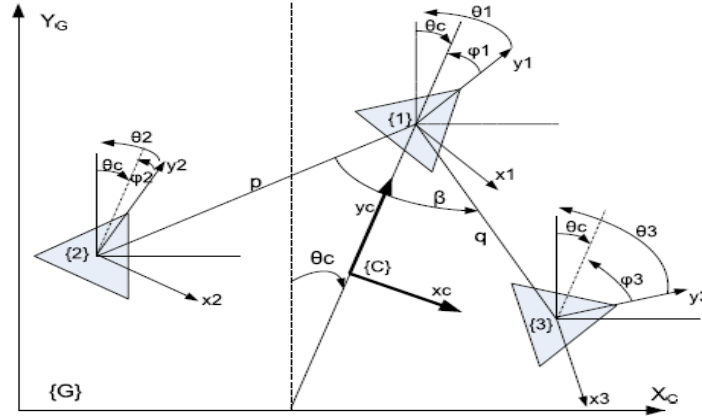
$$J(\vec{R}) = \begin{pmatrix} \frac{\partial c_1}{\partial r_1} & \frac{\partial c_1}{\partial r_2} & \dots & \frac{\partial c_1}{\partial r_9} \\ \frac{\partial c_2}{\partial r_1} & \frac{\partial c_2}{\partial r_2} & \dots & \frac{\partial c_2}{\partial r_9} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial c_9}{\partial r_1} & \frac{\partial c_9}{\partial r_2} & \dots & \frac{\partial c_9}{\partial r_9} \end{pmatrix}$$

dan inversenya:

$$\dot{\vec{R}} = J^{-1}(\vec{C}) * \dot{\vec{C}} \quad (3.8)$$

dengan

$$J^{-1}(\vec{C}) = \begin{pmatrix} \frac{\partial r_1}{\partial c_1} & \frac{\partial r_1}{\partial c_2} & \dots & \frac{\partial r_1}{\partial c_9} \\ \frac{\partial r_2}{\partial c_1} & \frac{\partial r_2}{\partial c_2} & \dots & \frac{\partial r_2}{\partial c_9} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial r_9}{\partial c_1} & \frac{\partial r_9}{\partial c_2} & \dots & \frac{\partial r_9}{\partial c_9} \end{pmatrix}$$



Gambar 3.2 Cluster Space and Robot Space Variables

3.3 Pemodelan Robot Dynamics

Persamaan *dynamics* diperoleh dari penurunan permodelan menggunakan formula *Lagrange dynamics*. Dengan mengacu pada pembahasan sub-bab sebelumnya, model dari *dynamics* sistem dinyatakan dengan persamaan sebagai berikut [1]:

$$M(r)\ddot{r} + b(r, \dot{r}) + g(r) + \tau_d = B(r)\tau - A^T(r) \quad (3.9)$$

dimana: $r = [x, y, q]^T$ dan $M(r) \in \mathfrak{R}^{3 \times 3}$ menunjukkan matriks inersia simetris definitif positif; $b(r, \dot{r}) \in \mathfrak{R}^{1 \times 3}$ adalah gaya *coriolis*, centripetal dan friksi; $g(r) \in \mathfrak{R}^{3 \times 1}$ adalah gaya gravitasi, τ_d adalah *disturbance*, dan $\tau \in \mathfrak{R}^{2 \times 1}$ adalah torsi. $A \in \mathfrak{R}^{1 \times 3}$ adalah matriks *constraint* yang dikalikan dengan pengali *Lagrange* $\lambda \in \mathfrak{R}^{1 \times 1}$, dan persamaan *constraint* adalah $A(r)\dot{r}=0$, dengan

$$M(r) = \begin{bmatrix} m & 0 & -ma \sin(q) \\ 0 & m & ma \cos(q) \\ -ma \sin(q) & ma \cos(q) & I + ma^2 \end{bmatrix}$$

$$b(r, \dot{r}) = \begin{bmatrix} 0 & 0 & ma\dot{q}^2 \cos(q) \\ 0 & 0 & ma\dot{q}^2 \sin(q) \\ 0 & 0 & 0 \end{bmatrix}; g(r): 0$$

$$B(r) = \frac{1}{2\rho} \begin{bmatrix} \cos \theta & \cos \theta \\ \sin \theta & \sin \theta \\ L & -L \end{bmatrix}; \tau = \begin{bmatrix} \tau_1 \\ \tau_2 \\ \tau_3 \end{bmatrix}; A^T(r) = \begin{bmatrix} -\sin \theta \\ \cos \theta \\ 0 \end{bmatrix}$$

Misalkan $S_r(r) \in \mathfrak{R}^{3 \times 2}$ menjadi matriks *full rank* sehingga $S_r^T(r) A^T(r) = 0$ maka:

$$S_r(r) = \begin{bmatrix} \cos \theta & 0 \\ \sin \theta & 0 \\ 0 & 1 \end{bmatrix}$$

Dalam hal memiliki $i \in i = 1,2,3$ robot, didapatkan model yang berbeda sehingga:

$$M_i(r_i) \ddot{r}_i + b_i(r_i, \dot{r}_i) + g_i(r_i) + \tau_d = B_i(r_i) \tau_i - A_i^T(r_i) \lambda_i \quad (3.10)$$

$S_{ri}(r_i) \in \mathfrak{R}^{3 \times 2}$ menjadi matriks *full rank*, sehingga $S_{ri}^T(r_i) A_i^T(r_i) = 0$.

3.4 Pemodelan *Dynamics* pada *Cluster Space*

Untuk memilih *cluster space state*, beberapa kondisi harus dipertimbangkan. *Cluster space* harus menggambarkan fungsi penggunaanya, seperti kontrol formasi. Jumlah *cluster* DOF (*Degrees of Freedom*) harus sama dengan jumlah DOF *robot space*. *Dynamics* pada *cluster* dapat dihitung dengan mengubah *dynamics* pada *robot space* melalui matriks Jacobian dengan $\dot{c} = J(r, \dot{r})$. *Dynamics* pada *cluster space* dimulai dari *robot space dynamics* dengan n robot, model dari *dynamics* sistem dinyatakan dengan persamaan sebagai berikut [1]:

$$\bar{M}(r)\ddot{r} + \bar{b}(r, \dot{r}) + \bar{g}(r) + \tau_d = \bar{B}(r)\tau - \bar{A}^T \bar{\lambda} \quad (3.11)$$

dengan

$$\bar{M}(r) = \begin{bmatrix} M_1 & 0^{3 \times 3} & \dots & 0^{3 \times 3} \\ \cdot & M_2 & 0^{3 \times 3} & \vdots \\ \cdot & \cdot & M_3 & 0^{3 \times 3} \end{bmatrix}; \bar{b}(r, \dot{r}) = \begin{bmatrix} b_1(r_1, \dot{r}_1) \\ b_2(r_2, \dot{r}_2) \\ b_3(r_3, \dot{r}_3) \end{bmatrix}$$

$$\bar{B}(r) = \begin{bmatrix} B_1 & 0^{3 \times 2} & \dots & 0^{3 \times 2} \\ \cdot & B_2 & 0^{3 \times 2} & \vdots \\ \cdot & \cdot & B_3 & 0^{3 \times 2} \end{bmatrix}; \bar{A}^T(r) = \begin{bmatrix} A_1^T & 0^{3 \times 1} & \dots & 0^{3 \times 1} \\ \cdot & A_2^T & 0^{3 \times 1} & \vdots \\ \cdot & \cdot & A_3^T & 0^{3 \times 1} \end{bmatrix}$$

$$\bar{\lambda} = [\lambda_1, \lambda_2, \lambda_3]^T; \bar{\tau} = [\tau_1, \tau_2, \tau_3]^T$$

Maka persamaan *dynamics* pada *cluster non-holonomic* yang dicoupling dengan *state cluster* dapat dilihat pada persamaan berikut:

$$\begin{aligned}\Lambda(c)\ddot{c} + \mu(c, \dot{c}) + p(c) + \tau_d &= \beta_h(c)\bar{\tau} \\ \Lambda(c)\ddot{c} + \mu(c, \dot{c}) + p(c) + \tau_d &= \beta(c)\bar{\tau} - \alpha^T(c)\bar{\lambda}\end{aligned}\quad (3.12)$$

dengan

$$\begin{aligned}\Lambda(c) &= J^{-T}(r)\bar{M}(r)J^{-1}(r) \\ \mu(c, \dot{c}) &= (r)\bar{b}(r, \dot{r}) - \Lambda(c)j(r, \dot{r})\dot{r} \\ p(c) &= J^{-T}(r)g(r) \\ \beta(c) &= J^{-T}(r)\bar{B}(r) \\ \alpha^T(c) &= J^{-T}(r)\bar{A}^T(r)\end{aligned}$$

Persamaan *constraint* $\alpha^T(c)\dot{c} = 0$ menjadi:

$$S_c(c) = J \begin{bmatrix} S_{r1} & 0^{3 \times 3} & \dots \\ \cdot & S_{r2} & 0^{3 \times 3} \\ \cdot & \cdot & S_{r3} \end{bmatrix} \quad (3.13)$$

Sehingga $S_c^T(c)\alpha^T(c) = 0$ dengan tambahan $V(t)$ menjadi $\dot{c} = S_c(c)V(t)$.

dengan

$$V(t) = [v_1, w_1, v_2, w_2, v_3, w_3]$$

Untuk v_i, w_i adalah *heading* dan *angular velocity* dari robot *non-holonomic*.

Didapatkan persamaan *derivative* sebagai berikut:

$$\ddot{c} = S_c(c)\dot{V}(t) + \dot{S}_c(c)V(t) \quad (3.14)$$

dengan perkalian kedua sisi dengan $S_c^T(c)$, persamaan sebagai berikut :

$$H\dot{V} + E + \tau_d = \bar{\tau} \quad (3.15)$$

dengan

$$\begin{aligned}H &= (S_c^T(c)\beta(c))^{-1}S_c^T(c)(\Lambda(c)\dot{S}_c(c)) \\ E &= (S_c^T(c)\beta(c))^{-1}S_c^T(c)(\Lambda(c)\dot{S}_c(c)V(t) + \mu(c, \dot{c}) + P(c)) \\ \tau_d &= Hf, f \in \mathfrak{R}^{3 \times nx1}, f = [f_{11}, f_{21}, f_{31}, f_{12}, f_{22}, f_{32}, \dots, f_{1.1}, f_{2.2}, f_{3.3}]\end{aligned}$$

Persamaan *dynamics* pada *cluster space* telah didapat menggunakan persamaan tanpa perhitungan *Lagrange*.

3.5 Pemodelan *Tracking Error Cluster Space*

Pemodelan *tracking error* yang dilakukan pertama menghitung *error cluster* dengan dimulai dari mendefinisikan nilai parameter untuk *c* sebagai nilai *actual cluster space*, $\vec{C} = (x_c, y_c, \theta_c, \phi_1, \phi_2, \phi_3, p, q, \beta)^T$, sedangkan nilai *reference* $c_d (\vec{C}_{cd} = (x_{cd}, y_{cd}, \theta_{cd}, \phi_{1d}, \phi_{2d}, \phi_{3d}, p_d, q_d, \beta_d)^T)$. Sehingga *error signal*, $\Delta c = W(\vec{C}_{cd} - \vec{C})$ dimana *W* disini adalah matriks *diagonal*. *Robot space* adalah transformasi dari signal *error cluster space* yang dikalikan dengan *inverse* matriks Jacobian [1].

$$\bar{\Delta r} = J^{-1} \Delta \vec{C} \quad (3.16)$$

dengan

$$\begin{aligned} \bar{\Delta r} &= [\Delta x_1, \Delta y_1, \Delta \theta_1, \Delta x_2, \Delta y_2, \Delta \theta_2, \Delta x_3, \Delta y_3, \Delta \theta_3]^T \\ \bar{\Delta r}_i &= [\Delta x_i, \Delta y_i, \Delta \theta_i]^T \end{aligned}$$

setelah itu *robot space* dimodifikasi menjadi persamaan sebagai berikut:

$$\begin{aligned} \Delta r &= [\Delta x_1, \Delta y_1, \Delta \theta_{1e}, \Delta x_2, \Delta y_2, \Delta \theta_{2e}, \Delta x_3, \Delta y_3, \Delta \theta_{3e}]^T \\ \Delta r_i &= [\Delta x_i, \Delta y_i, \Delta \theta_{ie}]^T \end{aligned} \quad (3.17)$$

dengan

$$\theta_{ie} = \tan^{-1}(\Delta y_i / \Delta x_i) - R_i$$

Persamaan transformasi kontrol robot sebagai berikut:

$$r_{ie} = \begin{bmatrix} x_{ie} \\ y_{ie} \\ \theta_{ie} \end{bmatrix} = \begin{bmatrix} \cos(\theta_i) & \sin(\theta_i) & 0 \\ -\sin(\theta_i) & \cos(\theta_i) & 0 \\ 0 & 0 & 1 \end{bmatrix} \Delta r_i \quad (3.18)$$

dimana: $(x_{ie}, y_{ie}, \theta_{ie})$ lokasi *error cluster*. $V_i = [v_i, w_i]^T$, $V_{id} = [v_{id}, w_{id}]^T$ dan v_i, w_i adalah *actual robot heading* dan *rotation velocity robot*, sedangkan v_{id}, w_{id}

adalah *desired robot heading* dan *rotation velocity robot*. Didapat persamaan *derivative error robot* yaitu:

$$\dot{r}_{ie} = \begin{bmatrix} \dot{x}_{ie} \\ \dot{y}_{ie} \\ \dot{\theta}_{ie} \end{bmatrix} = \begin{bmatrix} y_{ie}w_i - v_i + v_{id}\cos(\theta_{ie}) \\ -x_{ie}w_i + v_{id}\sin(\theta_{ie}) \\ w_{id} - w_i \end{bmatrix} \quad (3.19)$$

Dari kontrol klasik *kinematics* yang diberikan oleh Chen *et al* didapatkan kontrol *velocity* sebagai berikut:

$$v_c = \begin{bmatrix} v_{1c} \\ w_{1c} \\ v_{2c} \\ w_{2c} \\ v_{3c} \\ w_{3c} \end{bmatrix} = \begin{bmatrix} v_{1d}\cos(\theta_{1e}) + k_{11}x_{1e} \\ w_{1d} + k_{12}v_{1d}y_{1e} + k_{13}v_{1d}\sin(\theta_{1e}) \\ v_{2d}\cos(\theta_{2e}) + k_{21}x_{2e} \\ w_{2d} + k_{22}v_{2d}y_{2e} + k_{23}v_{2d}\sin(\theta_{2e}) \\ v_{3d}\cos(\theta_{3e}) + k_{31}x_{3e} \\ w_{3d} + k_{32}v_{3d}y_{3e} + k_{33}v_{3d}\sin(\theta_{3e}) \end{bmatrix} \quad (3.20)$$

dimana $k_{11}, k_{22}, k_{33} > 0, i|j = 1,2,3$.

Persamaan *error robot kinematics* didapat sebagai berikut:

$$e_c(t) = [e_{c1}(t), e_{c2}(t), e_{c3}(t)]^T = v_c(t) - v(t) \quad (3.21)$$

$$\dot{e}_c(t) = \dot{v}_c(t) - \dot{v}(t)$$

dengan

$$v = \begin{bmatrix} v_1 \\ w_1 \\ v_2 \\ w_2 \\ v_3 \\ w_3 \end{bmatrix}$$

3.6 Cluster Singularity

Kondisi *singularity* terjadi ketika nilai $p = 0$ dan $q = 0$ dimana p dan q disini merupakan jarak. Untuk menghindari kondisi tersebut digunakan teknik *collision avoidance*. Sistem *multi-robot* dapat menghindari tabrakan di antara robot, dengan menggunakan fungsi potensial untuk setiap kontrol robot. Skema kontrol desentralisasi pada *mobile-robot* untuk membentuk formasi yang dapat menghindari tabrakan merupakan fungsi potensial dan *input* kontrol yang tidak terikat. Rancangan *input* kontrol terbatas dengan potensi praktis terbatas berfungsi

untuk mengatasi masalah kontrol formasi untuk sistem *mobile-robot*. *Input* kontrol yang dibatasi dapat diselesaikan untuk kontrol *tracking mobile-robot* menggunakan fungsi potensial. Untuk menghindari tabrakan, penggunaan fungsi potensial hanya untuk melakukan *repulsive force*. Penggunaan aturan kontrol untuk setiap robot ketika robot lain berada di wilayah yang berpotensi tabrakan. Strategi menghindari tabrakan didasarkan pada fungsi potensial, yang diberikan gaya tolak untuk setiap robot. Fungsi potensial menggunakan persamaan berikut:

$$v_{ij}(p_i, p_j) = K_{ij} \ln(\cosh(p_{ij})) h_{ij}(p_i, p_j) \quad (3.22)$$

dimana

$$K_{ij} > 0, p_{ij} = \|p_i - p_j\| - c_i \text{ dan}$$

$$h_{ij}(p_i, p_j) = \begin{cases} 1 & \text{for } \|p_i - p_j\| < b_i \\ 0 & \text{jika tidak} \end{cases}$$

dengan $0 < a_i < b_i < c_i$ maka a_i dan b_i adalah parameter *collision avoidance* dan p_a merupakan *coordinate* robot yang harus dihindari oleh robot lainnya $i(t)$ dinotasikan sebagai *neighbor* robot i . Struktur potensi total energi dari semua *neighbor* di sekeliling robot i didefinisikan dengan persamaan:

$$U_i(p_i) = \sum_{j \in N_i(t)} V_{ij}(p_i, p_j) \quad (3.23)$$

Kekuatan struktural interaktif antara robot i dan j adalah gradien energi potensial yang didefinisikan dengan persamaan :

$$f_{ij}(p_j, p_i) = \Delta V_{ij} = K_{ij} \tanh(p_{ij})(p_j - p_i) / \|p_i - p_j\| \quad (3.24)$$

Fungsi ini menjelaskan energi yang dipaksakan ketika kondisi pada fungsi ke h aktif. Jika robot lain berdekatan dengan robot ke i di wilayah yang harus dihindari, persamaan total struktural kekuatan yang bekerja padanya didefinisikan sebagai berikut :

$$F_i(q_i) = \sum_{kj \in N(t)} f_{ij}(p_j, p_i) \quad (3.25)$$

3.7 Kontrol *Escaping Singularity*

Kontrol *escaping singularity* berfungsi untuk mengatur *velocity* robot apabila kondisi *singularity* terjadi. Persamaan kontrol *escaping singularity* sebagai berikut [3]:

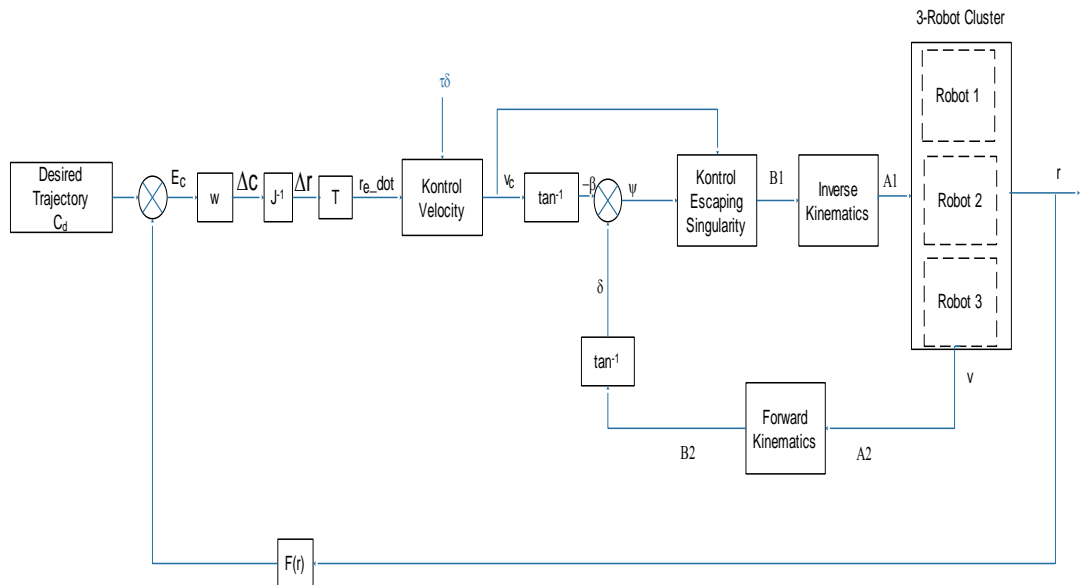
$$\begin{pmatrix} V_x \\ V_y \\ \dot{\phi} \end{pmatrix} = \begin{pmatrix} e^{-\lambda\psi} & 0 & 0 \\ 0 & e^{-\lambda\psi} & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} V_{xr} \\ V_{yr} \\ \dot{\phi}_r \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ K \end{pmatrix} \psi \quad (3.26)$$

Sinyal rotasi berasal dari *error* sudut $\psi = \delta - \beta$.

dengan

$$\delta = \tan^{-1}\left(-\frac{V_{ya}}{V_{xa}}\right) \text{ dan } \beta = \tan^{-1}\left(-\frac{V_{yr}}{V_{xr}}\right)$$

dimana $e^{-\lambda\psi}$: Kecepatan *linier*.



Gambar 3.3 Blok Diagram Perancangan Kontrol

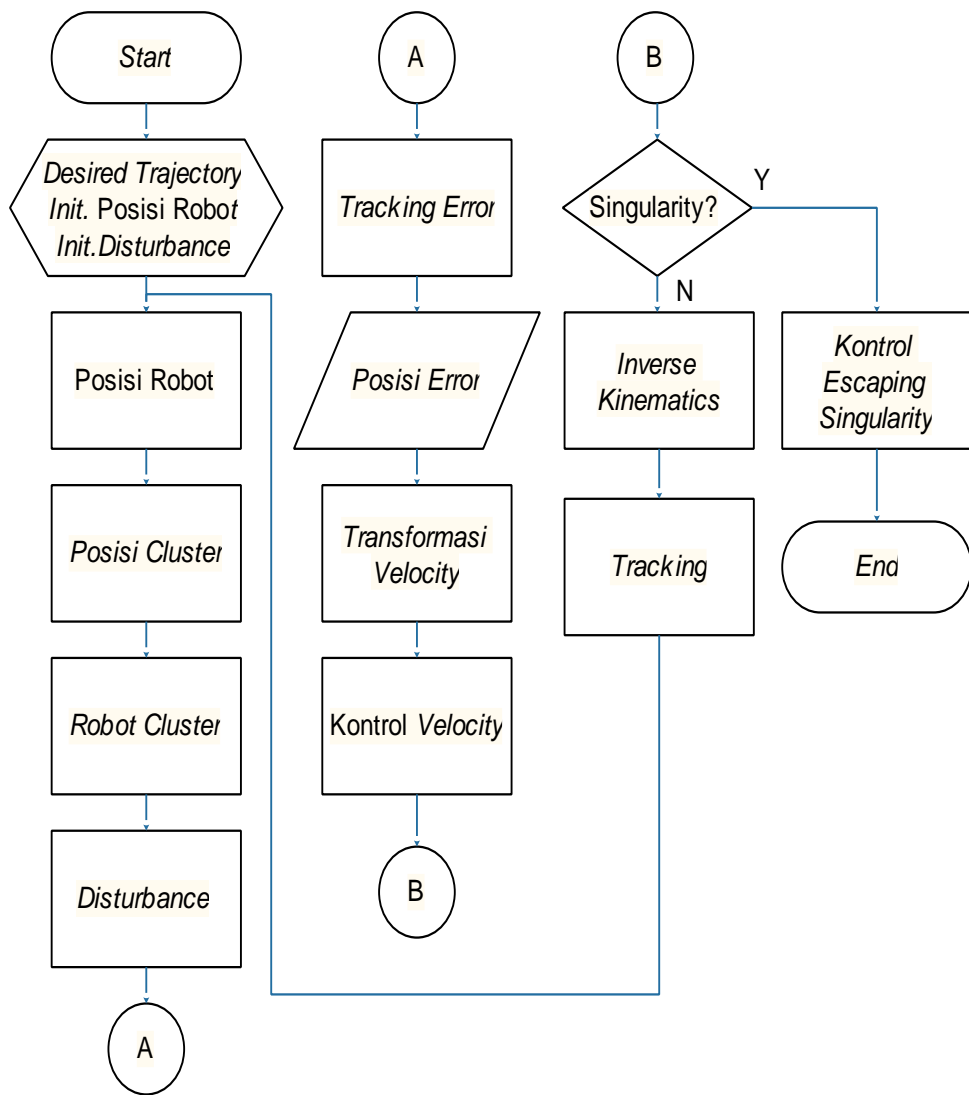
3.8 Blok Diagram Perancangan Kontrol

Gambar 3.3 menunjukkan arsitektur kontrol *cluster space*, gerakan *desired* dan aksi kontrol dihitung dalam *cluster space*, aksi kontrol dikonversi ke *robot space* melalui penggunaan *inverse Jacobian*. Gambar 3.3 menunjukkan arsitektur kontrol untuk kontrol *cluster space* berbasis *trajectory* dari sistem 3-robot.

Cluster membandingkan posisi dan kecepatan *cluster* dengan nilai *desired trajectory* C_d dan *output* untuk *cluster velocity*, yang diterjemahkan ke dalam kecepatan masing-masing robot melalui *inverse Jacobian*. Karena adanya *inverse Jacobian* pada *tracking* mengakibatkan adanya *singularity*, diberikan kontrol *escaping singularity* yang bertujuan untuk menyelesaikan permasalahan *singularity*. Keluaran dari kontrol *escaping singularity* masih berupa *cluster space* bukan dalam bentuk *robot space*, maka harus dikonversi dari *cluster space* ke *robot space* dengan diberi tambahan adanya *inverse kinematics* serta tambahan adanya *forward kinematics* dari *robot space* ke *cluster space*. Dari keluaran *dynamics* pada *cluster* didapat nilai r dan v setelah itu di *feedback* untuk mendapatkan nilai *error*. Pendekatan menggunakan kontrol *escaping singularity* dan *inverse kinematics* dilakukan pada *loop*. *Disturbance* yang disisipkan pada kontrol *velocity* dengan membangkitkan secara random.

3.9 Flowchart Perancangan

Dari blok diagram perancangan kontrol seperti pada sub bab sebelumnya yang telah dijabarkan dapat juga digambarkan dalam bentuk *flowchart*. Pada alur sistem kontrol, dijelaskan awal mulai sistem adalah inialisasi posisi robot dan lokasi tujuan yang ditunjukkan pada Gambar 3.4.



Gambar 3.4 *Flowchat* Perancangan

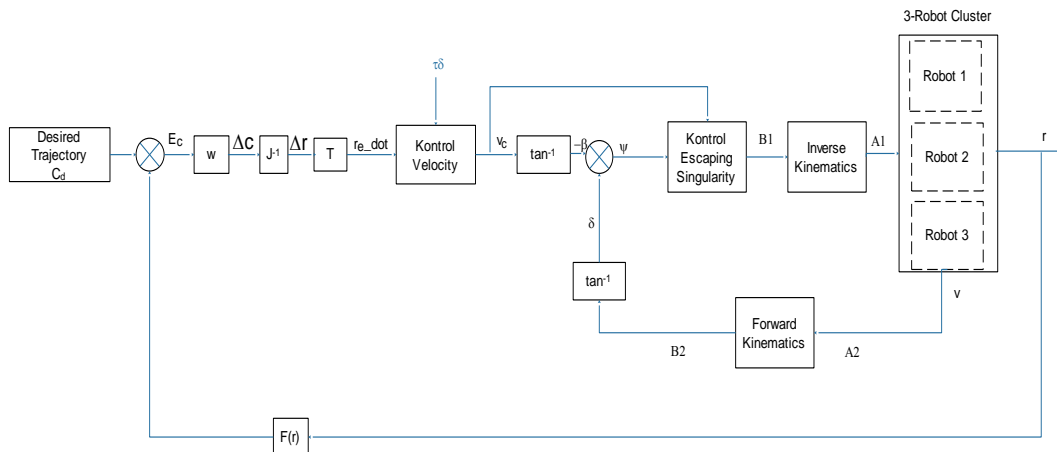
Halaman ini sengaja dikosongkan

BAB 4

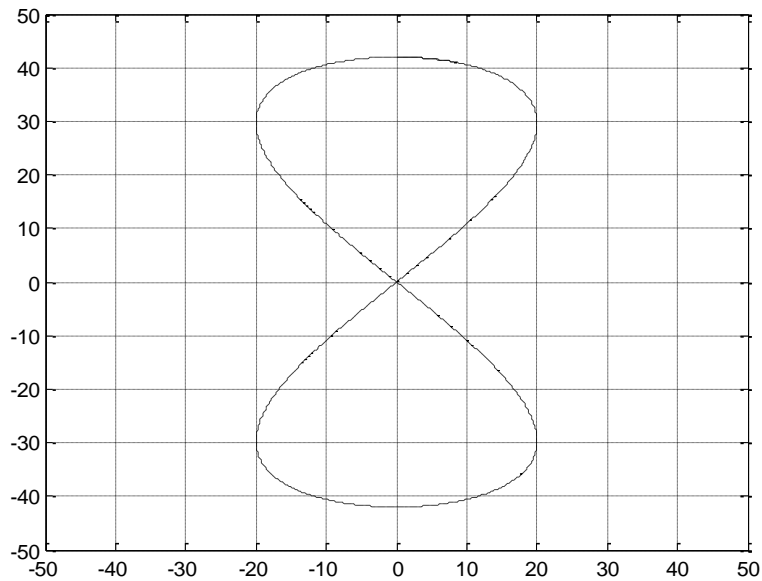
HASIL DAN PEMBAHASAN

Pada bab ini akan dijelaskan mengenai analisa dan pengujian sistem yang telah dibuat berdasarkan perancangan pada bab sebelumnya. Pengujian dilakukan melalui beberapa tahap pengujian dengan membuat lintasan berbentuk *berzier curve* menyerupai angka 8 yang merupakan *trajectory multi-robot*. Misi perjalanan *multi-robot* dijelaskan bagaimana suatu kontrol agar tidak terjadi tabrakan antara *mobile robot* satu dengan *mobile robot* yang lainnya yang berjalan pada *trajectory* yang telah ditentukan. Untuk menggambarkan kemampuan kontrol *cluster space*, 3-robot dalam *multi-robot* akan dijelaskan sebagai misi perjalanan yang memanfaatkan pendekatan *cluster space*.

Penggunaan *Cluster space* menyajikan pendekatan sederhana sebuah *multi-robot* dalam pembentukan kontrol agar tidak terjadi tabrakan antara *mobile robot* satu dengan *mobile robot* yang lainnya. Konsep *multi-robot* sebagai satu kesatuan *cluster* dan gerakan robot yang ditentukan sebagai fungsi dari sebuah *cluster*, yang terdiri dari posisi dan orientasi robot. Variabel ini memandu pemilihan sekumpulan variabel dalam kontrol robot sebagai variabel status yang membentuk sistem *cluster space*. Melalui transformasi *kinematics* dapat digunakan sebagai *cluster commands* untuk dikonversi menjadi perintah pada robot, dan untuk data status robot dapat dikonversi menjadi data status *cluster space* yang dijelaskan pada Gambar 4.1.



Gambar 4.1 Blok Pemodelan Kontrol



Gambar 4.2 *Plotting Reference Trajectory*

4.1 Pengujian Pembentukan *Reference Trajectory*

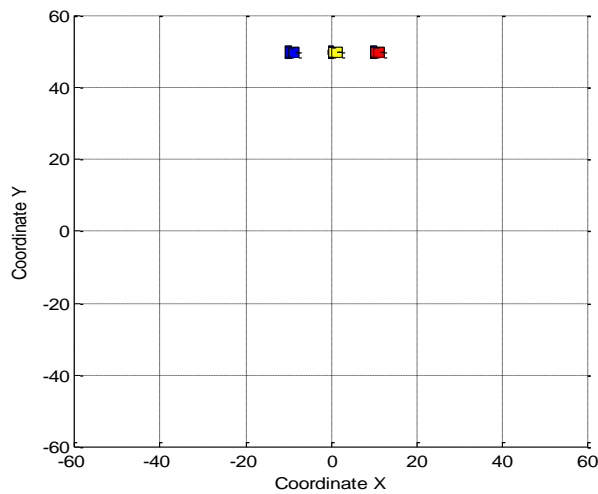
Pengujian pembentukan *reference trajectory* dengan membuat lintasan berbentuk *berzier curve* menyerupai angka 8 yang ditunjukkan pada Gambar 4.2 dengan cara *plotting*.

4.2 Pengujian *Plotting* Posisi Awal Robot

Posisi awal robot didefinisikan dalam bentuk *coordinate* yang mewakili variabel posisi x , posisi y dan *heading* robot. Konfigurasi posisi awal robot dapat dijelaskan sebagai berikut:

1. Robot 1 $x_{pr1} = 20, y_{pr1} = 50, \theta_{tar1} = 0$
2. Robot 2 $x_{pr2} = -10, y_{pr2} = 50, \theta_{tar2} = 0$
3. Robot 3 $x_{pr3} = 10, y_{pr3} = 50, \theta_{tar3} = 0$

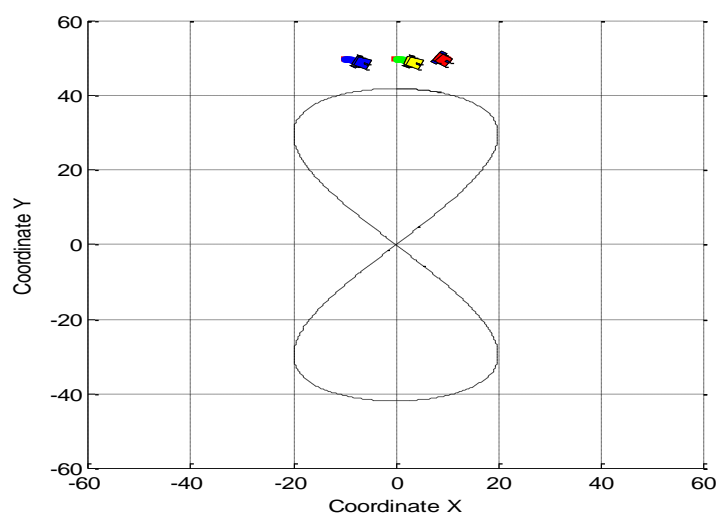
Variabel x_{pr} merupakan titik *coordinate* robot pada sumbu x , y_{pr} merupakan titik *coordinate* robot pada sumbu y dan θ_{tar} merupakan *heading* awal robot. Robot berwarna dasar merah merupakan robot 1, robot berwarna dasar biru merupakan robot 2 dan robot berwarna dasar kuning merupakan robot 3 yang dijelaskan pada Gambar 4.3.



Gambar 4.3 Inisialisasi Posisi Awal Robot

4.3 Pengujian *Plotting* Posisi Awal Robot dan *Reference Trajectory*

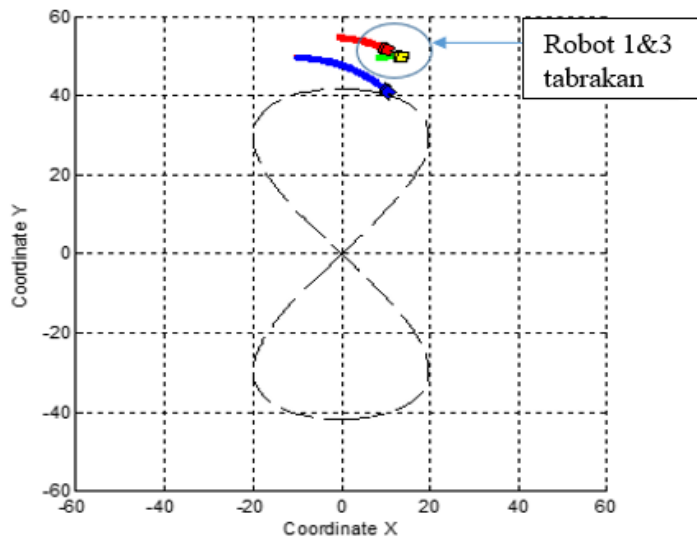
Pengujian untuk menggabungkan posisi awal robot dan *reference trajectory*. Pada Gambar 4.4 pengujian ini untuk membuktikan apakah posisi awal robot dan *reference trajectory* dapat disajikan dalam bentuk plot yang sesuai dengan perancangan.



Gambar 4.4 *Plotting* Posisi Awal Robot dan *Reference Trajectory*

4.4 Pengujian Simulasi Robot tanpa *Collision Avoidance*

Pada pengujian ini penulis menguji simulasi *multi-robot* tanpa *collision avoidance* robot dalam menghindari tabrakan antar robot. Pengujian ini untuk menunjukkan kejadian jika diantara ketiga robot terjadi tabrakan yang ditunjukkan pada hasil *plotting* simulasi pergerakan robot pada Gambar 4.5. Simulasi terjadinya tabrakan Gambar 4.5 menjelaskan robot 1 dan robot 3 terjadi tabrakan yang mengakibatkan terhentinya proses pergerakan robot 1 dan robot 3. Sementara robot 2 tidak terjadi tabrakan sehingga robot 2 tetap dapat menjalankan misi perjalanan.

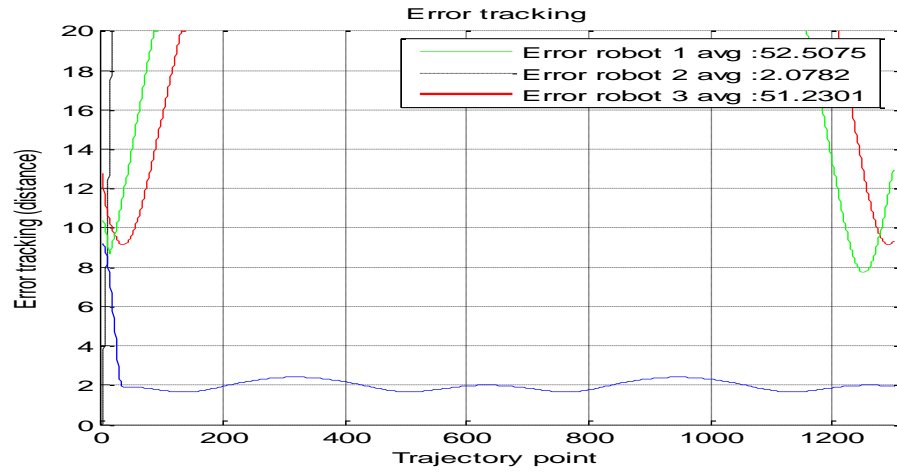


Gambar 4.5 Simulasi Terjadinya Tabrakan antara Robot 1 dan Robot

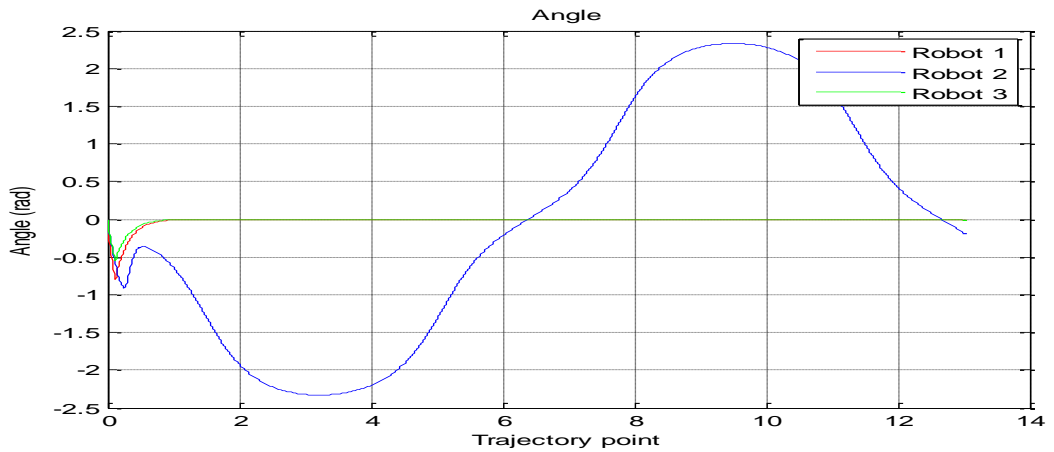
Pada hasil simulasi ditunjukkan pula *plotting error* posisi robot terhadap *reference trajectory* yang ditunjukkan Gambar 4.6. Karena robot 1 dan robot 3 terjadi tabrakan dan terhentinya proses pergerakan robot, maka pada hasil *tracking error* robot 1 dan robot 3 terjadi nilai kesalahan pelacakan yang sangat besar ditunjukkan dengan nilai *tracking error* rata-rata sebesar 52.5075 untuk robot 1 dan 51.2301 untuk robot 3. Untuk robot 2 yang tidak mengalami tabrakan diperoleh nilai *tracking error* yang relatif kecil dengan rata-rata *error* sebesar 2.0782.

Pengujian pada proses ini dapat dihasilkan *plotting* yang menunjukkan perubahan sudut robot yang ditunjukkan pada Gambar 4.7. Pada *plotting* menunjukkan perbedaan perubahan sudut ketiga robot yang menjelaskan perubahan

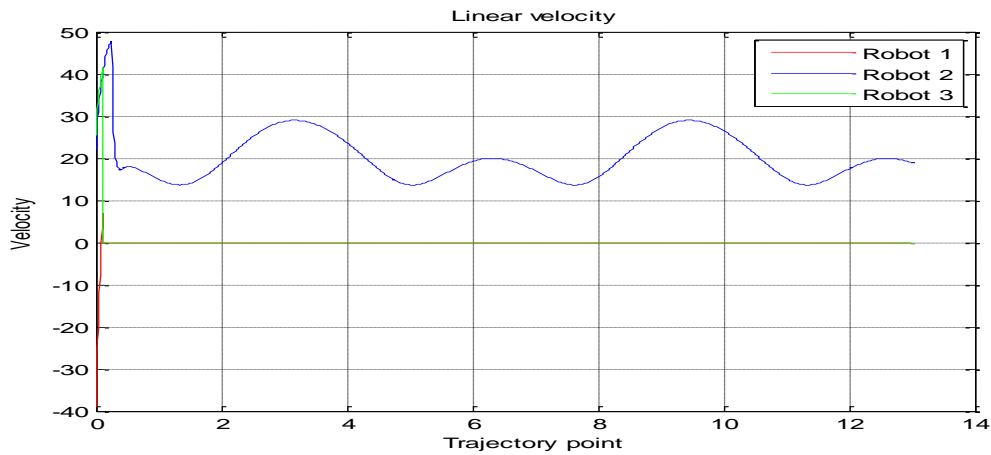
sudut yang terjadi karena robot tidak terjadi tabrakan, saat robot mengalami tabrakan akan ditunjukkan dengan perubahan sudut sebesar 0. Plot dengan warna biru menunjukkan perubahan sudut robot 2 saat menjalankan misi perjalanan, warna hijau dan merah menunjukkan bahwa robot 1 dan robot 3 tidak mengalami perubahan sudut orientasi karena mengalami tabrakan.



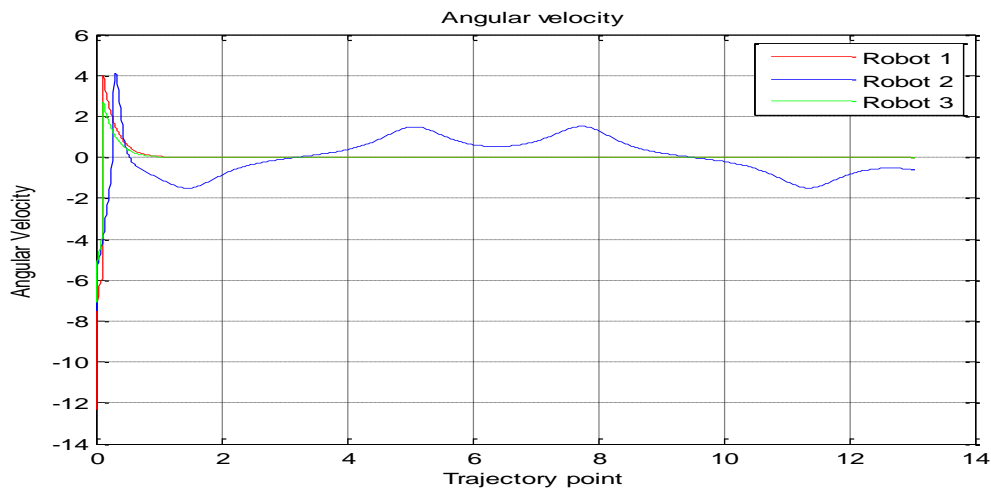
Gambar 4.6 Tracking Error Multi-Robot



Gambar 4.7 Perubahan Sudut Orientasi Multi-Robot



Gambar 4.8 Perubahan *Linear Velocity Multi-Robot*



Gambar 4.9 Perubahan *Angular Velocity Multi-Robot*

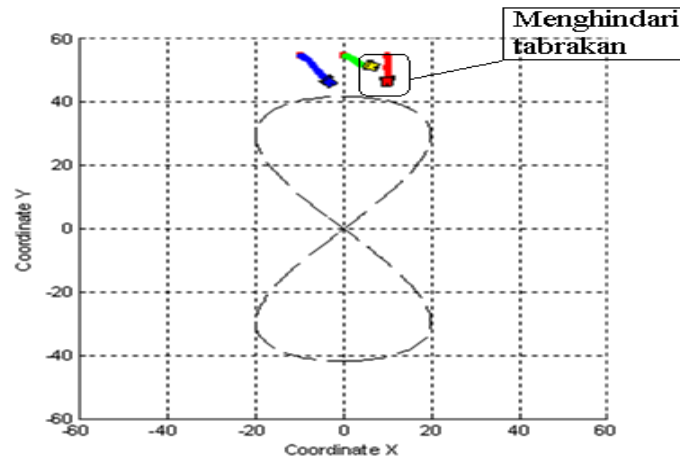
Proses terjadinya tabrakan antara robot 1 dan robot 3 dapat dibuktikan pada hasil *plotting* data *linear velocity* masing-masing robot pada Gambar 4.8. Robot yang mengalami tabrakan akan menghasilkan nilai *linear velocity* sebesar 0 karena kedua robot tidak melakukan pergerakan. Robot 2 mengalami perubahan nilai *linear velocity* karena robot 2 melakukan pergerakan sesuai dengan lintasan.

Gambar 4.9 menjelaskan perubahan *angular velocity* pada pengujian simulasi *multi-robot* yang telah dilakukan penulis. Gambar 4.9 menjelaskan *angular velocity* pada robot yang mengalami tabrakan dengan nilai 0 dan *angular velocity* pada robot 2 yang tidak mengalami tabrakan. *Angular velocity* pada robot

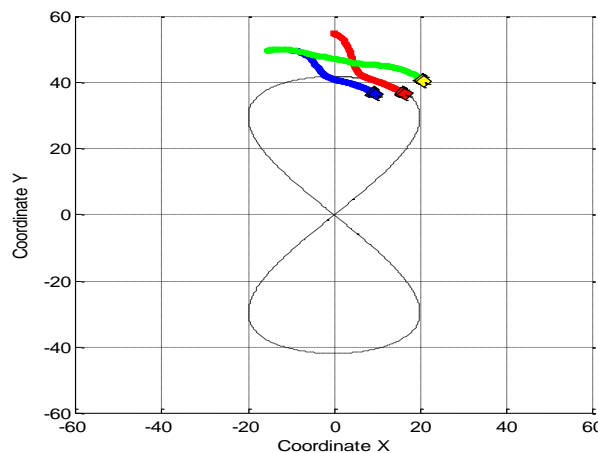
2 mengalami perubahan *velocity* karena robot 2 dapat melakukan perjalanan sesuai dengan lintasan yang dibuat.

4.5 Pengujian Simulasi Robot Menggunakan *Collision Avoidance*

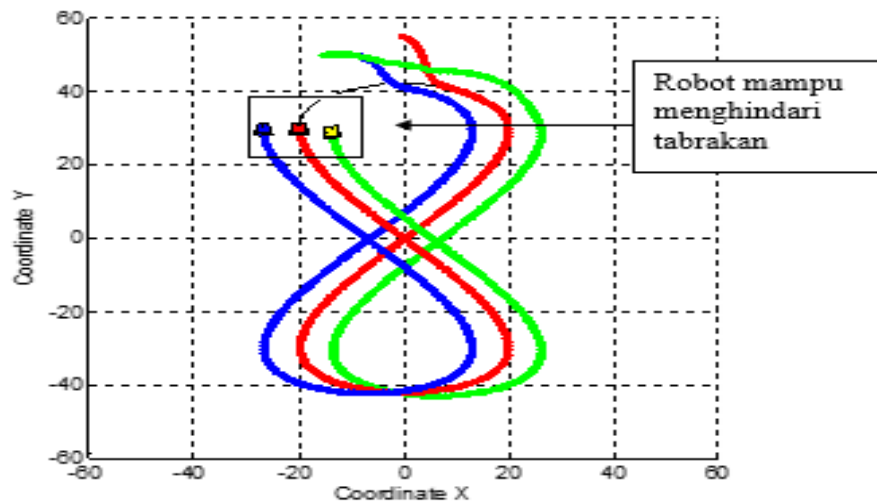
Pengujian *multi-robot* menggunakan *collision avoidance* robot dalam menghindari tabrakan antar robot. Pengujian ini untuk menunjukkan kejadian jika diantara ketiga robot terjadi tabrakan dan bagaimana robot dapat mengatasi permasalahan tabrakan yang ditunjukkan pada hasil *plotting* simulasi pergerakan robot pada Gambar 4.10, Gambar 4.11 dan Gambar 4.12. Hasil simulasi Gambar 4.10, 4.11 dan Gambar 4.12 menjelaskan terjadinya potensi terjadi tabrakan antara robot 1 dan robot 2.



Gambar 4.10 Simulasi Potensi Terjadinya Tabrakan antara Robot 1 dan Robot 3

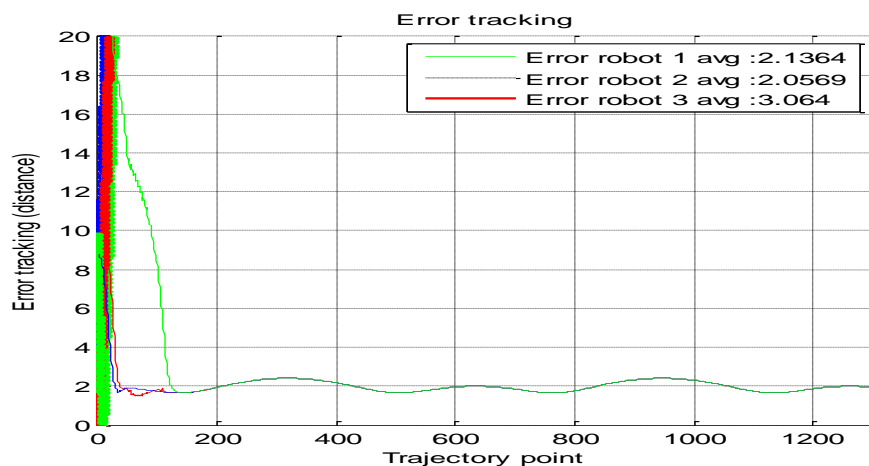


Gambar 4.11 Simulasi Robot 1 Menghindari Tabrakan dengan Robot 3

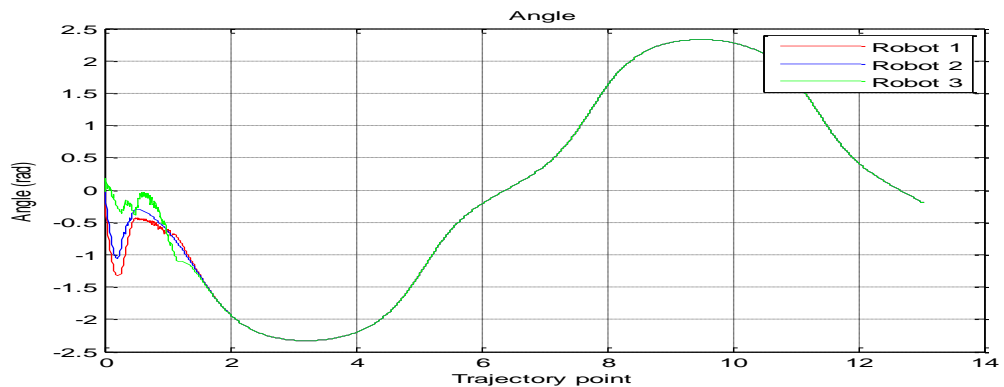


Gambar 4.12 Simulasi *Collision Avoidance* Robot 1 dan Robot 3

Pada hasil simulasi ditunjukkan pula *plotting error* posisi robot terhadap *reference trajectory* yang ditunjukkan Gambar 4.13. Hasil simulasi menjelaskan potensi tabrakan antara robot 1 dan robot 3 dan proses terjadinya *collision avoidance* pada robot 1 dan robot 3 sehingga mampu terhindar dari tabrakan. Pada hasil simulasi *tracking error* robot 1 dan robot 3 terjadi nilai kesalahan pelacakan yang lebih besar dari pada robot 2, ditunjukkan dengan nilai *tracking error* rata-rata sebesar 3.064 untuk robot 3 dan 2.1364 untuk robot 1. Untuk robot 2 yang tidak mengalami tabrakan diperoleh nilai *tracking error* yang lebih kecil dengan rata-rata *error* sebesar 2.0569.



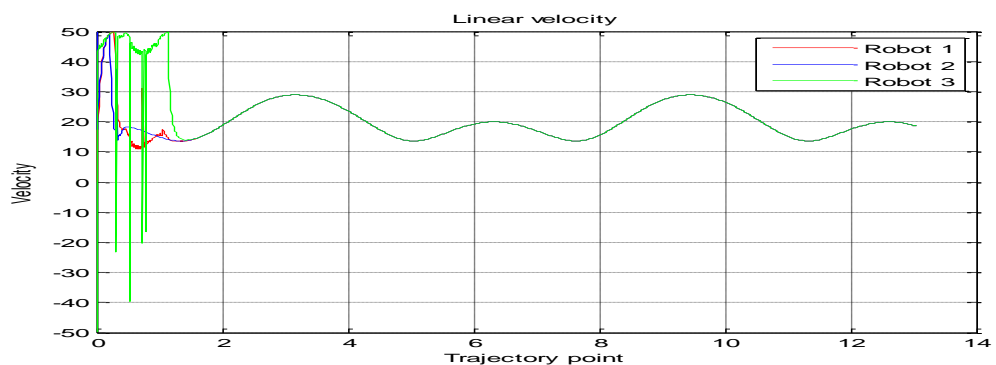
Gambar 4.13 *Tracking Error Multi-Robot*



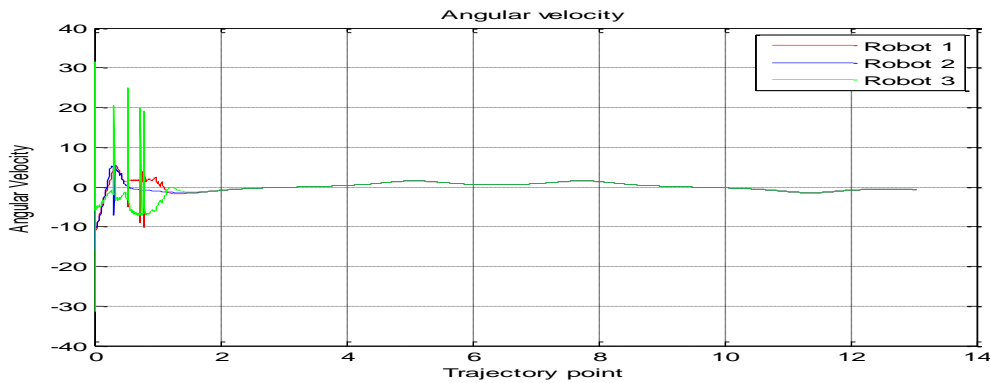
Gambar 4.14 Perubahan Sudut Orientasi *Multi-Robot*

Pengujian pada proses ini dapat dihasilkan *plotting* untuk menunjukkan perubahan sudut robot yang ditunjukkan pada Gambar 4.14. Pada *plotting* menunjukkan perbedaan perubahan sudut ketiga robot yang menjelaskan perubahan sudut orientasi saat melakukan misi perjalanan, saat robot mengalami potensi tabrakan akan ditunjukkan dengan perubahan sudut sebesar yang berbeda dengan perubahan sudut orientasi robot 2. Plot dengan warna biru menunjukkan perubahan sudut robot 2 saat menjalankan misi perjalanan, warna hijau dan merah menunjukkan bahwa robot 1 dan robot 3 mengalami berpotensi terjadi tabrakan.

Proses terjadinya potensi tabrakan antara robot 1 dan robot 3 dapat dibuktikan pada hasil *plotting* data *linear velocity* masing-masing robot pada Gambar 4.15. Robot yang mengalami potensi tabrakan akan menghasilkan nilai *linear velocity* yang berbeda dibandingkan dengan nilai *linear velocity* robot yang tidak berpotensi tabrakan.



Gambar 4.15 Perubahan *Linear Velocity Multi-Robot*



Gambar 4.16 Perubahan *Angular Velocity Multi-Robot*

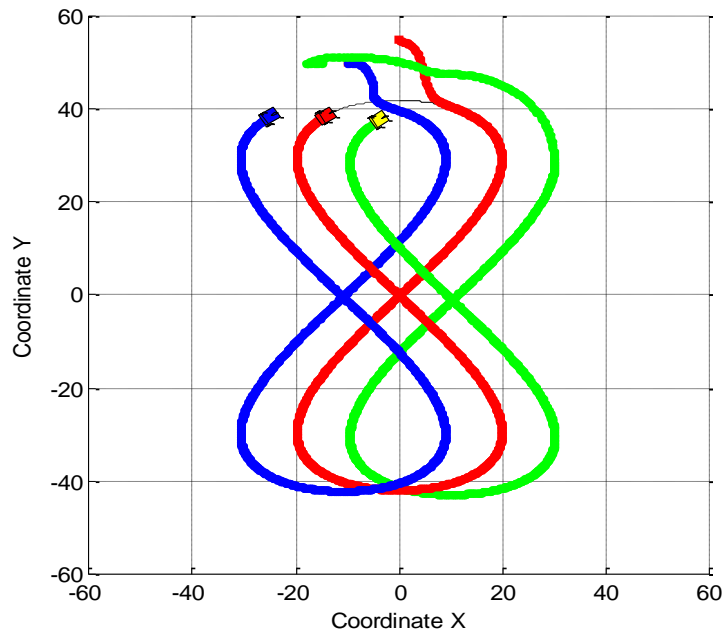
Gambar 4.16 menjelaskan perubahan *angular velocity* pada pengujian simulasi *multi-robot* yang telah dilakukan. Gambar 4.16 menjelaskan *angular velocity* pada robot yang berpotensi tabrakan dan *angular velocity* pada robot 2 yang tidak mengalami tabrakan. *Angular velocity* pada robot 2 mengalami perubahan *velocity* karena robot 2 dapat melakukan perjalanan sesuai dengan lintasan yang dibuat.

4.6 Pengujian Simulasi Robot Menggunakan *Collision Avoidance* dengan Parameter $\beta = 90^\circ$ & $p = 10$

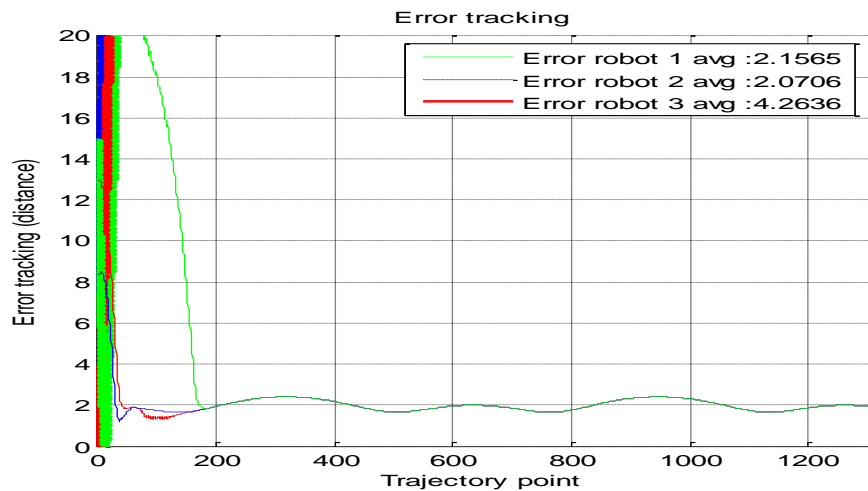
Pada pengujian ini penulis menguji simulasi *multi-robot* menggunakan *collision avoidance* robot dengan nilai parameter p sebesar 10. Nilai p merupakan nilai jarak antar robot 1 dan robot 2, sedangkan nilai parameter β sebesar 90° . Nilai β merupakan nilai sudut *vertex* robot 1. Pengujian ini untuk membuktikan apakah sistem *multi-robot* yang dibuat dapat melakukan kontrol kestabilan terhadap *reference trajectory* dan kestabilan jarak juga sudut sehingga terbentuk sebuah formasi perjalanan robot dalam menuju ke titik tujuan. Lintasan sebagai *reference trajectory* dalam bentuk *berzier curve*. Gambar 4.17 menunjukkan hasil simulasi *multi-robot* dalam melakukan misi perjalanan dengan jejak perjalanan yang didefinisikan dengan warna merah untuk robot 1, warna biru untuk robot 2 dan warna hijau untuk robot 3.

Pada hasil simulasi ditunjukkan pula *plotting error* posisi robot terhadap *reference trajectory* yang ditunjukkan Gambar 4.18. Pada hasil simulasi *tracking*

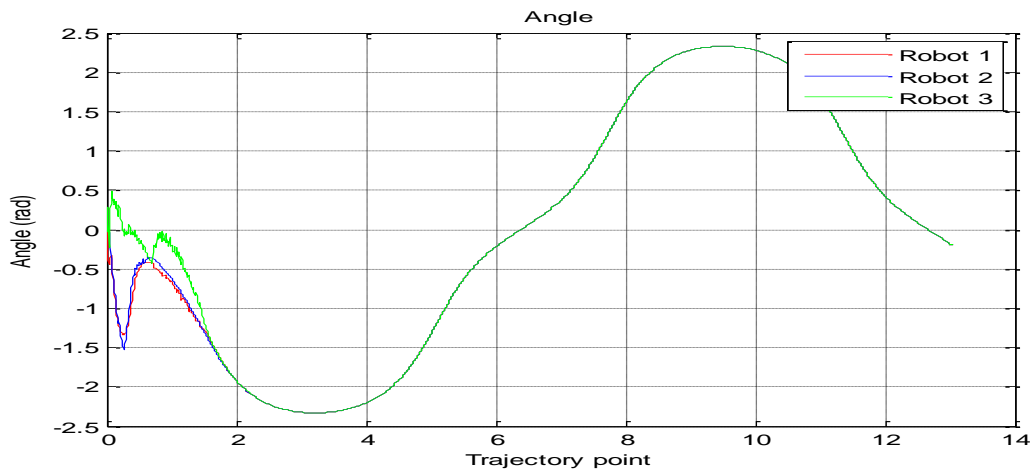
error robot 1, robot 2 dan robot 3 terjadi nilai kesalahan pelacakan yang relatif sama, hasil menunjukkan dengan nilai *tracking error* rata-rata sebesar 2.1565 untuk robot 1 dan 2.0706 untuk robot 2 dan 4.2636 untuk robot 3. Dengan nilai *tracking error* yang relatif sama maka formasi robot akan tetap terjaga.



Gambar 4.17 Simulasi *Multi-Robot* $\beta=90^\circ$ & $p=10$

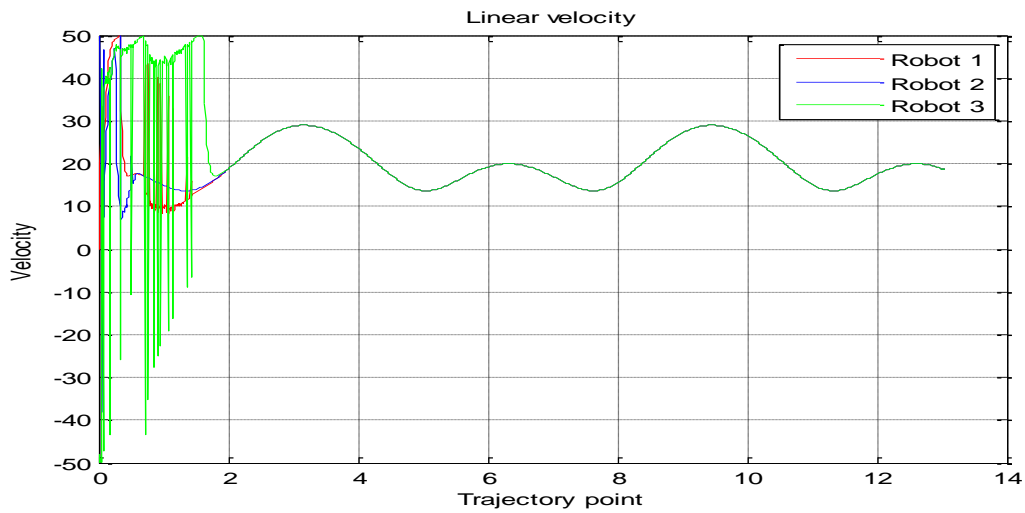


Gambar 4.18 *Tracking Error Multi-Robot*

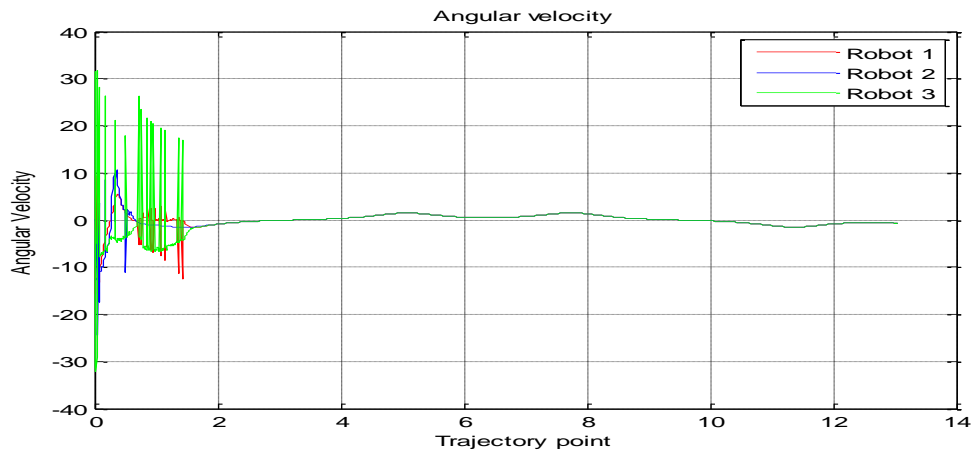


Gambar 4.19 Perubahan Sudut Orientasi *Multi-Robot*

Pada pengujian ini dapat dihasilkan *plotting* untuk menunjukkan perubahan sudut robot yang ditunjukkan pada Gambar 4.19. Pada *plotting* menunjukkan perbedaan perubahan sudut ketiga robot yang menjelaskan perubahan sudut orientasi saat melakukan misi perjalanan, perubahan sudut orientasi ketiga robot relatif sama.



Gambar 4.20 Perubahan *Linear Velocity Multi-Robot*



Gambar 4.21 Perubahan *Angular Velocity Multi-Robot*

Proses perjalanan ketiga robot tidak menghasilkan nilai *linear velocity* yang relatif sama karena pada proses simulasi potensi tabrakan yang muncul antar robot dapat dihindarkan. Hasil *linear velocity multi-robot* ditunjukkan pada Gambar 4.20.

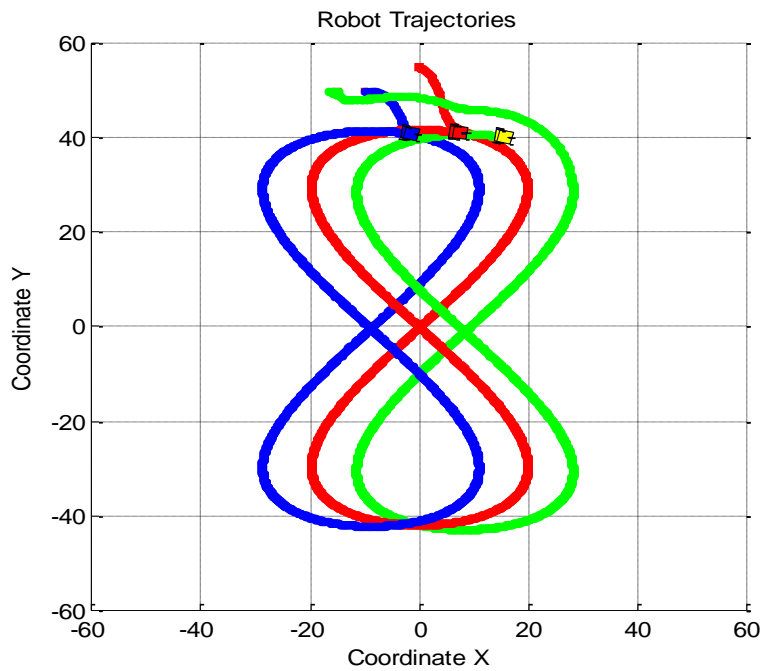
Gambar 4.21 menjelaskan perubahan kecepatan sudut atau *angular velocity* pada pengujian simulasi *multi-robot* yang telah dilakukan.

4.7 Pengujian Simulasi Robot Menggunakan *Collision Avoidance* dengan Parameter $\beta = 90^\circ$ & $p = 8$

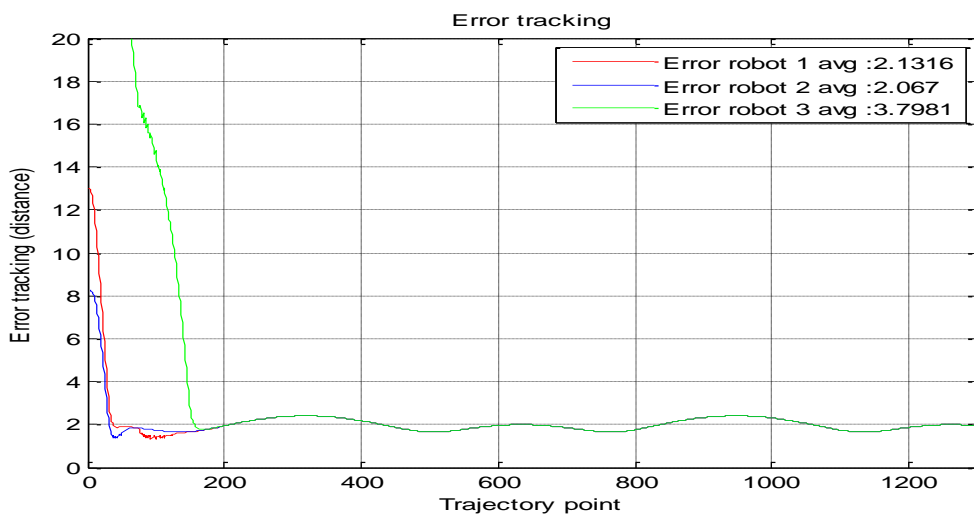
Pengujian *multi-robot* menggunakan *collision avoidance* robot dengan nilai parameter p sebesar 8. Nilai p merupakan nilai jarak antar robot 1 dan robot 2, sedangkan nilai parameter β sebesar 90° . Nilai β merupakan nilai sudut *vertex* robot 1. Pengujian ini untuk membuktikan apakah sistem *multi-robot* yang dibuat dapat melakukan kontrol kestabilan terhadap *reference trajectory* dan kestabilan jarak juga sudut, sehingga terbentuk sebuah formasi perjalanan robot dalam menuju ke titik tujuan. Lintasan sebagai *reference trajectory* dalam bentuk *berzier curve*. Gambar 4.22 menunjukkan hasil simulasi *multi-robot* dalam melakukan misi perjalanan dengan jejak perjalanan yang didefinisikan dengan warna merah untuk robot 1, warna biru untuk robot 2 dan warna hijau untuk robot 3.

Pada hasil simulasi ditunjukkan pula *plotting error* posisi robot terhadap *reference trajectory* yang ditunjukkan Gambar 4.23. Pada hasil simulasi *tracking*

error robot 1, robot 2 dan robot 3 terjadi nilai kesalahan pelacakan yang relatif sama, dengan nilai *tracking error* rata-rata sebesar 2.1316 untuk robot 1 dan 2.067 untuk robot 2 dan 3.7981 untuk robot 3. Dengan nilai *tracking error* yang relatif sama maka formasi robot akan tetap terjaga.



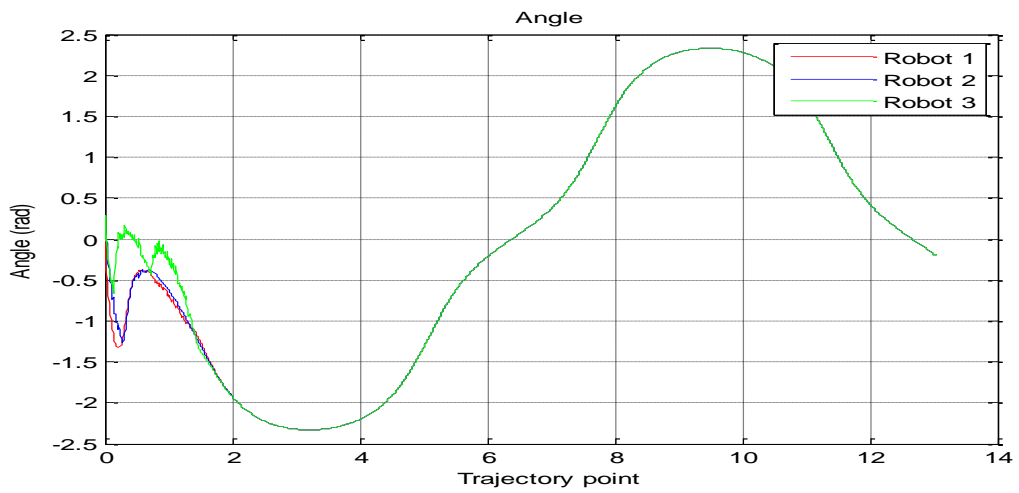
Gambar 4.22 Simulasi *Multi-Robot* $\beta=90^\circ$ & $p=8$



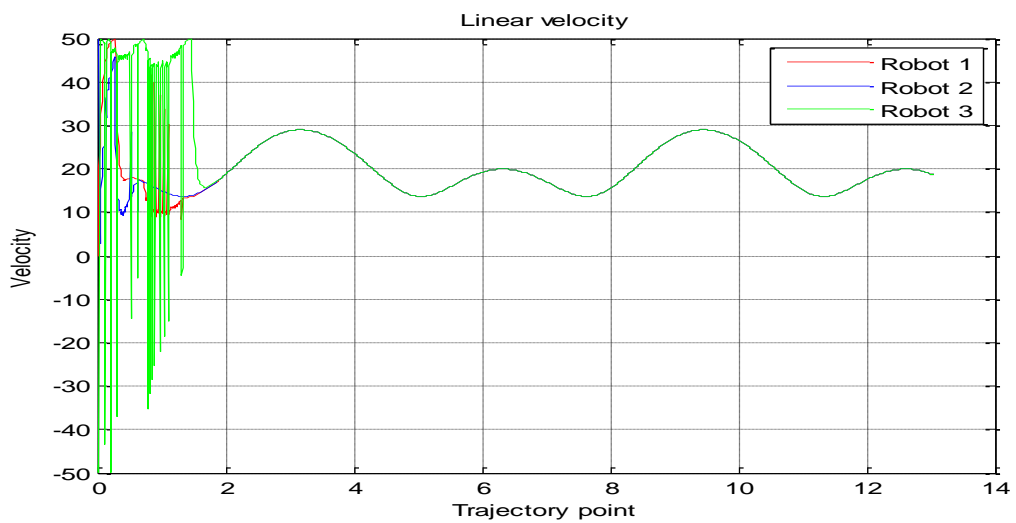
Gambar 4.23 *Tracking Error Multi-Robot*

Pada pengujian ini dapat dihasilkan *plotting* untuk menunjukkan perubahan sudut robot yang ditunjukkan pada Gambar 4.24. Pada *plotting* menunjukkan perbedaan perubahan sudut ketiga robot yang menjelaskan perubahan sudut orientasi saat melakukan misi perjalanan, perubahan sudut orientasi ketiga robot relatif sama.

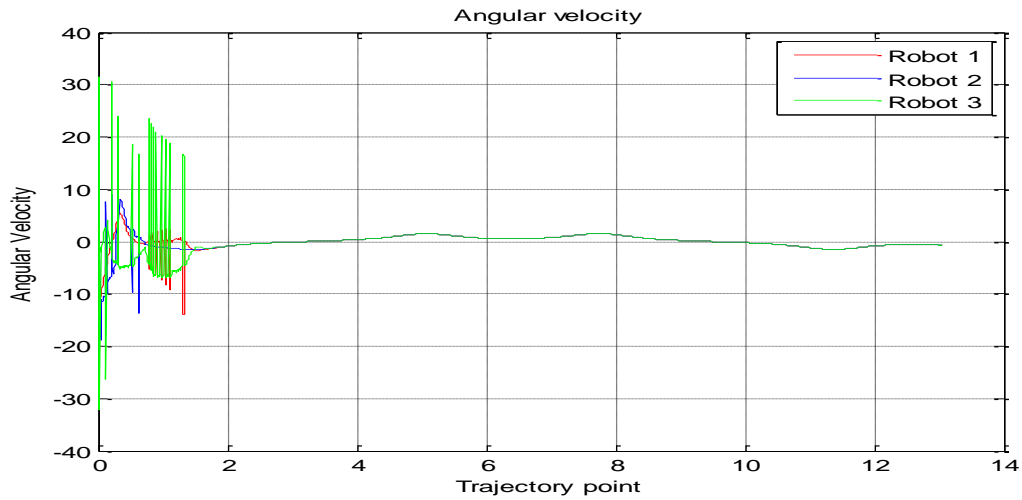
Proses perjalanan ketiga robot tidak menghasilkan nilai *linear velocity* yang relatif sama karena pada proses simulasi potensi tabrakan yang muncul antar robot dapat dihindarkan. Hasil *linear velocity multi-robot* ditunjukkan pada Gambar 4.25



Gambar 4.24 Perubahan Sudut Orientasi *Multi-Robot*



Gambar 4.25 Perubahan *Linear Velocity Multi-Robot*



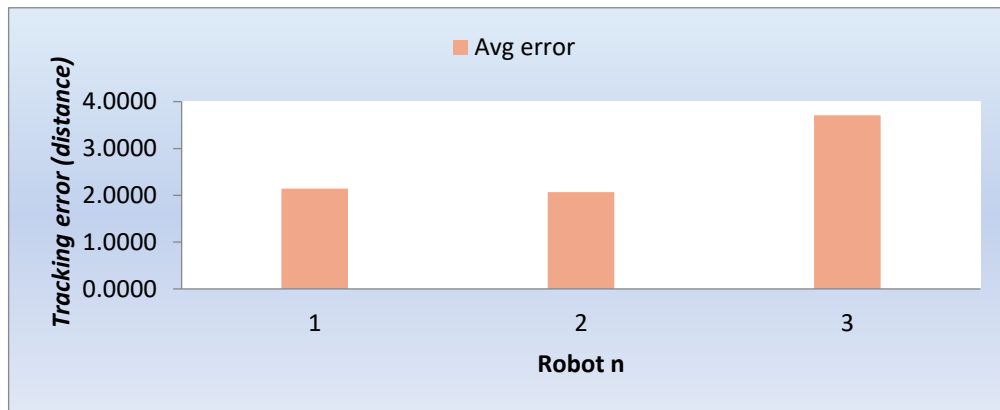
Gambar 4.26 Perubahan *Angular Velocity Multi-Robot*

Gambar 4.26 menjelaskan perubahan *angular velocity* pada pengujian simulasi *multi-robot* yang telah dilakukan. Hasil menjelaskan *angular velocity* pada robot yang relatif sama sehingga masing-masing robot dijalankan dengan *velocity* orientasi yang sama. Gambar 4.26 menjelaskan hasil *plotting angular velocity multi-robot*.

Berdasarkan dari tiga pengujian simulasi robot menggunakan *collision avoidance* dapat diperoleh nilai rata-rata *tracking error* pergerakan robot dalam misi perjalanan berdasarkan *refrence trajectory* yang ditunjukkan pada Tabel 4.1 dan Gambar 4.27.

Tabel 4.1 Rata-Rata *Tracking Error*

No	<i>Tracking Error Robot 1</i>	<i>Tracking Error Robot 2</i>	<i>Tracking Error Robot 3</i>
1	2.1364	2.0569	3.064
2	2.1565	2.0706	4.2636
3	2.1316	2.067	3.7981
Σ	2.1415	2.0648	3.7086

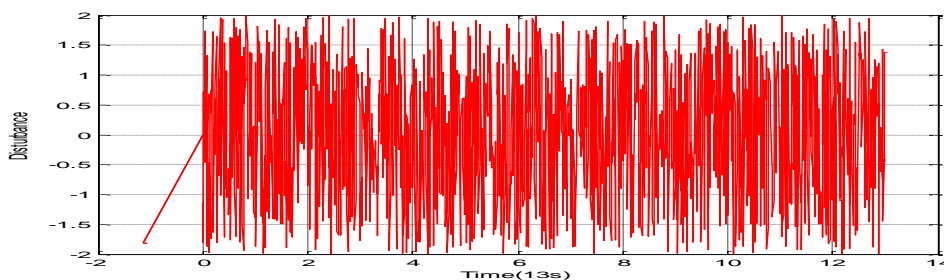


Gambar 4.27 Rata-Rata *Tracking Error*

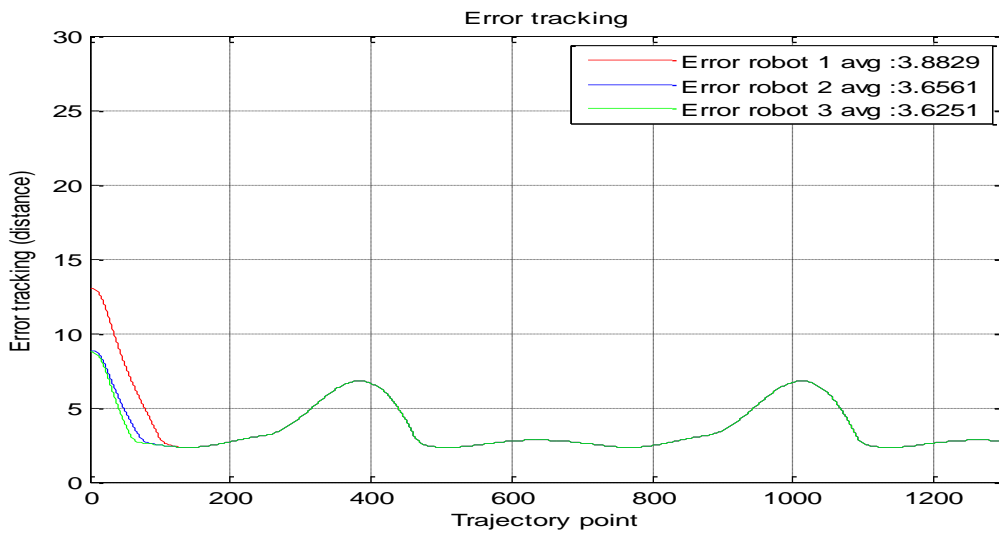
4.8 Pengujian Simulasi Robot Menggunakan *Collision Avoidance* dengan *Disturbance* $\beta = 60^\circ$

Pengujian *multi-robot* menggunakan *collision avoidance* robot dengan dengan *disturbance* yang disisipkan pada *velocity controller*. Gambar 4.28 menunjukkan sinyal *disturbance* yang dibangkitkan secara random.

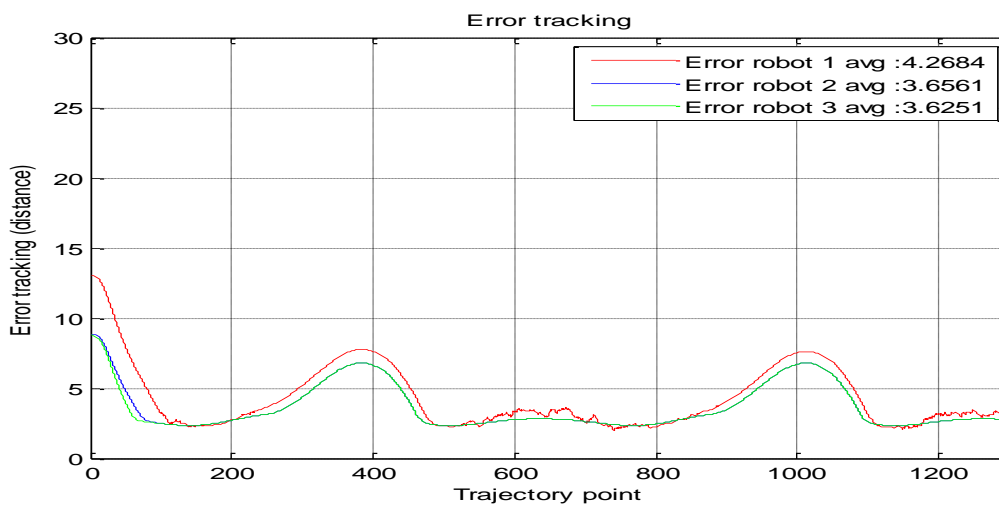
Pengujian yang dilakukan adalah untuk mendapatkan perbandingan antara tanpa *disturbance* dan dengan adanya *disturbance*. Pada *plotting* perbandingan yang dihasilkan penulis ingin mengetahui efek yang ditimbulkan dengan adanya *disturbance*. Untuk membangkitkan *disturbance* menggunakan sinyal random terhadap fungsi waktu dengan simpangan dari -2 hingga 2. Gambar 4.29 dan Gambar 4.30 merupakan perbandingan *tracking error* pada *multi-robot* tanpa *disturbance* dan dengan *disturbance*. Hasil *tracking error* menunjukkan perbedaan *error* tanpa *disturbance* dan dengan adanya *disturbance*. Dengan adanya *disturbance* menghasilkan *tracking error* yang lebih besar.



Gambar 4.28 Ketika diberi *Disturbance*

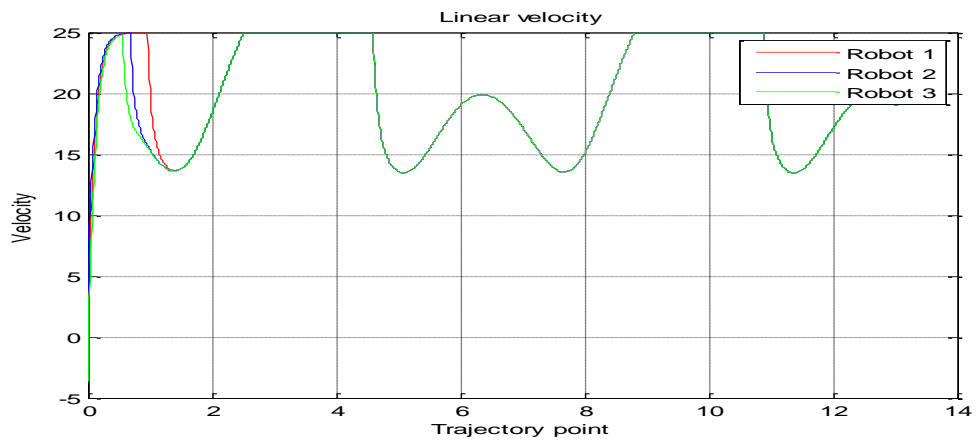


Gambar 4.29 *Tracking Error* tanpa *Disturbance*

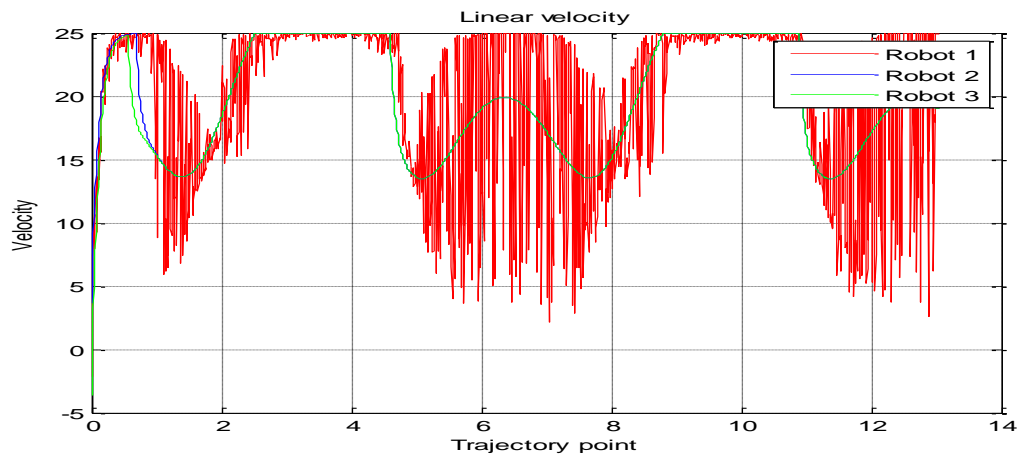


Gambar 4.30 *Tracking Error* dengan *Disturbance*

Pada hasil perbandingan *linear velocity* tanpa *disturbance* dan dengan *disturbance* menunjukkan perbedaan laju perubahan *velocity*, dimana *linear velocity* dengan *disturbance* mengalami osilasi. Sementara tanpa adanya *disturbance linear velocity* tidak mengalami osilasi yang dijelaskan pada Gambar 4.31 dan 4.32.

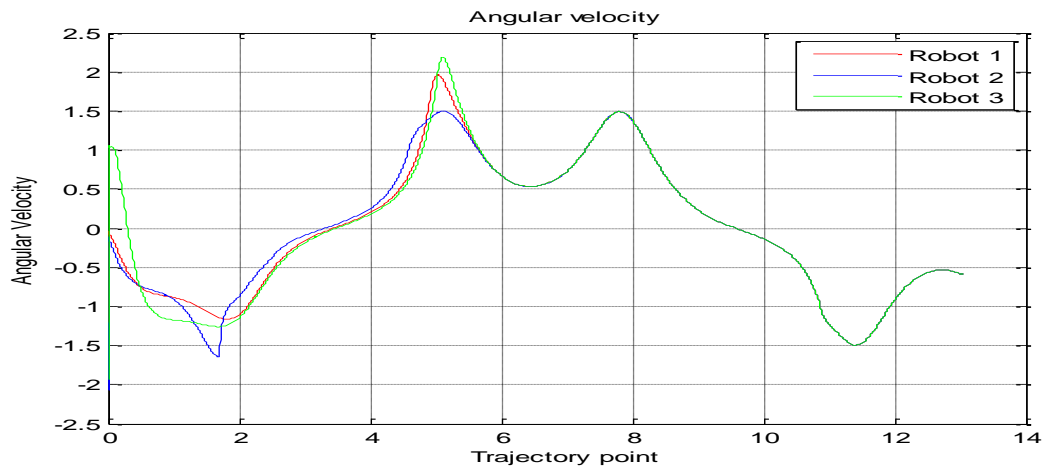


Gambar 4.31 *Linear Velocity* tanpa *Disturbance*

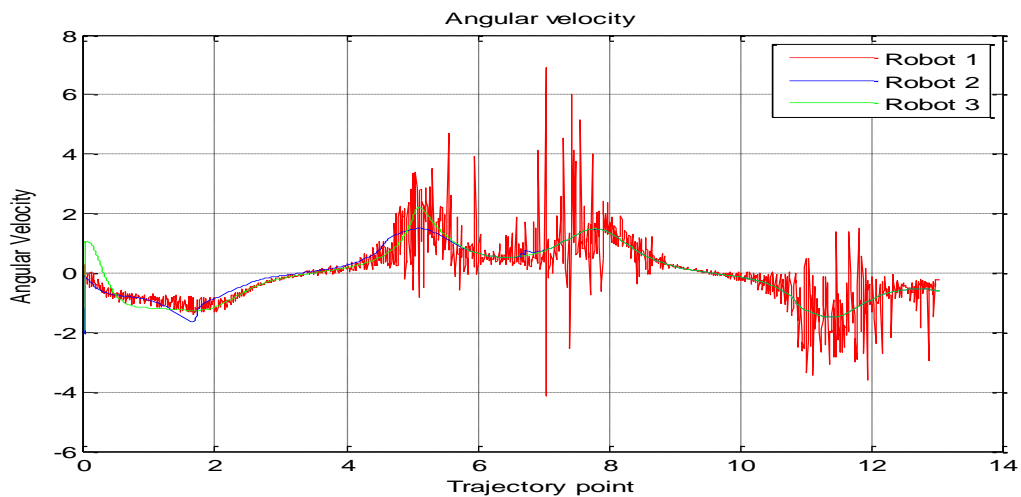


Gambar 4.32 *Linear Velocity* dengan *Disturbance*

Pada hasil perbandingan *angular velocity* tanpa *disturbance* dan dengan *disturbance* menunjukkan perbedaan laju *angular velocity*, dimana *angular velocity* dengan *disturbance* mengalami osilasi. Sementara tanpa adanya *disturbance* *angular velocity* tidak mengalami osilasi yang ditunjukkan pada Gambar 4.33 dan Gambar 4.34.



Gambar 4.33 *Angular Velocity* tanpa *Disturbance*



Gambar 4.34 *Angular Velocity* dengan *Disturbance*

Perbandingan dari hasil penelitian sebelumnya pada desain tesis ini ditunjukkan dengan *disturbance* pada *leader* (robot 1) dengan penelitian sebelumnya [2]. *Disturbance* berupa sinyal random atau *noise*. Pada penelitian sebelumnya, menggunakan nilai $\beta=60^\circ$ pada *multi-robot nonholonomic* dan belum diuji *tracking error* apabila terdapat *noise* pada *leader*. Perbandingan tersebut dapat dilihat pada Tabel 4.2 bahwa robot tanpa *disturbance* memiliki *tracking error* yang lebih kecil. Tetapi, *tracking error* pada robot *followers* memiliki hasil yang sama. Hal ini membuktikan bahwa *cluster space* dengan kontrol *escaping singularity* dapat mencegah tabrakan pada *multi-robot* meskipun terdapat *noise* pada robot *leader*.

Tabel 4.2 Perbandingan *Tracking Error* tanpa *Disturbance* dan dengan *Disturbance* $\beta=60^\circ$

Robot	<i>Tracking Error</i> tanpa <i>Disturbance</i>	<i>Tracking Error</i> dengan <i>Disturbance</i>
1	3.8829	4.2684
2	3.6561	3.6561
3	3.6251	3.6251
Σ	3.7213	3.8498

Halaman ini sengaja dikosongkan

BAB 5

PENUTUP

Pada sub bab ini akan dijelaskan mengenai kesimpulan dan saran berdasarkan pengujian dan analisa *trajectory tracking* pada *non-holonomic multi-robot*.

5.1 Kesimpulan

Kesimpulan-kesimpulan yang diperoleh dapat disebutkan, yaitu :

1. Hasil pengujian yang telah dilakukan menunjukkan kontrol *cluster space* yang diterapkan dapat meningkatkan kontrol yang baik dan hasil kontrol formasi pergerakan robot dengan *tracking error* keseluruhan dengan nilai rata-rata untuk tanpa *disturbance* sebesar 3.7213 dan dengan *disturbance* sebesar 3.8498.
2. Tanpa adanya *collision avoidance* pada konsep *multi-robot* mengakibatkan robot tidak dapat melanjutkan perjalanan sesuai *reference trajectory* jika terjadi tabrakan. Hal ini disebabkan robot tidak memiliki kemampuan untuk mengatur ulang lintasan dalam menyelesaikan permasalahan *collision avoidance*.
3. Spesifikasi kontrol formasi yang diterapkan pada *non-holonomic multi-robot* mampu membentuk formasi dengan 1 *leader* dan 2 *followers*.

5.2 Saran

Berdasarkan hasil penelitian *trajectory tracking non-holonomic multi-robot* yang telah dilakukan dapat diberikan saran untuk pengembangan sistem lebih lanjut. Saran-saran tersebut diantaranya :

1. Mengembangkan sistem *multi-robot* dengan formasi yang dapat diatur.
2. Mengembangkan kontrol *dynamics* dan *kinematics* yang lebih baik dengan kontrol yang mampu beradaptasi dengan berbagai variasi *tracking error*.

Halaman ini sengaja dikosongkan

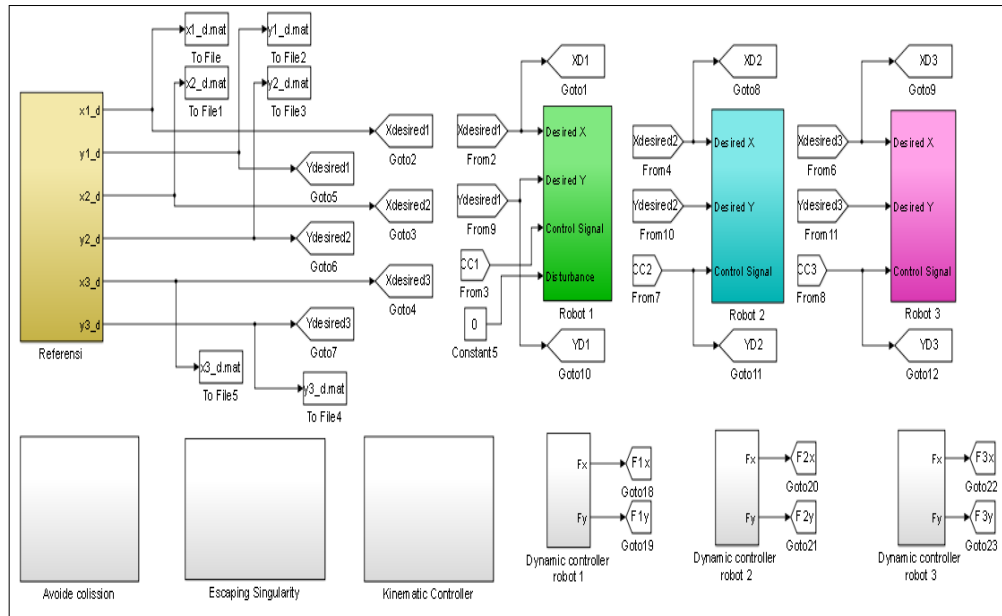
DAFTAR PUSTAKA

- [1] N.Mohammad and E.Sami, "Adaptive Sliding-Mode Cluster Space Control of a non-holonomic multi-robot system with Applications", *IET Control Theory Appl.*, Vol. 11 Iss. 8, pp. 1264-1273, 2017.
- [2] I. Mas, O. Petrovic and C. Kitts, "Error Characterization in the Vicinity of Singularities in Multi-Robot Cluster Space Control", *Proceedings of the 2008 IEEE International Conference on Robotics and Biomimetics Bangkok, Thailand*, February 21 - 26, 2009.
- [3] A.Shenawy, A.Wagner and E.Badreddin, "Solving The Singularity Problem For a Holonomic Mobile Robot", *IFAC (International Federation of Automatic Control)*, B6 23-29, Automation Laboratory, University of Mannheim, Germany, 2006.
- [4] P.Arga, A.Trihastuti, E.Rusdhianto, "Reinforcement Point and Fuzzy Input Design of Fuzzy Q-Learning for Mobile Robot Navigation System", *Institut Teknologi Sepuluh Nopember, Surabaya*, 2019.
- [5] K.Arya and A.Trihastuti, "Disturbance Compensation Using CTC with NDOB for Formation Control of Mobile Robots", *Institut Teknologi Sepuluh Nopember, Surabaya*, 2017.
- [6] M.Miaomiao, H.Chen, X.Liu, "Tracking and Stabilization Control of WMR by Dynamic Feedback Linearization", *Chinese Control and Decision Conference (CCDC), China*, 2011.
- [7] W.Gang, Z.Chenghui, Y.Yu and L.Xiaoping, "Adaptive Sliding Mode Trajectory Tracking Control for WMR Considering Skidding and Slipping via Extended State Observer", *School of Automation, Beijing University of Posts and Telecommunications, Beijing 100876, China*, 2019.
- [8] C. A. Kitts and I. Mas, "Cluster Space Specification and Control of Mobile Multirobot Systems", *IEEE/ASME Transactions on Mechatronics*, 14(2), pp. 207-218, April 2009.
- [9] C. A. Kitts and M. Egerstedt, "Design, Control, and Application of Real-World Multirobot Systems", *IEEE Robotics & Automation Magazine*, 15(1), p. 8, March 2008.
- [10] A.Reza, "Geometric Jacobians Derivation and Kinematic Singularity Analysis for Smokie Robot Manipulator & the Barrett WAM", *5th International Conference on Robotics and Mechatronics (ICROM)*, 1 Tehran, Iran, 2017.
- [11] I.Mas and C.A Kitts, "Dynamic Control of Mobile Multirobot Systems: The Cluster Space Formulation", *IEEE Access*, 2, pp. 558-570, 2014.

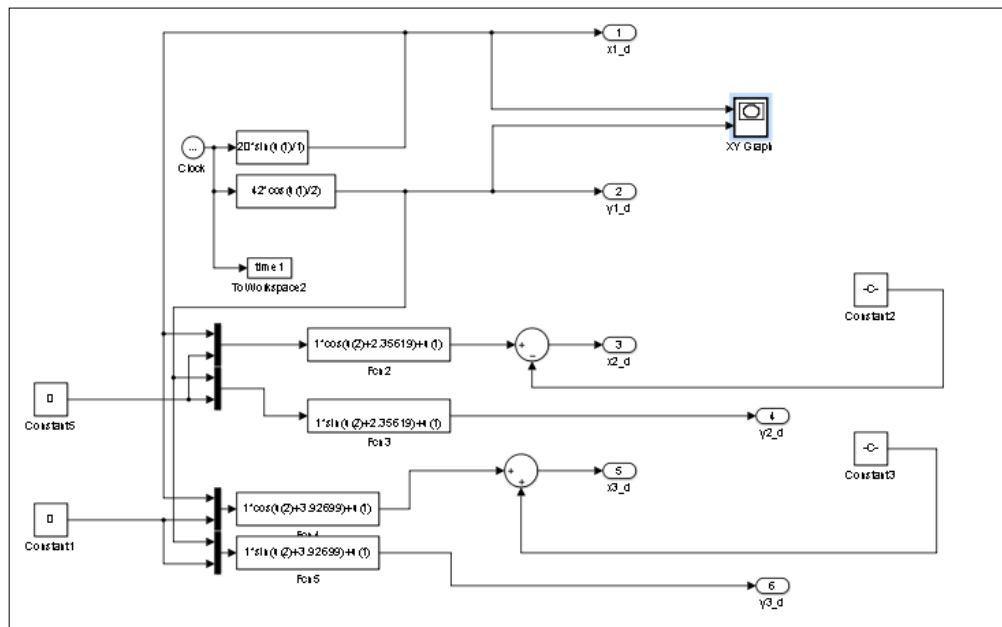
- [12] R. Ishizu and C.A.Kitts, "The Design, Simulation and Implementation of Multi-Robot Collaborative Control from Cluster Perspective", Santa Clara Univ. Master of Science in Elec. Eng. Thesis, Dec 2005.
- [13] I. Mas, O. Petrovic and C. Kitts,"Cluster Space Specification and Control of a 3-Robot Mobile System",IEEE International Conference on Robotics and Automation, pp.3763-3768, May 2008.
- [14] D.Abhijit and M.Swarnendu,"Kinematic Analysis of Wheeled Mobile Robots", National Institute of Technolog, Rourkela, 2007.

LAMPIRAN

1. Blok Simulink Permodelan *Multi-Robot*



2. Blok Simulink *Reference Trajectory*



3. Kode Program Matlab *Plotting*.

```
[a b] = size(x1);
dist = load('dist.mat');
figure(1)
plot(dist.val(1,:),dist.val(2,),'-r','LineWidth',1.0);
hold on;
grid on;
ylabel('Disturbance')
xlabel('Time(13s)')
error1 = [];
error2 = [];
error3 = [];
index = [];
avg_err1 = 0;
avg_err2 = 0;
avg_err3 = 0;
for i=1:1:a
    x_dev = (x1_d.val(2,i)-x1(i,2))^2;
    y_dev = (y1_d.val(2,i)-y1(i,2))^2;
    dev = x_dev + y_dev;
    error1(i) = sqrt(dev);
    avg_err1 = avg_err1 + error1(i);
    x_dev = (x2_d.val(2,i)-x2(i,2))^2;
    y_dev = (y2_d.val(2,i)-y2(i,2))^2;
    dev = x_dev + y_dev;
    error2(i) = sqrt(dev);
    avg_err2 = avg_err2 + error2(i);
    %plot(i,error_loc,'--b','LineWidth',3);
    %hold on;
    x_dev = (x3_d.val(2,i)-x3(i,2))^2;
    y_dev = (y3_d.val(2,i)-y3(i,2))^2;
    dev = x_dev + y_dev;
    error3(i) = sqrt(dev);
    avg_err3 = avg_err3 + error3(i);
    %plot(i,error_loc,'--g','LineWidth',3);
    %hold on;
    index(i) = i;
end
numerr1 = avg_err1/a;
numerr2 = avg_err2/a;
numerr3 = avg_err3/a;
figure(2)
plot(index,error1,'-r','LineWidth',1.0);
hold on;
plot(index,error2,'-b','LineWidth',1.0);
hold on;
plot(index,error3,'-g','LineWidth',1.0);
hold on;
xlabel('Trajectory point')
ylabel('Error tracking (distance)')
```

```

axis([0 a 0 30]) % figure limits
%axis square
grid on;
title('Error tracking');
legend(['Error robot 1 avg :' num2str(numerr1)], ['Error
robot 2 avg :' num2str(numerr2)], ['Error robot 3 avg :'
num2str(numerr3)]);
figure(3)
plot(tout, th1(:,2), '-r', 'LineWidth', 1);
hold on;
plot(tout, th2(:,2), '-b', 'LineWidth', 1);
hold on;
plot(tout, th3(:,2), '-g', 'LineWidth', 1);
hold on;
grid on;
title('Angle');
legend('Robot 1', 'Robot 2', 'Robot 3');
xlabel('Trajectory point')
ylabel('Angle (rad)')
figure(4)
plot(tout, v1(:,2), '-r', 'LineWidth', 1);
hold on;
plot(tout, v2(:,2), '-b', 'LineWidth', 1);
hold on;
plot(tout, v3(:,2), '-g', 'LineWidth', 1);
hold on;
grid on;
title('Linear velocity');
legend('Robot 1', 'Robot 2', 'Robot 3');
xlabel('Trajectory point')
ylabel('Velocity')

```

4. Kode Program Matlab *Draw robot*.

```

function Robot(a, x1, x2, x3, y1, y2, y3, th1, th2, th3)
for i=1:1:a
    theta1 = th1(i,2);
    theta2 = th2(i,2);
    theta3 = th3(i,2);
    plot(x1(i,2), y1(i,2), '-r', 'LineWidth', 1.5);
    plot(x2(i,2), y2(i,2), '-b', 'LineWidth', 1.5);
    plot(x3(i,2), y3(i,2), '-g', 'LineWidth', 1.5);
    [pline1, fillline1] =
draw_robot(theta1, x1(i,2), y1(i,2), 'r');
    [pline2, fillline2] =
draw_robot(theta2, x2(i,2), y2(i,2), 'b');
    [pline3, fillline3] =
draw_robot(theta3, x3(i,2), y3(i,2), 'y');

```

Halaman ini sengaja dikosongkan

BIOGRAFI PENULIS



Penulis bernama lengkap Anggitasari Putri Widanis, dilahirkan di Grobogan 19 November 1994. Penulis menempuh pendidikan di SDN 04 Purwodadi pada tahun 2001. Tahun 2007 penulis melanjutkan pendidikan di SMPN 01 Purwodadi. Kemudian pada tahun 2010 masuk ke SMAN 01 Purwodadi. Melanjutkan kuliah pada jenjang Diploma IV Program Studi Instrumentasi & Elektronika dengan konsentrasi Teknik Instrumentasi Kilang di STEM Akamigas Cepu angkatan 2013. Setelah lulus pada tahun 2017, Pada tahun yang sama, penulis diterima di Program Magister di Jurusan Teknik Elektro, Institut Teknologi Sepuluh Nopember dengan bidang keahlian Teknik Sistem Pengaturan. Pada saat ini, penulis sedang menyelesaikan studi magisternya pada bulan Januari 2020.