TESIS – EE185401

PENCARIAN JALUR TERPENDEK PADA 3D *VISIBILITY ROADMAP* DENGAN PENGHALUSAN JALUR BERDASARKAN KEMAMPUAN MANUVER *FIXED-WING*

DILA MARTA PUTRI
07111850020001

DOSEN PEMBIMBING
Dr. Trihastuti Agustinah, ST., MT.

PROGRAM MAGISTER
BIDANG KEAHLIAN TEKNIK SISTEM PENGATURAN
DEPARTEMEN TEKNIK ELEKTRO
FAKULTAS TEKNOLOGI ELEKTRO DAN INFORMATIKA CERDAS
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA
2022

TESIS – EE185401

PENCARIAN JALUR TERPENDEK PADA 3D *VISIBILITY ROADMAP* DENGAN PENGHALUSAN JALUR BERDASARKAN KEMAMPUAN MANUVER *FIXED-WING*

DILA MARTA PUTRI
07111850020001

DOSEN PEMBIMBING
Dr. Trihastuti Agustinah, ST., MT.

PROGRAM MAGISTER
BIDANG KEAHLIAN TEKNIK SISTEM PENGATURAN
DEPARTEMEN TEKNIK ELEKTRO
FAKULTAS TEKNOLOGI ELEKTRO DAN INFORMATIKA CERDAS
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA
2022

LEMBAR PENGESAHAN TESIS

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

Oleh
DILA MARTA PUTRI
NRP: 07111850020001

Tanggal Ujian: 13 Januari 2022
Periode Wisuda: Maret 2022

Disetujui oleh
Pembimbing:

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

Penguji:

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

2. Dr.Ir. Ari Santoso, DEA.
   NIP: 19660218 199102 1001

Kepala Departemen Teknik Elektro
Dedet Candra Riawan, S.T., M.Eng., Ph.D.
NIP: 197311192000031001

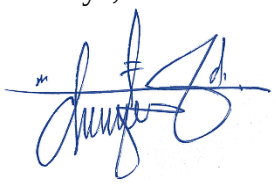
Halaman ini sengaja dikosongkan

PERNYATAAN KEASLIAN TESIS

Dengan ini saya menyatakan bahwa isi keseluruhan Tesis saya dengan judul "**PENCARIAN JALUR TERPENDEK PADA 3D *VISIBILITY ROADMAP* DENGAN PENGHALUSAN JALUR BERDASARKAN KEMAMPUAN MANUVER *FIXED-WING***" adalah benar-benar 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.

Surabaya, 20 Januari 2022



Dila Marta Putri

NRP. 07111850020001

Halaman ini sengaja dikosongkan

PENCARIAN JALUR TERPENDEK PADA 3D *VISIBILITY ROADMAP* DENGAN PENGHALUSAN JALUR BERDASARKAN KEMAMPUAN MANUVER *FIXED-WING*

Nama mahasiswa : Dila Marta Putri
NRP : 07111850020001
Pembimbing : Dr. Trihastuti Agustinah, S.T., M.T

ABSTRAK

Penelitian ini membahas perencanaan jalur untuk *fixed-wing* pada lingkungan 3D statis. Lingkungan perkotaan seperti gedung-gedung yang berbentuk kubus diadaptasi untuk merepresentasikan lingkungan 3D. Algoritma *visibility roadmap* merupakan algoritma yang digunakan untuk membuat lingkungan 3D pada penelitian ini. Algoritma ini menyediakan *node* dan *edge* yang digunakan oleh Theta* sebagai algoritma pencari jalur terpendek setelah diberikan koordinat awal dan tujuan di lingkungan 3D. Jalur terpendek yang didapatkan akan dihaluskan menggunakan metode perencanaan *smoothing path* berbasis Bézier *curve* yang memenuhi batasan yang diusulkan, yaitu berdasarkan koridor keselamatan dan kemampuan manuver dari *fixed-wing*. Hasil metode ini akan dibandingkan dengan penelitian sebelumnya dengan menerapkan kriteria komparatif yang relevan. *Smoothing path* yang dihasilkan diuji dengan menerbangkan *fixed-wing* pada jalur tersebut. Hasil simulasi menunjukkan bahwa metode yang diusulkan lebih aman dibandingkan metode sebelumnya dengan jarak terkecil ke halangan sebesar 4.73 m dan jalur tersebut dapat dilalui oleh *fixed-wing* secara akurat.

Kata kunci: Theta*, *fixed-wing*, Bézier *curve*, manuver, *visibility roadmap*

Halaman ini sengaja dikosongkan

SHORTEST PATH SEARCH ON 3D VISIBILITY ROADMAP WITH SMOOTHING PATH BASED ON FIXED-WING MANEUVERABILITY

By : Dila Marta Putri
Student Identity Number : 07111850020001
Supervisor : Dr. Trihastuti Agustinah, S.T., M.T

ABSTRACT

This research discusses path planning for fixed-wing in a static 3D environment. Urban environments such as cube-shaped buildings are adapted to represent 3D environments. The visibility roadmap algorithm is the algorithm used to create a 3D environment in this research. This algorithm provides nodes and edges which are used by Theta* as the shortest pathfinding algorithm after being given the initial and destination coordinates in a 3D environment. The shortest path obtained will be smoothed using a Bézier curve-based smoothing path planning method that meets the proposed constraints, namely based on the safety corridor and maneuverability of the fixed-wing. The results of this method are compared with previous studies by applying the relevant comparative criteria. The resulting smoothing path is tested by flying a fixed-wing on the path. The simulation results show that the proposed method is safer than the previous method with the smallest distance to the obstacle of 4.73 m and the path can be traversed by fixed-wing accurately.

Key words: Theta*, fixed-wing, Bézier curve, maneuver, visibility roadmap

Halaman ini sengaja dikosongkan

KATA PENGANTAR

Alhamdulillah, segala puji syukur pada kehadirat Allah SWT, atas segala karunia dan ridho-NYA, sehingga tesis dengan judul "**Pencarian Jalur Terpendek pada 3D *Visibility Roadmap* dengan Penghalusan Jalur berdasarkan Kemampuan Manuver *Fixed-Wing***" ini dapat diselesaikan.

Tesis ini disusun untuk memenuhi salah satu persyaratan memperoleh gelar Magister Teknik (M.T.) dalam bidang studi Sistem Pengaturan, Jurusan Teknik Elektro, Institut Teknologi Sepuluh Nopember.

Segala ucapan terima kasih penulis ucapkan kepada Ayahanda Syamsul Afri dan Ibunda Yasnar atas doa, semangat, kepercayaan dan kasih sayang yang tak terbatas pada penulis, kepada Ibu Dr. Trihastuti A., atas motivasi, bimbingan, nasihat, kesabaran, arahan dan waktu yang telah diluangkan kepada penulis untuk berdiskusi mulai awal perkuliahan hingga teselesaikannya Tesis ini, kepada teman-teman pascasarjana fakultas teknik elektro dan para Tendik yang telah membantu selama masa perkuliahan.

Penulis hanya percaya bahwa **"Usaha Tidak akan Mengkhianati Hasil"**. Maka dari itu, penyelesaian Tesis ini juga tak lepas dari segala usaha dan doa penulis serta bantuan dari berbagai pihak. Semoga Tesis ini bermanfaat bagi pembaca terutama untuk penulis.

Surabaya, 20 Januari 2022
Penulis

Halaman ini sengaja dikosongkan

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

Halaman ini sengaja dikosongkan

BAB 1
PENDAHULUAN

1.1 Latar Belakang

Tesis ini membahas tentang masalah perencanaan jalur 3D untuk *fixed-wing* di lingkungan statis. Adapun kendala yang banyak ditemukan dalam masalah perencanaan jalur adalah bagaimana menentukan metode yang efisien untuk menentukan jalur yang aman dan dapat dilalui *fixed-wing* yang bergerak di lingkungan 3D yang kompleks. Penghindaran halangan juga merupakan permasalahan yang biasa ditemukan dalam penelitian bidang ini, seperti halnya untuk robot otonom di lingkungan 2D, *aircraft* dan UAV di lingkungan 3D. Oleh karena itu banyak metode atau algoritma yang telah dikembangkan untuk menyelesaikan permasalahan tersebut.

Algoritma yang paling sering digunakan di antaranya adalah algoritma *rapidly exploring random tree* (RRT) [1] dan *probabilistic road-maps* (PRM) [2], dimana metode acak dan stokastik digunakan untuk menghasilkan *waypoint* dan jalur pada metode ini. Pada umumnya, pemilihan jalurnya menggunakan teknik *machine learning* seperti *evolutionary algorithm* (EA) [3], *particle swarm optimization* (PSO)[4], dan *ant colony algorithm* (ACO) [5]. Metode-metode ini awalnya banyak di desain untuk perencanaan jalur 2D dan optimalisasinya tergantung dari durasi *running* dari algoritma *machine learning* yang dipilih.

Metode yang tepat untuk menemukan jalur dalam skala titik yang besar dan resolusi tinggi pada lingkungan 3D di antaranya adalah metode Silhouette [6], *navigation mesh* [7], *visibility graph* [8], dan *uniform grid* [9]. Semua metode pembuatan jalur tersebut dapat menghasilkan perubahan *heading* yang besar kecuali *visibility graph*. Metode-metode yang telah disebut dapat menghasilkan *waypoint* dan grafik lintasan yang dapat dioptimalkan dengan algoritma pencarian jalur seperti A*[10]. Jalur yang dihasilkan oleh algoritma pencarian jalur ini tidak cukup halus karena masih adanya perubahan *heading* yang tajam. Oleh karena itu, algoritma ini perlu perbaikan untuk membuat jalur yang halus, sehingga memudahkan UAV untuk melintasi jalur tersebut, terutama *fixed-wing* yang memiliki belokan minimum.

Ide tesis ini didapat berdasarkan penelitian [9][11][12], dimana pada [9] peneliti membandingkan dua algoritma heuristik yaitu A* dan Theta*. Hasil pada penelitian tersebut menunjukkan bahwa performa algoritma Theta* memberikan nilai terbaik dalam menemukan jalur terpendek dibandingkan dengan algoritma A*. Peneliti mengujikan kelayakan algoritma ini di lingkungan perkotaan yang sederhana yang memiliki banyak data *node* dari *grid* yang diberikan, sehingga algoritma pencari membutuhkan waktu komputasi yang lama. Oleh karena itu untuk membuat lingkungan perkotaan dengan skala yang besar serta data *node* yang lebih sedikit, maka digunakan algoritma *visibility roadmap* seperti yang telah dilakukan oleh [11].

Peneliti [11] memodelkan lingkungan 3D statis menggunakan algoritma *visibility roadmap*. Algoritma ini memodelkan data ketinggian peta dengan satu set *convex obstacle* dengan menghitung *convex hull*. Algoritma *visibility roadmap* ini juga efektif dalam penghindaran halangan, karena peneliti dapat memodelkan *node* dengan jarak yang aman. Algoritma *visibility graph* ini juga memiliki kelebihan yaitu dapat mengontrol konsumsi energi seperti yang telah dilakukan oleh penelitian [11] dengan menambahkan algoritma *bounded space* pada *visibility roadmap*.

Pada penelitian [11], algoritma pembuatan *node* belum mempertimbangkan kinematika dari robot, sehingga membutuhkan proses *smoothing* untuk merelokasi urutan *node* untuk mendapatkan jalur yang dapat dilalui oleh *fixed-wing*. Sebuah solusi yang dapat digunakan untuk memperhalus jalur sesuai dengan radius belok dan tingkat batasan pendakian adalah penggunaan Bézier *curve*. Ini adalah solusi saat ini yang dipakai sebagai *post-smoother* dalam perencanaan jalur yang dikembangkan oleh [12]. Pada penelitian [12], proses penghalusan jalur menggunakan Bézier *curve* memberikan batasan aman tabrakan (*safety corridor)* untuk memperkirakan jalur baru yang dibuat di sekitar jalur asli. Oleh karena itu, pada tesis ini mengusulkan penambahan batasan baru yaitu menambahkan batasan aman manuver, di mana jalur yang dipilih juga berada dalam batasan manuver maksimum (*curvature-turn radius*) dari *fixed-wing*.

1.2 Rumusan Masalah

Rumusan masalah dari penelitian ini adalah bagaimana merancang perencanaan jalur dalam menemukan jalur terpendek pada *visibility roadmap* dengan menggunakan algoritma Theta* dan *smoothing path* menggunakan metode Bézier *curve* yang memenuhi batasan *safety corridor* dan kemampuan manuver dari *fixed-wing*.

1.3 Tujuan

Tujuan dari penelitian ini adalah merancang perencanaan *smoothing path* di sekitar jalur yang sudah ditentukan pada lingkungan 3D menggunakan metode Bézier *curve* yang memenuhi batasan *safety corridor* dan kemampuan manuver dari *fixed-wing*.

1.4 Batasan Masalah

Dalam penelitian ini terdapat batasan masalah untuk membatasi permasalahan yang muncul di antaranya:

1. *Fixed-wing* yang digunakan adalah tipe *fixed-wing* ultrastick-25,
2. pemilihan jalur tidak memperhitungkan konsumsi energi dari *fixed-wing*,
3. lingkungan perkotaan direpresentasikan sebagai blok-blok persegi,
4. bangunan/halangan yang dibuat untuk merepresentasikan lingkungan perkotaan terdiri dari 5, 9, 10, 15, 20 dan 25 bangunan, dan
5. nilai manuver diambil dari simulasi *fixed-wing*.

1.5 Kontribusi

Kontribusi dari penelitian ini adalah menambah batasan kemampuan manuver (*curvature - turn radius*) dari *fixed-wing* ultrastick-25 pada proses *smoothing path*.

Halaman ini sengaja dikosongkan

BAB 2
KAJIAN PUSTAKA

Pada bab ini akan dibahas tentang kajian penelitian terkait yang telah dilakukan dan menjadi referensi untuk membuat tesis yang akan dilakukan. Selain itu, juga terdapat penjelasan tentang metode atau semua teori yang akan digunakan untuk keperluan dalam pembuatan tesis.

2.1 Kajian Penelitian Terkait

Pada penyusunan tesis ini, dilakukan kajian terhadap penelitian-penelitian terkait yang telah ada. Berbagai metode dan struktur kontrol yang digunakan pada setiap pustaka akan dipaparkan dalam bab ini, sehingga diperoleh ide dasar untuk penyusunan tesis.

2.1.1. *Path Planning Strategies for UAVs in 3D Environment* [9]

Paper ini membahas tentang strategi perencanaan jalur dan membandingkan dua algoritma heuristik yaitu Theta* (*basic version*) dan A* untuk lingkungan 3D. Kedua algoritma tersebut diujikan pada dua macam lingkungan statis yaitu halangan orografis dan lingkungan perkotaan (*urban environment*). Pemodelan lingkungan 3D menggunakan alat perencanaan berbasis MATLAB yaitu *Digital Elevation Maps* (DEMs) untuk meminimalkan benturan pada lingkungan orografis.

Algoritma A * secara iteratif mengevaluasi biaya pemindahan dari sel saat ini ke salah satu tetangganya melalui fungsi biaya yang ditentukan. Fungsi ini (F) diperoleh dengan menambahkan dua buah istilah:

- H proporsional dengan jarak estimasi heuristik dari sel yang dievaluasi ke tujuan.
- G proporsional dengan jarak dari sel saat ini ke yang dievaluasi.

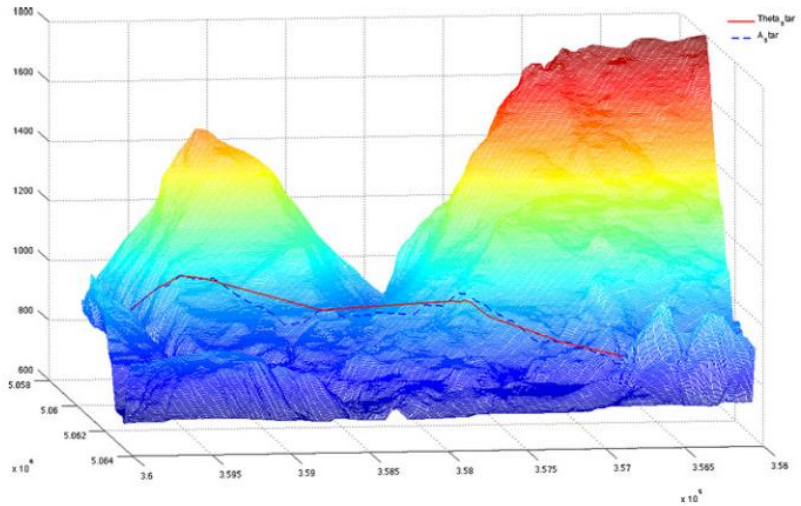
Nilai G adalah 0 untuk sel awal dan akan meningkat ketika algoritma memperluas sel-sel. Misal, pada setiap langkah algoritma menjumlahkan biaya pemindahan dari sel awal ke sel saat ini, jarak dari sel saat ini ke sel tetangganya.

Salah satu kelemahan terpenting dari algoritma A* berada pada batasan-batasan *heading* yang terhubung dengan karakteristik *grid*. Beberapa algoritma telah dikembangkan dari algoritma A*, salah satunya adalah Theta*.

Theta* merupakan salah satu algoritma yang menyempurnakan pencarian grafik yang mendapatkan jalur umum dengan *heading* yang lebih sedikit, karena Theta* dapat memlerluas jalur dengan pemeriksaan *line-of-sight*. Diketahui bahwa algoritma pencarian grafik ini memilih langkah per langkah dari satu *node* ke *node* berikutnya. Menentukan *parent* dari *node* sebelumnya hanya sampai posisi saat ini, dan *neighbour node* dievaluasi untuk langkah selanjutnya.

Pada saat algoritma Theta* memperluas pencarian, ini menghasilkan 2 tipe jalur yaitu dari *node* saat ini ke *neighbour* (seperti pada A*) dan dari *parent node* saat ini ke *neighbour*. Dibandingkan dengan A*, *parent* dari sebuah *node* tidak harus menjadi *neighbour* dari *node* tersebut selama ada garis pandang (LoS) yang menghubung antara kedua *node* tersebut, sehingga jalur yang ditemukan oleh Theta* merupakan solusi untuk menemukan jalur terpendek dan lebih mulus dari algoritma A*.

Kedua algoritma ini diujikan pada 2 lingkungan 3D yaitu halangan orografis yang dibuat menggunakan peta DEMs dan lingkungan perkotaan (*urban environment*) menggunakan GUI di MATLAB .



Gambar 2.1 Representasi halangan orografis 3D: Jalur 1

Jalur di dataran tinggi dibuat sebagai representasi dari halangan orografis, pada Gambar 2.1 menunjukkan jalur jarak menengah yang memiliki karakteristik *map* sebagai berikut:

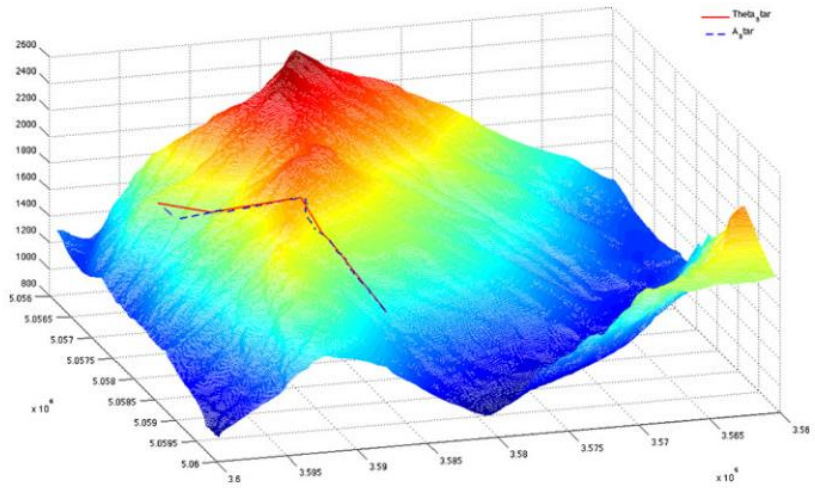
Jumlah node : 7,634,088

Δlat : 10 m

$\Delta long$: 10 m

ΔZ : 5 m

Dimensi matriks lingkungan : $357 \times 396 \times 54$ ($lat \times long \times Z$)



Gambar 2.2 Representasi halangan orografis 3D: Jalur 2

Jalur kedua yang dibuat di dataran tinggi merupakan jalur dari sebuah pemisahan halangan orografis (*orographic obstacle separation*) yang dapat dilihat pada Gambar 2.2 yang memiliki karakteristik *map* berikut:

Jumlah *node* : 16,727,040

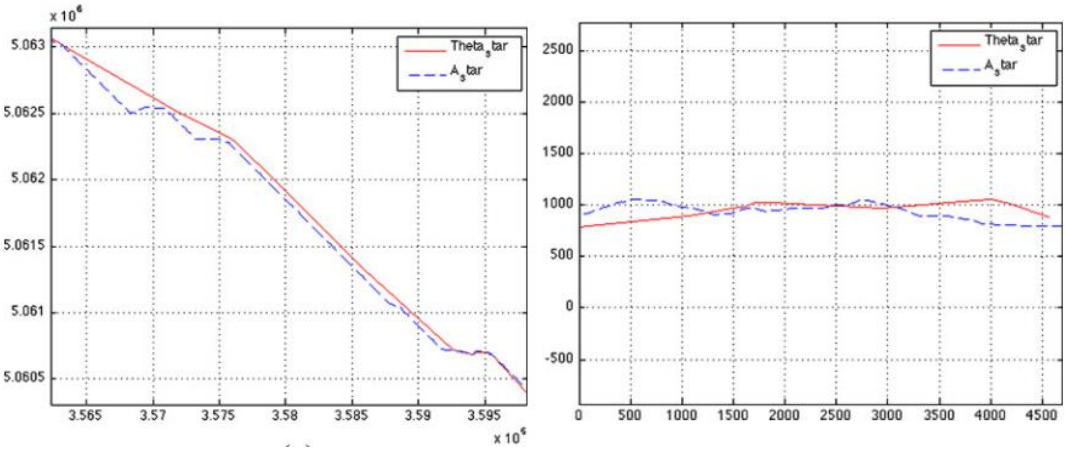
Δlat : 10 m

$\Delta long$: 10 m

ΔZ : 5 m

Dimensi matriks lingkungan : $384 \times 396 \times 110$ ($lat \times long \times Z$)

Hasil dari penelitian ini menunjukkan bahwa performansi dari algoritma Theta* lebih baik jika dibandingkan A* yang dapat dilihat pada Gambar 2.3 dan Gambar 2.4. Algoritma Theta* menghasilkan jalur yang lebih pendek, dapat mengurangi perubahan *heading* sebesar 69%, perubahan ketinggian sebesar 91% dan jumlah *path point* sebesar 93% untuk jalur 1 yang dapat dilihat pada Tabel 2.1. Untuk jalur 2, algoritma Theta* juga mampu menghasilkan jalur yang lebih pendek serta mampu mengurangi perubahan heading sebesar 86%, perubahan ketinggian sebesar dan jumlah *path point* masing-masing sebesar 95% yang dapat dilihat pada Tabel 2.2. Waktu komputasi yang dibutuhkan oleh Theta* sedikit lebih tinggi, namun Theta* tetap lebih unggul dilihat dari hasil jalur yang dihasilkan.

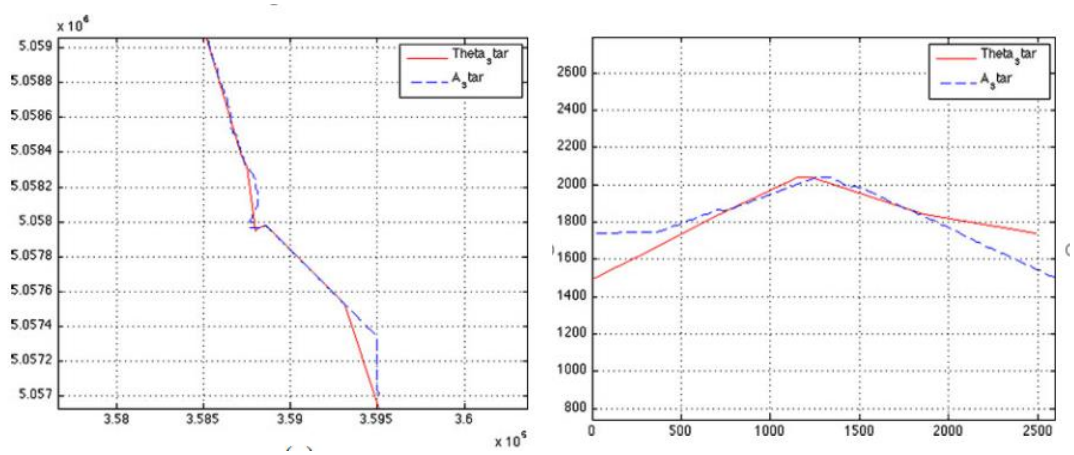


(a) (b)

Gambar 2.3 Representasi halangan orografis 3D: Jalur 1. (a) *longitude-latitude plane*. (b) *flight altitude*

Tabel 2.1 Perbandingan hasil halangan orografis: Jalur 1

Jalur 1	**A***	**Theta***
Panjang jalur (m)	4850	4618
Waktu komputasi (s)	1.203	1.393
Jumlah perubahan *heading*	42	13
Jumlah perubahan ketinggian	159	15
Jumlah *path point*	258	17



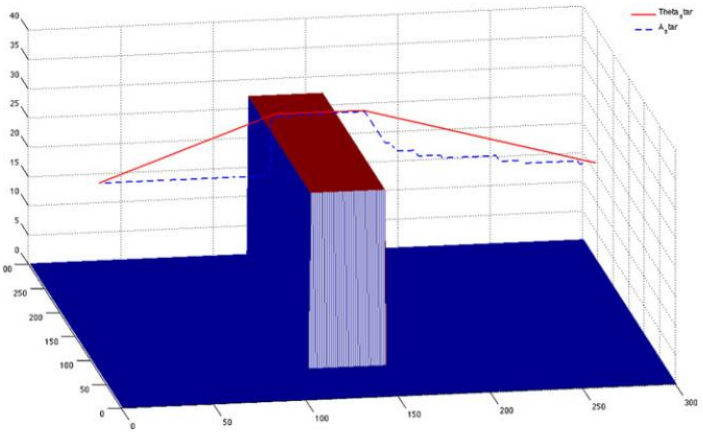
(a) (b)

Gambar 2.4 Representasi halangan orografis 3D: Jalur 2. (a) *longitude-latitude plane*. (b) *flight altitude*

Tabel 2.2 Perbandingan hasil halangan orografis: Jalur 2

Jalur 2	**A***	**Theta***
Panjang jalur (m)	2776	2653
Waktu komputasi (s)	1.622	1.638
Jumlah perubahan *heading*	66	9
Jumlah perubahan ketinggian	174	8
Jumlah poin jalur	220	11

Pengujian algoritma pada lingkungan perkotaan juga dibuat dalam dua bentuk jalur. Jalur pertama menempatkan satu bangunan di tengah jalur pencarian dapat dilihat pada Gambar 2.5. Sedangkan jalur kedua dibuat sebuah lingkungan dengan beberapa bangunan dapat dilihat pada Gambar 2.6. Pemodelan lingkungan untuk lingkungan perkotaan ini dibuat sangat sederhana, tetapi berguna untuk menguji algoritma dengan peta yang meniru karakteristik dari lingkungan kompleks.



Gambar 2.5 Lingkungan perkotaan 3D: Jalur 1

Jalur pertama yang dibuat di lingkungan perkotaan 3D yang dapat dilihat pada Gambar 2.5 memiliki karakteristik *map* seperti berikut:

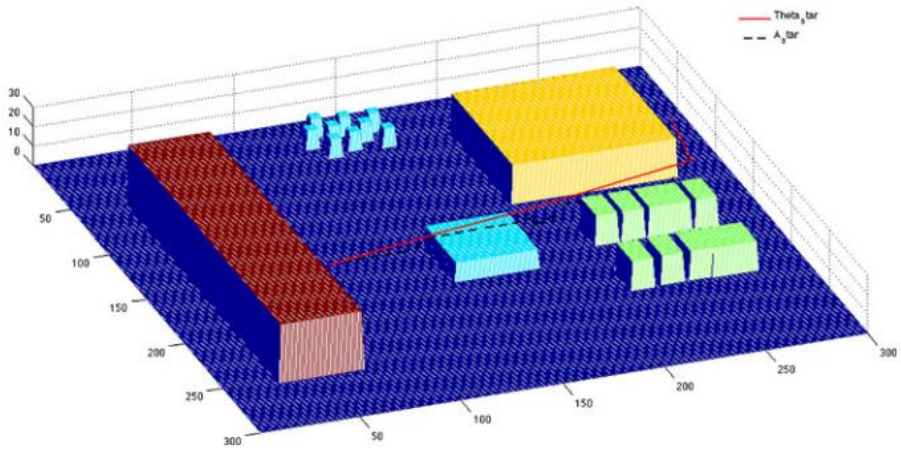
Jumlah *node* : 9,990,000

ΔX : 1 m

ΔY : 1 m

ΔZ : 0.5 m

Dimensi matriks lingkungan : $300 \times 300 \times 111$ $(lat \times long \times Z)$



Gambar 2.6 Lingkungan perkotaan 3D: Jalur 2

Jalur kedua dibuat pada lingkungan perkotaan 3D yang terdiri dari beberapa gedung yang terlihat pada Gambar 2.6 memiliki karakteristik *map* berikut:

Jumlah titik : 152,064

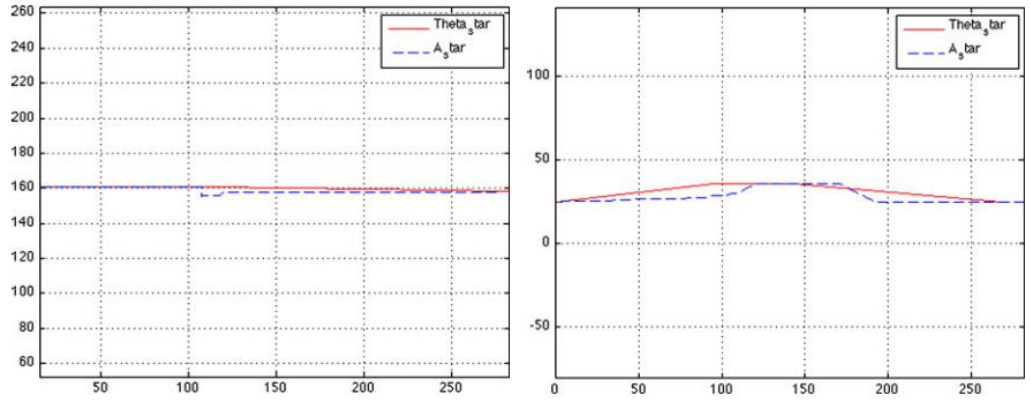
ΔX : 1 m

ΔY : 1 m

ΔZ : 0.5 m

Dimensi matriks lingkungan : $300 \times 300 \times 111$ $(lat \times long \times Z)$

Hasil dari pembuatan jalur pada lingkungan perkotaan 3D menunjukkan bahwa algoritma Theta* lebih baik jika dibandingkan A* yang dapat dilihat pada Gambar 2.7 dan Gambar 2.8. Algoritma Theta* menghasilkan jalur yang lebih pendek yaitu sebesar 269 m untuk jalur 1 dan 247 m untuk jalur 2, memiliki nilai *heading*, ketinggian dan jumlah *path point* yang jauh lebih kecil jika dibandingkan dengan performansi algoritma A* yang dapat dilihat pada Tabel 2.3 dan Tabel 2.4. Untuk waktu komputasi pada jalur 1, algoritma Theta* membutuhkan waktu yang lebih cepat dibandingkan A*. Pada jalur 2, algoritma Theta* waktu komputasi yang dihasilkan lebih lama dibandingkan dengan A*. Secara keseluruhan, algoritma Theta* tetap lebih unggul dilihat dari hasil jalur yang dihasilkan.

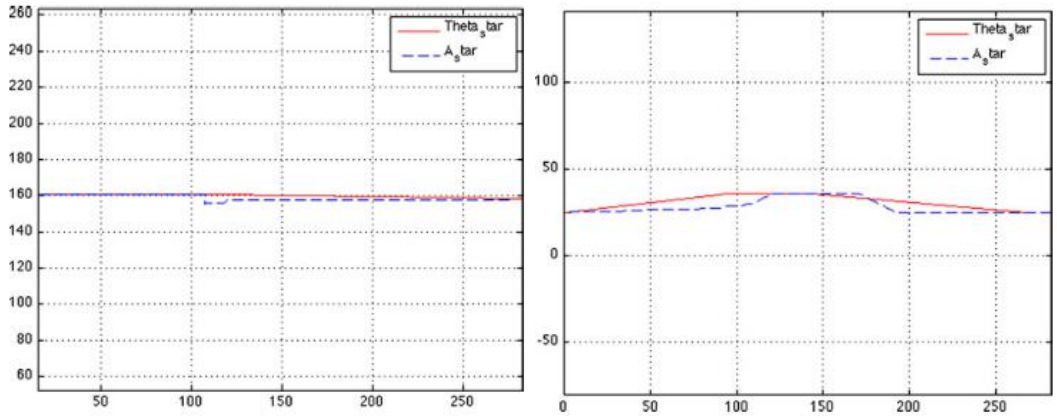


(a) (b)

Gambar 2.7 Lingkungan perkotaan 3D: Jalur 1. (a) x-y *plane*. (b) *flight altitude*

Tabel 2.3 Perbandingan hasil lingkungan perkotaan 3D: Jalur 1

Jalur 1	A*	Theta*
Panjang jalur (m)	287	269
Waktu komputasi (s)	5.718	3.081
Jumlah perubahan *heading*	15	2
Jumlah perubahan ketinggian	42	2
Jumlah *path point*	282	4



(a) (b)

Gambar 2.8 Lingkungan perkotaan 3D: Jalur 2. (a) x-y *plane*. (b) *flight altitude*

Tabel 2.4 Perbandingan hasil lingkungan perkotaan 3D: Jalur 2

Jalur 2	**A***	**Theta***
Panjang jalur (m)	264	247
Waktu komputasi (s)	1.047	1.176
Jumlah perubahan *heading*	14	4
Jumlah perubahan ketinggian	22	3
Jumlah *path point*	244	5

Perbandingan hasil algoritma A* dan Theta* yang dapat dilihat pada Tabel 2.1 – 2.4, dapat disimpulkan bahwa algoritma Theta* memiliki hasil yang lebih baik. Dari hasil simulasi, rata-rata waktu komputasi yang dibutuhkan untuk menemukan titik target sedikit lebih lama dibandingkan dengan algoritma A*.

Namun jalur yang dihasilkan lebih pendek, dengan perubahan *heading* dan perubahan ketinggian yang sedikit dibandingkan algoritma A*. Peneliti mengatakan bahwa *paper* ini memiliki beberapa kekurangan. pertama peneliti tidak mempertimbangkan kinematika UAV sebagai bagian dalam *path generation* dan ini menjadi masalah utama untuk kendaraan *nonholonomic* yang membutuhkan proses perataan untuk merealokasi urutan *node* yang membentuk *path* yang dapat diterbangkan. Oleh karena itu peneliti menyarankan menambahkan sebuah algoritma untuk *safety* dan *smooth* untuk penerbangan. Kedua, karena pengecekan algoritma pada metode *mesh grid* untuk bangunan yang terpisah lebih sulit oleh karena itu pengujian pada *urban environment* dibuat lebih sederhana karena hanya untuk menguji kemampuan algoritma. Sehingga peneliti menyarankan untuk mengaplikasikan algoritma ini pada lingkungan perkotaan yang lebih kompleks.

Ide yang didapat dari *paper* ini yaitu menggunakan algoritma Theta* untuk menemukan jalur terpendek pada lingkungan perkotaan 3D.

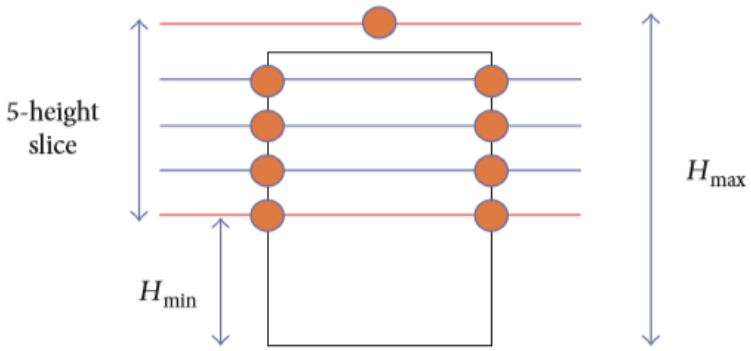
2.1.2. *Efficient Energy Flight Path planning Algoritm Using 3-D Visibility roadmap for Small Unmanned Aerial Vehicle* [11]

Paper ini menyajikan algoritma *path planning* dalam lingkungan 3D dengan *convex obstacle* statis untuk *small unmanned aerial vehicle* (SUAV) *fixed-wing*. *Path planning* yang digunakan pada *paper* ini menggunakan metode *visibility roadmap* untuk menentukan semua jalur yang dapat dilalui oleh SUAV dari titik awal ke titik tujuan yang telah ditetapkan dan memperhitungkan konsumsi energi yang dibutuhkan dalam pemilihan jalur.

Paper ini membuat 2 tahap prosedur yaitu pertama fase *preprocessing*, pada fase ini membuat *visibility graph* penuh yang disebut *visibility roadmap* yang menyediakan setiap koneksi *node* yang mungkin dilalui. Kedua, fase pada pencarian dimulai ketika modul pencarian jalur menerima koordinat awal dan koordinat tujuan. Setelah menghubungkan koordinat ini ke *roadmap*, dan memperoleh jalur penerbangan yang dioptimalkan menggunakan algoritma A* tepat setelah misi dimulai.

Untuk membangun *roadmap*, langkah pertama melibatkan pengambilan sampel 3D untuk *node* yang memungkinkan. Peta ini dibagi menjadi k-layer oleh

bidang horizontal k dengan jarak d_{cut} yang sama dari H_{min} ke H_{max}. Untuk menemukan *node* yang terletak di dalam batas *obstacle* dengan menghitung titik persimpangan bidang potong dan tepi halangan.

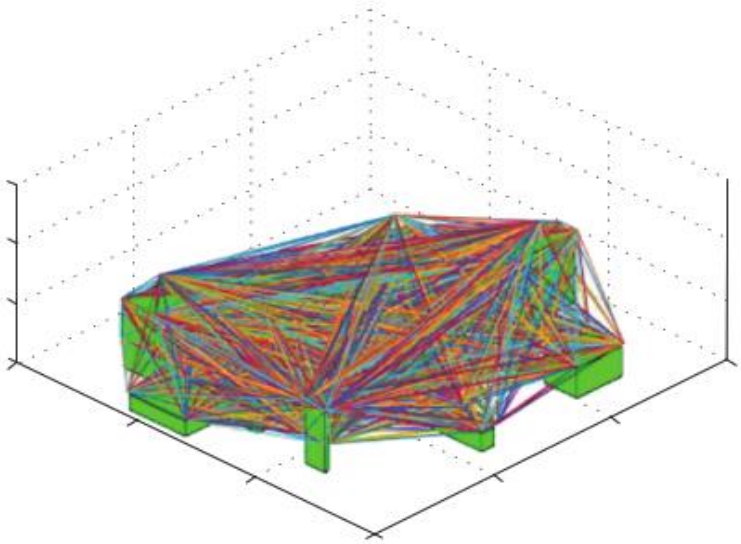


Gambar 2.9 *Roadmap* pada *Single Obstacle*

Penjelasan tentang pembuatan *visibility roadmap* membuat *single obstacle* dapat dilihat pada Gambar 2.9. *Paper* ini juga membuat lingkungan 3D dengan beberapa jumlah halangan dan waktu proses pembuatannya dapat dilihat pada Tabel 2.5. Jika dilihat pada table dapat disimpulkan bahwa semakin banyak halangan yang dibuat maka waktu prosesingnya juga semakin lama.

Tabel 2.5 Jumlah halangan dan *processing time*

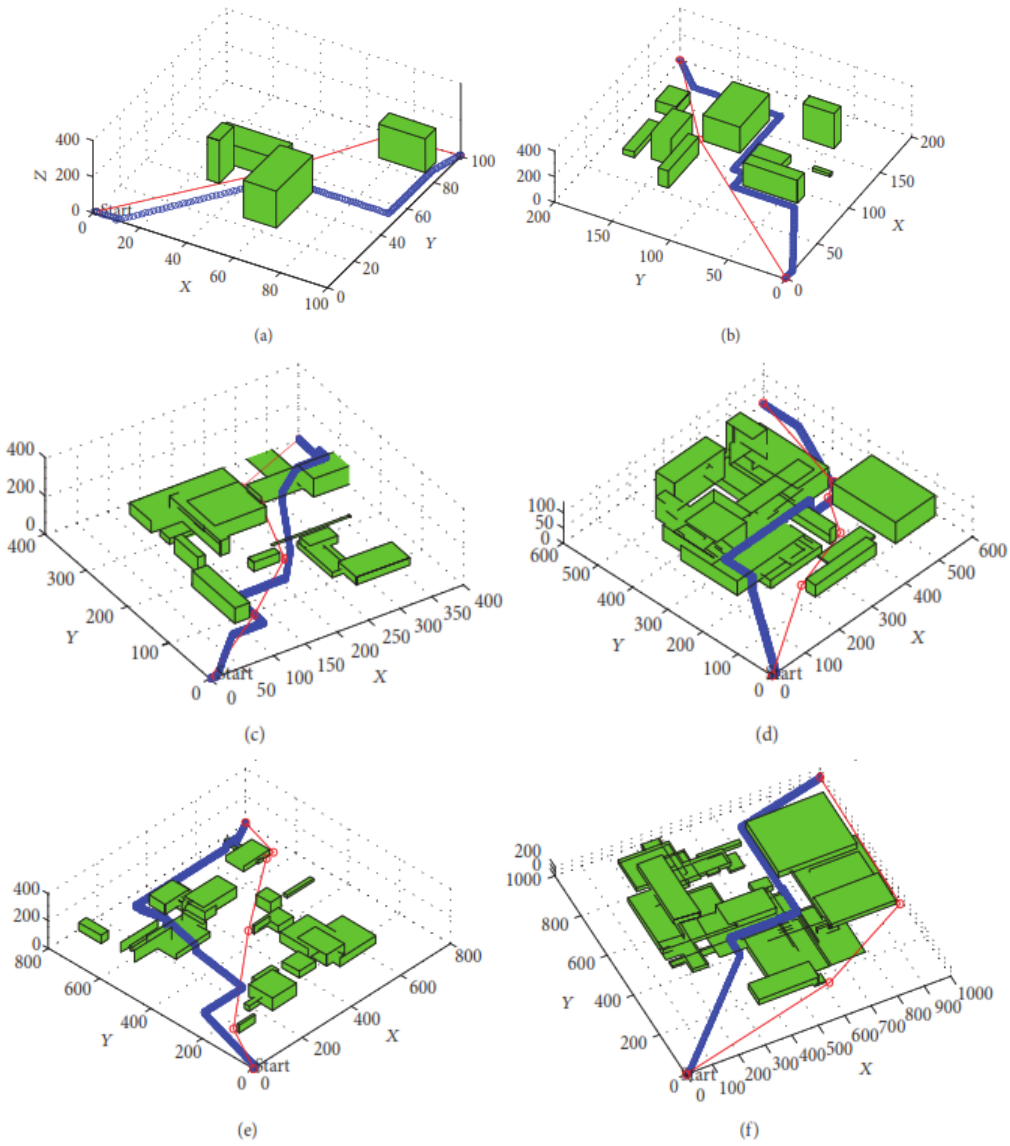
Jumlah Halangan	***Processing Time* (s)**
5	1.4
10	11.4
15	26.7
20	53.8
25	101
30	133



Gambar 2.10 Visualisasi pada 10 halangan

Pada Gambar 2.10 dapat dilihat keseluruhan *edge* yang terhubung. *Edge* ini merupakan *traversable path* yang dibuat untuk mempermudah algoritma pencari jalur untuk menemukan titik tujuan. Tujuan dari *paper* ini adalah untuk merencanakan jalur yang memiliki efisiensi energi yang lebih kecil. Oleh karena itu dengan menambahkan algoritma *bounded space* pada *visibility roadmap* maka algoritma pencari akan menyeleksi jalur berdasarkan kecilnya konsumsi energi.

Visualisasi dari dua metode yang digunakan (*visibility roadmap* dan *grid*) dapat dilihat pada Gambar 2.11. Garis merah menunjukkan metode *visibility roadmap* dan garis biru menunjukkan metode *grid.*



Gambar 2.11 Simulasi dari dua metode untuk semua variasi halangan

Tabel 2.6 Perbandingan Metode *Roadmap* dan *Grid*

Number of tests	Proposed visibility roadmap			Grid method		
	Energy	Distance	Heading changes	Energy	Distance	Heading changes
5	38,246	879.8	2.4	42,050	905.2	6.8
10	37,945	872.6	2.8	41,895	902.9	7.8
20	37,905	866.2	3.1	47,678	938.3	8
30	37,746	868	3.7	41,690	898.3	7.5
40	38,724	879.2	2.9	42,530	911.9	8.5
50	38,507	877.8	3.1	43,684	928.1	9.1

*Pape*r ini membandingkan hasil dari penelitiannya dengan tujuan penelitian yang sama, tetapi pembuatan *node* dengan menggunakan metode *grid*. Hasil perbandingan kedua metode ini dapat dilihat pada Tabel 2.6, berdasarkan tabel tersebut dapat diketahui bahwa konsumsi energi yang dihasilkan metode *visibility roadmap* memberikan nilai yang lebih kecil, dan jarak yang lebih pendek dari metode *grid*.

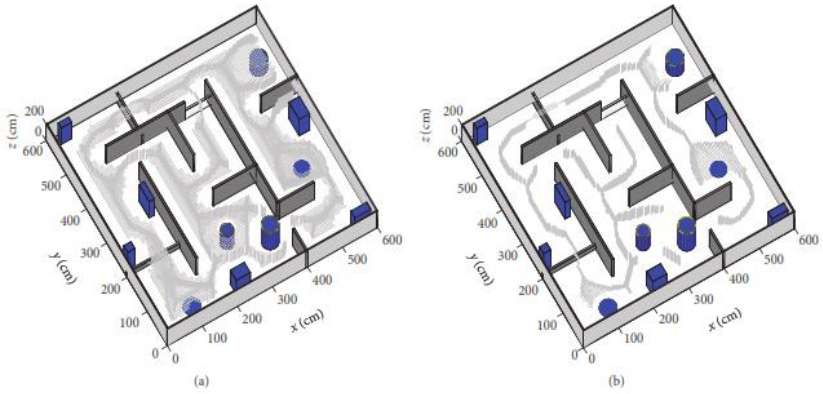
Ide yang dapat diambil dari penelitian ini adalah menggunakan metode *visibility graph* untuk pembuatan peta lingkungan 3D yang akan merepresentasikan lingkungan perkotaan. Alasan dari penggunaan algoritma ini adalah karena algoritma ini dapat mengurangi jumlah node, memperpendek jalur, serta ketersedian edge yang akan memudahkan algoritma pencarian untuk menentukan jalur terpendek yang akan di lalui oleh *Fixed-wing* UAV.

2.1.3. *Filtered Medial Surface Based Approach for 3D Collision-Free Path Planning Problem* [12]

Penelitian yang dilakukan oleh Benzaid dkk memperhatikan kemulusan jalur dan jalur bebas tabrakan untuk UAV saat melintasi jalur yang telah dipilih untuk mencapai titik tujuan. Peneliti memperkenalkan algoritma skeletonisasi 3D untuk membuat lingkungan yang akan dilalui oleh UAV berdasarkan algoritma representasi bentuk 2D atau disebut sebagai FMS (*Filtered Medial Surface*) yang memperhitungkan dimensi dari UAV.

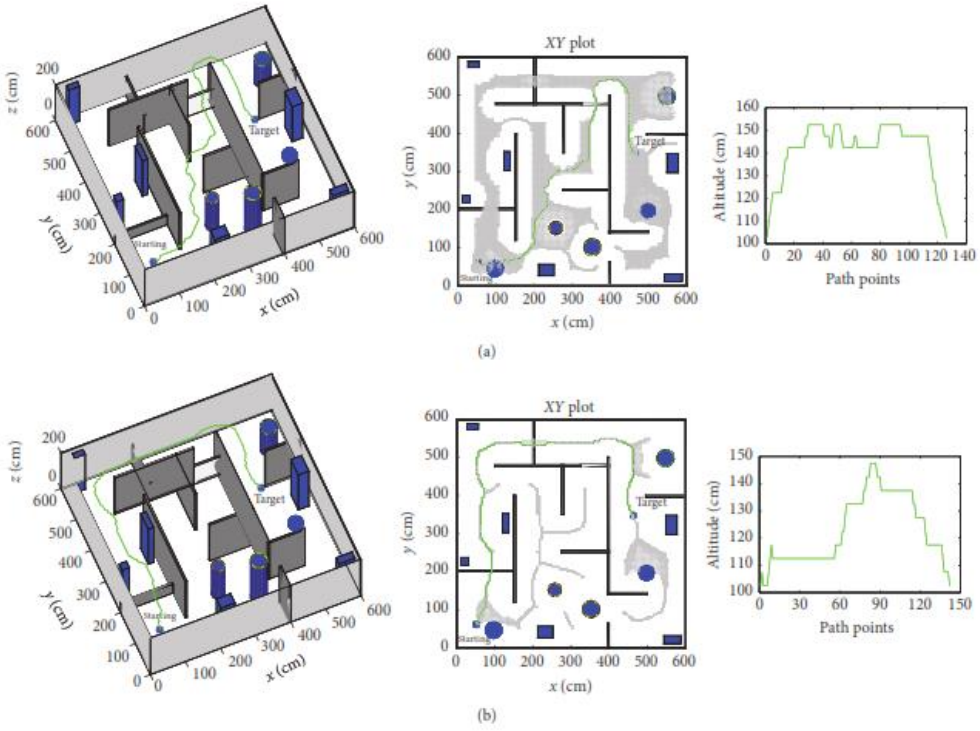
Algoritma A* diterapkan sebagai algoritma penemuan jalur terpendek pada FMS. Untuk menghilangkan osilasi dan belokan yang kencang di jalur yang dihasilkan oleh A*, peneliti mengusulkan pendekatan perataan berbasis koridor keselamatan baru menggunakan Bézier *curve*. Kemudian Peneliti melakukan studi komparatif dari pendekatan yang diusulkan (Bézier *curve*) dengan algoritma A*

secara langsung pada FMS dengan serangkaian kriteria kuantitatif. Pada Gambar 2.12 dibuat lingkungan 3D untuk 2 macam dimensi UAV.



Gambar 2.12 FMS berdasarkan ukuran UAV (a) $d = 20cm$ (b) $d = 40\ cm$

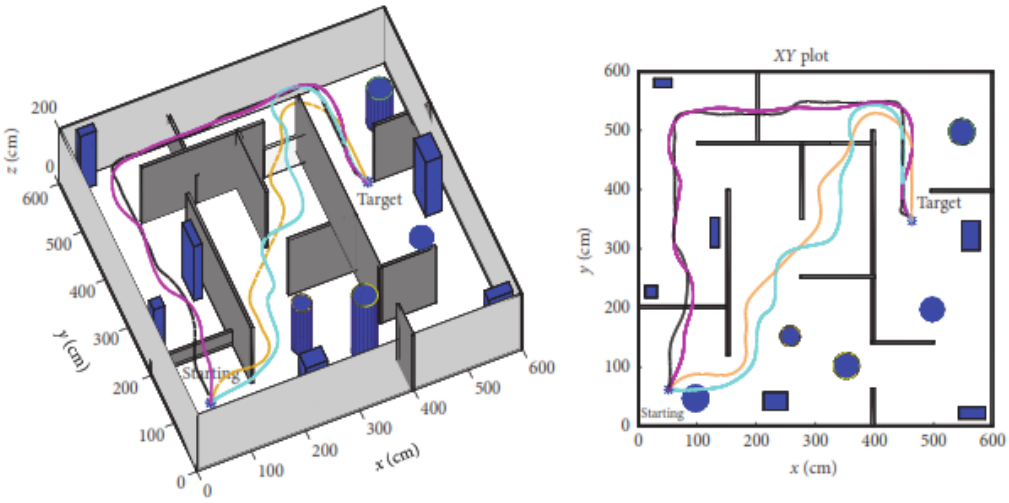
Setelah lingkungan 3D dibuat di FMS kemudian Algoritma A* mencari jalur terpendek yang akan dilalui UAV pada FMS. Penemuan jalur oleh A* dapat dilihat pada Gambar 2.13.



Gambar 2.13 Jalur terpendek pada FMS dengan dimensi UAV yang berbeda dengan altitude plot. (a) 20 cm. (b) 40 cm

Jalur yang dihasilkan harus dilalui oleh UAV secara elegan dan tenang tanpa adanya osilasi, namun jalur yang dihitung pada FMS belum tentu mulus. Oleh karena itu untuk mengatasi masalah ini, peneliti mengusulkan langkah pemrosesan tambahan berdasarkan penggunaan 3D Bézier *curve*, sehingga dapat dilihat pada Gambar 2.14 membandingkan jalur sebelum dan setelah menambahkan proses smoothing. Hasil dari kriteria kuantitatif yang diberikan pada Tabel 2.7.

Pada penelitian ini, peneliti menambahkan sebuah fungsi objektif untuk pembuatan jalur baru Bézier *curve.* Fungsi objektif ini berfungsi sebagai batasan agar jalur baru yang dibuat di sekitar jalur asli merupakan jalur yang berada dalam koridor aman.



Gambar 2.14 Perbandingan jalur sebelum dan setelah proses smoothing dengan ukuran UAV yang berbeda (ungu dan biru, Bézier *curve*) (oranye dan hitam, A *)

Dari hasil perbandingan pada Tabel 2.7 dapat disimpulkan bahwa metode Bézier *curve* mampu membuat jalur lebih halus dan bebas tabrakan. Oleh karena itu, tesis ini mengajukan metode ini sebagai *pra-smoothing* jalur setelah mendapatkan jalur terpendek menggunakan algoritma Theta*.

Tabel 2.7 Perbandingan kuantitif antara jalur A* dan Bézier *curve*

	A-star on the OG		Proposed approach	
	case 1	case 2	case 1	case 2
Input data number	236644	61841	23324	7913
Path length (m)	9.03	11.08	9.95	11.41
Number of path points	155	209	126	142
Smallest dist. to obstacles (cm)	25.63	42.86	31.13	46.28
Mean of dist. to obstacles (cm)	40.18	50.51	48.42	56.88

2.2 Teori Dasar

Pada bab ini terdapat teori dasar yang menunjang dalam merumuskan dan menyelesaikan masalah yang dihadapi dalam mengerjakan tesis. Bagian awal terdapat teori tentang quadcopter secara umum dan konsep gerak dari quadcopter. Bagian selanjutnya membahas tentang metode kontrol gerak quadcopter.

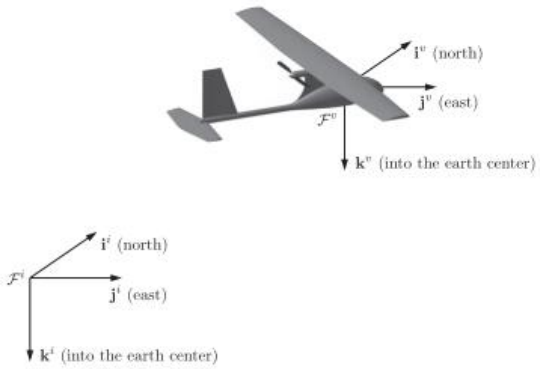
2.2.1 *Fixed-wing* UAV [13]

Pada pemodelan *fixed-wing*, akan dilakukan analisa dari *frame* koodinat UAV, *wind triangle*, parameter yang diperlukan pada pemodelan *fixed-wing*, kinematika dan dinamika *fixed-wing*.

Frame Koordinat UAV

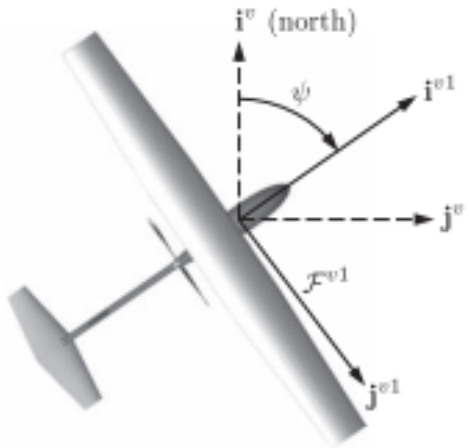
Untuk menentukan arah vektor dari UAV maka digunakan koordinat Cartesian. Pada bagian ini membahas berbagai koordinat pada sistem yang digunakan pada UAV dimulai dari posisi dan orientasi UAV dan sesnsor, dan transformasi antara koordinat sistem. Pada setiap koordinat memiliki inisialisasi yang berbeda. Koordinat frame yang digunakan:

- *Inertial frame* (f^i) : *Frame* ini adalah *frame* dari bumi dan biasa disebut *North-East-Down* (NED) *frame*.
- *Vehicle frame* (f^v) : Pusat dari *vehicle frame* adalah pusat massa UAV. Sumbu pada vehicle frame sesuai dengan sumbu inertial *frame*. *Vehicle frame* dapat dilihat pada Gambar 2.15.



Gambar 2.15 *Vehicle frame*

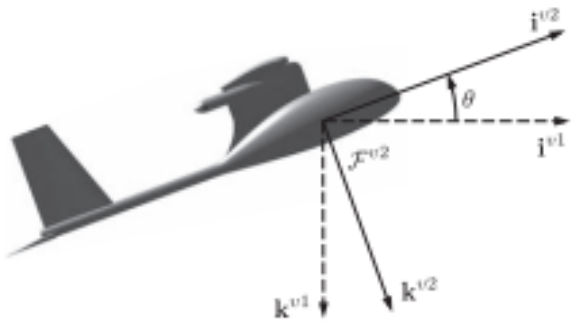
- *Vehicle*-1 *frame* (f^{v1}) : Pada *frame vehicle*-1 adalah rotasi UAV pada sudut (ψ) dengan menggunakan kaidah tangan kanan. *Vehicle-1 frame* dapat dilihat pada Gambar 2.16.



Gambar 2.16 *Vehicle-1 frame*

dimana, i^{v1} adalah arah hidung UAV, j^{v1} adalah arah sumbu sayap kanan, sedangkan k^{v1} bersesuaian dengan k^{v} dana rah menuju bumi.

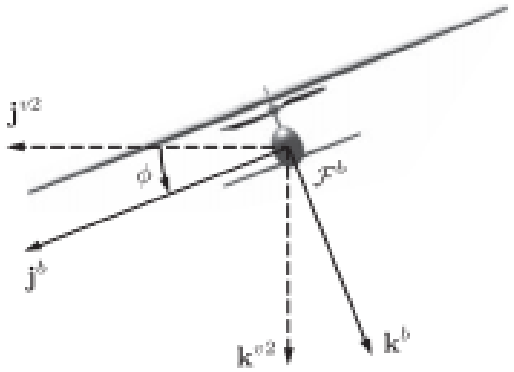
- *Vehicle*-2 *frame* (f^{v2}) : Pada frame *vehicle*-2 adalah rotasi UAV pada sudut (θ) dengan menggunakan kaidah tangan kanan. *Vehicle-2 frame* dapat dilihat pada Gambar 2.17.



Gambar 2.17 *Vehicle*-2 *frame*

dimana i^{v2} adalah arah hidung UAV, j^{v2} adalah arah sumbu sayap kanan, sedangkan k^{v2} merupakan arah tegak lurus UAV menuju bumi.

- *Body frame* (f^b) : Rotasi *body frame* terhadap sudut *roll* (ϕ) dengan menggunakan kaidah tangan kanan. *Body frame* dapat dilihat pada Gambar 2.18.



Gambar 2.18 *Body frame*

dimana i^b, j^b, dan k^b adalah unit vektor dari bada UAV dimana badan UAV sebagai arah sumbu x, y dan z.

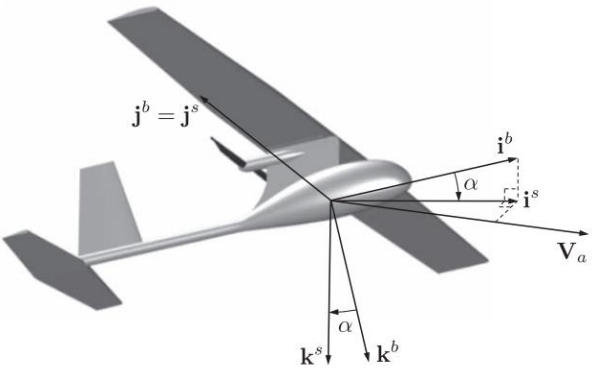
Apabila semua bentuk transformasi pada *vehicle frame* dan *body frame* digabungkan maka akan menjadi persamaan berikut:

$$\Re_v^b(\phi,\theta,\psi) = \Re_{v2}^b(\phi)\Re_{v1}^{v2}(\theta)\Re_v^{v1}(\psi) \tag{2.1}$$

$$\Re_v^b(\phi,\theta,\psi) = \begin{pmatrix} c\theta c\psi & c\theta s\psi & -s\theta \\ s\phi s\theta c\psi - c\phi s\psi & s\phi s\theta s\psi + c\phi c\psi & s\phi c\theta \\ c\phi s\theta c\psi + s\phi s\psi & c\phi s\theta s\psi - s\phi c\psi & c\phi c\theta \end{pmatrix} \tag{2.2}$$

dimana $s = sin, c = cos$.

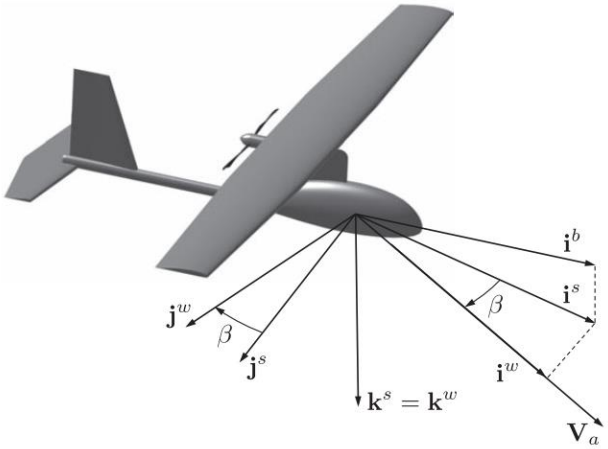
Stability frame (f^s) yang dapat dilihat pada Gambar 2.19 merupakan sudut yang dihasilkan antara rotasi badan pesawat pada sumbu y^b terhadap gaya aerodinamis terjadi saat gerakan dari UAV dipengaruhi oleh udara yang berada pada sekitar pesawat, dimana kecepatan udara dinotasikan dengan V_a. Untuk menghasilkan gaya angkat (*lift*), sayap pada pesawat harus terbang pada sudut positif dengan tidak mengabaikan arah dari kecepatan udara. Sudut ini disebut dengan *angle of attack* yang dinotasikan dengan α dimana pesawat akan berotasi pada sumbu *y* dari *body* pesawat.



Gambar 2.19 *Stability frame*

Sumbu J^b sama dengan sumbu sayap pesawat. Sedangkan sumbu i^s sesuai dengan sumbu dari arah kecepatan udara sedangkan i^b merupakan sumbu dari hidung pesawat, sehingga sudut α merupakan sudut yang dihasilkan dari i^s dengan i^b.

Wind frame (f^w) merupakan sudut antara arah angina dengan sumbu $x^b - z^b$ pada pesawat yang biasanya disebut dengan sudut side slip (β). *Wind frame* dapat dilihat pada Gambar 2.20.



Gambar 2.20 *Wind frame*

Sehingga total transformasi dari *body frame* terhadap *wind frame* adalah:

$$\Re_b^w(\alpha,\beta) = \Re_s^w(\beta)\Re_b^s(\alpha) \tag{2.3}$$

$$\Re_b^w(\alpha,\beta) = \begin{pmatrix} cos\,\beta\,cos\,\alpha & sin\,\beta & cos\,\beta\,sin\,\alpha \\ -sin\,\beta\,cos\,\alpha & cos\,\beta & -sin\,\beta\,sin\,\alpha \\ -sin\,\alpha & 0 & cos\,\alpha \end{pmatrix} \tag{2.4}$$

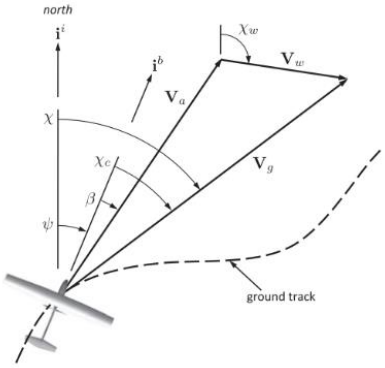
Wind Triangle

Wind triangle menggambarkan hubungan navigasi pada pesawat antara *groundspeed vector*, arah kecepatan udara, dan arah angin. Sudut antara inertia North (x^i) dan arah kecepatan inertia pada horizontal pesawat disebut dengan *course angle* (χ). *Crab angle* (χ_c) merupakan sudut antara *course angle* dan *heading* (arah hidung pesawat menurut kompas).

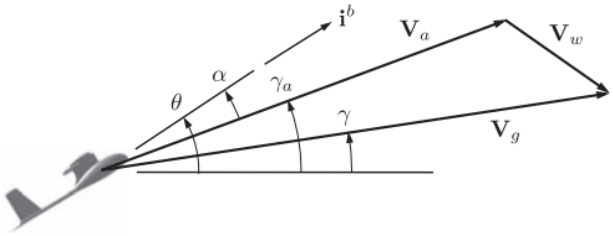
Flight jalur *angle* (γ) merupakan sudut antara sumbu horizontal pesawat terhadap kecepatan pada bumi (V_g). Sehingga terdapat dua sudut utama yang ditransformasikan dari *body frame* menuju *flight* jalur *frame* (χ, γ). Ada beberapa keadaan:

- *Crab angle* $(\chi_c) = 0$
- *Sideslip angle* $(\beta) = 0$
- $V_a = V_g$

Gambar 2.21 ini merupakan hubungan dari *wind triangle* secara horizontal sedangkan Gambar 2.22 menampilkan hubungan dari *wind triangle* secara vertikal.



Gambar 2.21 *Wind triangle* secara horizontal



Gambar 2.22 *Wind triangle* secara vertikal

Kinematika *Fixed-wing*

- Koefisien Aerodinamika gerak Longitudinal

Gerak longitudinal adalah gerak $x^b - z^b$ terhadap badan pesawat yang dapat disebut dengan gerak *pitch*, dimana gerak tersebut akan dipengaruhi oleh gaya angkat (*lift force* (f_L), *drag force* (f_D) dan *pitch moment* (m)).

- Koefisien Aerodinamika gerak Lateral

Gerak lateral adalah gerak pada *yaw* dan *roll* pesawat. Gerak lateral dipengaruhi oleh *side force* (f_Y), momen *yaw* (r), dan momen *roll* (p).

Dinamika *Fixed-wing*

Total gaya yang terjadi pada pesawat adalah:

$$\dot{p}_n = (\cos\theta\cos\psi)u + (\sin\phi\sin\theta\cos\psi - \cos\phi\sin\psi)v + (\cos\phi\sin\theta\cos\psi + \sin\phi\sin\psi)w$$

$$\dot{p}_e = (\cos\theta\sin\psi)u + (\sin\phi\sin\theta\sin\psi - \cos\phi\sin\psi)v + (\cos\phi\sin\theta\sin\psi + \sin\phi\cos\psi)w$$

$$\dot{h} = u\sin\theta - v\sin\phi\cos\theta - w\cos\phi\cos\theta$$

$$\dot{u} = rv - qw - g\sin\theta + \frac{\rho V_a^2 S}{2m}\left[C_X(\alpha)\frac{cq}{2V_a} + C_{X_{\delta e}}(\alpha)\delta_e\right] + \frac{\rho s_{prop} c_{prop}}{2m}\left[(k_{moto}\delta_t)^2 - V_a^2\right]$$

$$\dot{v} = pw - ru - g\cos\theta\sin\phi + \frac{\rho V_a^2 S}{2m}\left[\begin{array}{l} C_{Y_0} + C_{Y_\beta}\tan^{-1}\left(\frac{v}{\sqrt{u^2+w^2}}\right) + C_{Y_p}p \\ + C_{Y_r}r\frac{br}{2V_a} + C_{Y_{\delta a}}\delta_a + C_{Y_{\delta r}}\delta_r \end{array}\right]$$

$$\dot{w} = qu - pv - g\cos\theta\cos\phi + \frac{\rho V_a^2 S}{2m}\left[C_Z(\alpha) + C_{Z_q}(\alpha)\frac{cq}{2V_a} + C_{Z_{\delta e}}(\alpha)\delta_e\right] \qquad (2.5)$$

$$\dot{p} = \Gamma_1 pq - \Gamma_2 qr + \frac{\rho V_a^2 Sb}{2m}\left[\begin{array}{l} C_{p_0} + C_{p_\beta}\tan^{-1}\left(\frac{v}{\sqrt{u^2+w^2}}\right) + C_{p_p}\frac{bp}{2V_a} \\ + C_{p_r}\frac{br}{2V_a} + C_{p_{\delta a}}\delta_a + C_{Y_{\delta r}}\delta_a \end{array}\right]$$

$$\dot{q} = \Gamma_5 pr - \Gamma_6(p^2 - r^2) + \frac{\rho V_a^2 sc}{2J_y}\left[C_{m_0} + C_{m_\alpha} + C_{m_q}\frac{cq}{2V_a} + C_{m_{\delta e}}\delta_e\right]$$

$$\dot{r} = \Gamma_7 pq - \Gamma_1 qr + \frac{\rho V_a^2 Sb}{2}\left[C_{r_0} + C_{r_\beta}\beta + C_{r_p}\frac{bp}{2V_a} + C_{r_r}\frac{br}{2V_a} + C_{r_{\delta a}}\delta_a + C_{r_{\delta r}}\delta_r\right]$$

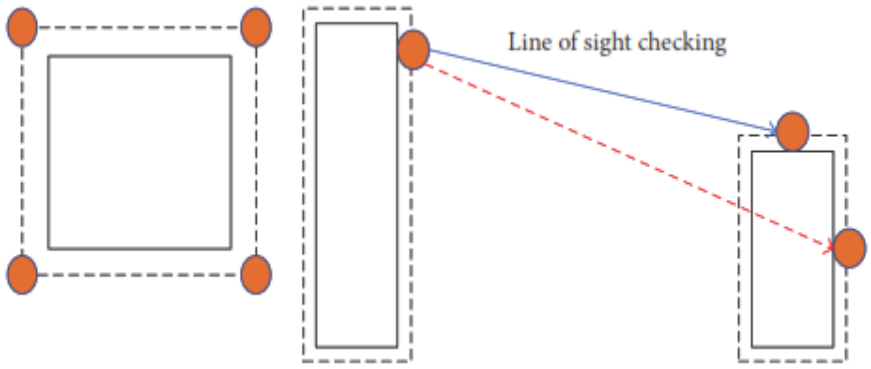
$$\dot{\phi} = p + q\sin\phi\tan\theta + r\cos\phi\tan\theta$$

$$\dot{\psi} = q\sin\phi\sec\theta + r\cos\phi\sec\theta$$

$$\dot{\theta} = q\cos\phi - r\sin\phi$$

2.2.2 *Visibility Graph* [11]

Visibility graph adalah sebuah graf lokasi yang saling terlihat, biasanya untuk satu set *node* dan halangan pada bidang *euclidean*. Masing-masing *node* pada graf mewakili sebuah lokasi *node*, dan masing-masing *edge* mewakili setiap hubungan yang terlihat diantara *nodes*, yaitu jika ruas garis yang menghubungkan dua lokasi *node* tidak melewati halangan apa pun, maka sebuah *edge* akan di gambarkan antara *node* pada graf. Pembuatan *edge* pada sekumpulan *node* di dalam halangan, kompleksitas waktu dari algoritma bergantung pada fungsi pemeriksaan *line of sight* (LOS). Mengingat karakteristik halangan adalah *convex* sehingga menghitung LOS dalam waktu linier dapat digunakan. Langkah-langkah memperluas LOS dapat dilustrasikan pada Gambar 2.23.

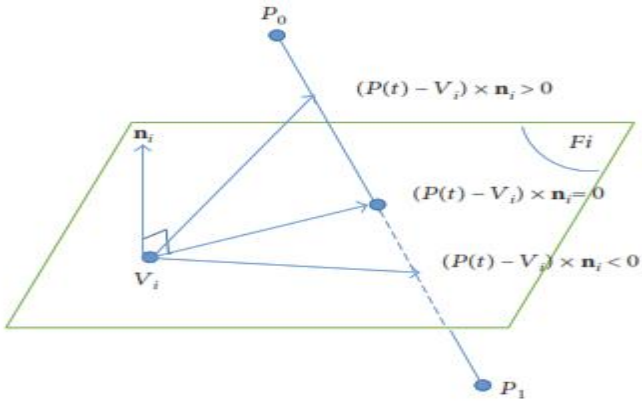


Gambar 2.23 Ilustrasi pemeriksaan *line of sight* (LOS)

Ada dua perbedaan kasus pada prosedur pemeriksaan fungsi LOS yaitu, hubungan *node* dalam halangan yang sama atau dengan halangan yang berbeda. Menurut karakteristik *convex*, *node* awal dan tujuan memiliki LOS jika mereka berada dalam satu *facet*, jika tidak mereka tidak memiliki LOS. Untuk *node* yang berasal dari halangan yang berbeda, akan di proses dengan mengeksploitasi algoritma *convex intersection* untuk mengecek visibilitas.

Algoritma pengecekan LOS menganggap sebuah halangan sebagai sebuah *convex polyhedron* Ω yang terdiri dari satu set *convex polygon faces* F_i, dimana i adalah *polygon faces*. *Faces* dipilih berdasarkan *doubly connected edge list* (DCEL) dari halangan. Struktur data DCEL dari halangan memudahkan kita untuk

menghitung vektor normal dari F_i yang disebut sebagai n_i yang memiliki arah keluar ke halangan. Maksudnya adalah setiap titik P_i yang berada di sisi n_i adalah bagian luar dari halangan. Definisikan V_i sebagai titik dibidang *faces* F_i, *vertex* F_i ini dipilih karena untuk penyederhanaan. Tentukan P_0 dan P_1sebagai dua node pada sebuah ruang. Sebuah segmen garis $S = P_0P_1$ direpresentasikan oleh sebuah persamaan parametrik $P(t)$ adalah sebuah LOS antara dua *node*. Jika segmen garis ini berpotongan dengan $convex\ polyhedron$, LOS terhalang oleh halangan yang diproses saat ini, seperti yang digambarkan pada gambar 2.24.



Gambar 2.24 Perpotongan segmen garis ke *convex face* pada 3D

Menurut geometri dasar, perpotongan terjadi ketika $P(t) - V_i \times n_i = 0$. Oleh karena itu nilai t_i dihitung sebagai $(V_i - P_0 \times n_i)/((P_1 - P_0) \times n_i)$. Karena *vector* normal n_i menunjuk ke arah luar bidang $faces$, sehingga dapat menentukan nilai t_i ketika garis segmen S masuk dan keluar. Misalnya, $(P_1 - P_0) \times n_i < 0$ ketika t_i masuk dan $(P_1 - P_0) \times n_i > 0$ ketika t_i keluar. Kemudian untuk menghitung $t_E = max$ (0, t_i yang masuk) dan $t_L = min$ (1, t_i yang keluar). Jika orde dari t_E dan t_L adalah $0 \leq t_E \leq t_L \leq 0$, kemudian garis segmen *S* berpotongan dengan *convex polyhedron* Ω. Saat pemeriksaan LOS antara setiap pasangan elemen pada sekumpulan *node*, jika *node* memiliki LOS maka koneksi (*node* asal dan tujuan) disimpan sebagai *edge*.

Pseudocode untuk algoritma *visibility graph* dalam pembuatan *roadmap* dapat dilihat pada Gambar 2.25.

```
Input: Elevation matrix M in R^3.
Output: Γ = [N, P]
1.  FOR (each cut layer in the map; i = 1, k)
2.      cutPlane = Slice(M, H_min, H_max, k); //the parametric equation of the cut plane.
3.      FOR (each obstacle; j = 1, n)
4.          IF Obstacle -> highestNode(Z) < k * Dcut //check if the obstacle highest node is lower than the cut plane
5.          THEN
6.              For (each edge of the obstacle)
7.              tentativeNode = FindIntersection (cutPlane, edge);
8.              newNode = enlarge(tentativeNode, Dsafe);
9.              N = Add(newNode,obstacleIndex, facetIndex);
10.         ELSE
11.             newNode = [Obstacle -> highestNode(X)
12.                        Obstacle -> highestNode(X)
13.                        k*Dcut]
14.             N = Add(newNode,obstacleIndex, facetIndex1, facetIndex2);
15.             BREAK;
16. FOR (Each pair of Node in set N; i = 1,NumberOfNode; j = i + 1,NumberOfNode)
17.     IF (LOScheck() is TRUE)
18.     THEN
19.         P = Add(N(i), N(j));
20.         P = Add(N(j), N(i));
21.  R = (N,P);
22.  SaveToFile(R);
```

Gambar 2.25 *Pseudocode* pembuatan *roadmap* [11]

Penjelasan simbol-simbol yang digunakan di *pseudocode* pada Gambar 2.25 yaitu:

M : Data peta digital

Γ : *Roadmap*

N : Kumpulan *node* yang dapat dilalui dalam ruang

P : Kumpulan *edge* yang dapat dilalui diantara dua *node*

H_{min} : Ketinggian terbang minimum

H_{max} : Ketinggian terbang maksimum

D_{cut} : Jarak antara *cut layers*

D_{safe} : Jarak aman saat mendekati halangan

n : Jumlah halangan

k : Jumlah *cut layers*

LOS : Line of sight antara dua *node*

2.2.3 Algoritma Theta* [9][14]

Theta* adalah algoritma pencarian *any-angle* yang berarti pencarian di segala sudut. Secara empiris, Theta* dapat mencari jalan yang lebih pendek jika dibandingkan dengan algoritma A*. Hal ini dikarenakan Theta* menggabungkan kemampuan A* pada grafik visibilitas (dimana perubahan arah terjadi hanya pada sudut-sudut dari sel yang terhalang) dan kemampuan A* pada *grids* (dimana setiap sudut berkembang sejumlah dengan sel yang ada). Walaupun demikian, algoritma Theta* memiliki kelemahan dalam proses pencarian yang lama. Perbedaan utama antara Theta* dan A* di *grid* adalah bahwa pada Theta*, *parent* dari sebuah *node* dapat berupa sudut mana pun, sedangkan *parent* dari sebuah *node* pada A* harus berdekatan dengan *node* tersebut.

2.2.4 Bézier *curve* [12]

Pada tahun 1962, seorang *engineer* Prancis yang bernama Pierre Bézier menemukan sebuah metode untuk mendefinisikan kurva dengan kontrol *polygon*. Berdasarkan prinsip aproksimasi, metode ini membangun sebuah set fungsi basis polinomial yang secara efektif dapat menjelaskan kurva atau sketsa matematika yang digambar oleh perancang. Oleh karena itu, kurva polinomial parametrik yang ditentukan oleh titik kontrol dan banyak digunakan dalam grafik komputer untuk membuat kurva yang halus. Seperti disebutkan sebelumnya, definisi kurva Bézier sangat bergantung pada jumlah titik kontrol kurva. $n + 1$ dapat mendefinisikan kurva derajat n polinomial. Dengan mempertimbangkan $P_0, P_1, \dots \dots P_{n,}$ $(n \geq 1), n + 1\, point$ sebagai kontrol point. n derajat dari kurva Bézier dihubungkan dengan titik-titik ini adalah kurva C yang didefinisikan dengan $M(t)$ dimana:

$$M(t) = \sum_{i=0}^{n} P_i \, B_{i,n}(t), \;\; t \in [0, 1] \tag{2.6}$$

dimana $P_i \, (i = 1 : n)$ adalah kontrol point dari Bézier *curve* atau disebut juga kontrol *polygon*. $B_{i,n}$ adalah fungsi dasar Bernstein dan persamaan polinomialnya adalah

$$B_{i,n}(t) = C(n, i)t^i(1 - t)^{n-i}, i \; 0,1,2, \dots, n \tag{2.7}$$

substitusikan $C(n,i) = \frac{n!}{i!(n-i)!}$ ke (2.7), sehingga fungsi dasar Bernstein dapat ditulis menjadi menjadi

$$B_{i,n}(t) = \frac{n!}{i!(n-i)!} t^i (1-t)^{n-i} \tag{2.8}$$

Pada lingkungan 3D, P_i merepresentasikan titik (x_i, y_i, z_i) sehingga titik $M(t)$ memiliki koordinat:

$$\begin{cases} x(t) = \sum_{i=0}^{n} B_i(t)\, x_i, \\ y(t) = \sum_{i=0}^{n} B_i(t)\, y_i, \\ z(t) = \sum_{i=0}^{n} B_i(t)\, z_i, \end{cases} \tag{2.9}$$

Bézier *curve* sangat cocok digunakan untuk menyelesaikan permasalahan penghalusan jalur, karena mudah digunakan dan dapat menjelaskan banyak bentuk kurva. Bézier *curve* juga memenuhi sifat *endpoint interpolation property* dan juga *endpoint tangen property*. Untuk membentuk jalur mulus yang komplek, *cubic* Bézier *curve* dapat dihubungkan dengan sifat *endpoint tangen property*, maksudnya adalah jika *cubic* Bézier *curve* C memiliki empat titik kontrol sehingga *cubic* Bézier *curve* C bersinggungan dari $[P_0, P_1]$ $[P_2, P_3]$ pada $[P_0, P_3]$ sehingga;

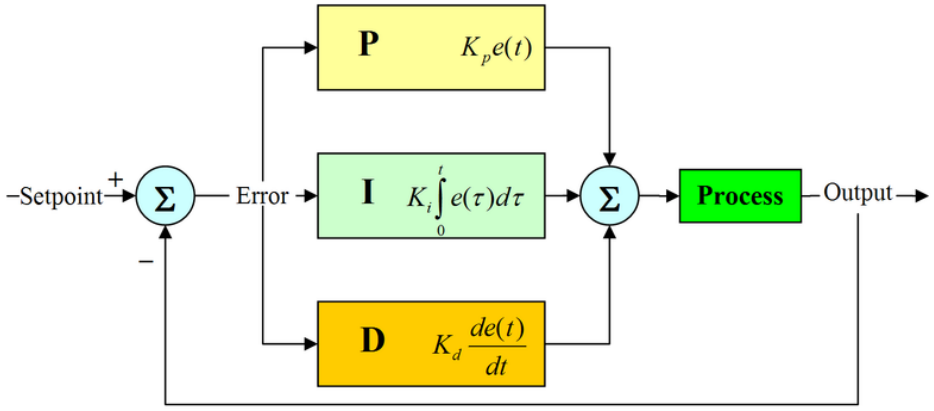
$$\begin{aligned} M(0) &= 3(P_1, P_0) \\ M(1) &= 3(P_3, P_2) \end{aligned} \tag{2.10}$$

2.2.5 *Proportional Integral Derivative* (PID)[15]

Kontroler PID merupakan kontroler mekanisme umpan balik yang biasanya dipakai pada sistem instrumentasi. Kontroler PID secara kontinyu menghitung nilai kesalahan sebagai beda antara *setpoint* yang diinginkan dan variabel proses yang terukur, untuk blok diagram kontroler PID dapat dilihat pada Gambar 2.26. Kontroler mencoba untuk meminimalkan nilai kesalahan setiap waktu dengan penyetelan variabel kontrol ke nilai baru yang ditentukan oleh persamaan berikut:

$$u(t) = K_p e(t) + K_i \int_0^t e(t)dt + K_d \frac{de(t)}{dt} \tag{2.11}$$

dimana K_p, K_i, K_d, semuanya positif, masing-masing adalah koefisien untuk suku proporsional, integral dan derivatif.



Gambar 2.26 Blok diagram kontroler PID

Pada model PID ini,

- **P** bertanggung jawab untuk nilai kesalahan saat ini. Contoh, jika nilai kesalahan besar dan positif, maka keluaran kontrol juga besar dan positif.
- **I** bertanggung jawab untuk nilai kesalahan sebelumnya. Contoh, jika keluaran saat ini kurang besar, maka kesalahan akan terakumulasi terus menerus, dan kontroler akan merespon dengan keluaran lebih tinggi.
- **D** bertanggung jawab untuk kemungkinan nilai kesalahan mendatang, berdasarkan pada rute perubahan tiap waktu.

Beberapa aplikasi mungkin hanya menggunakan satu atau dua suku untuk memberikan kontrol sistem yang sesuai. Hal ini dapat dicapai dengan mengatur parameter yang lain menjadi nol. Kontroler PID dapat menjadi PI, PD, P atau I tergantung aksi apa yang digunakan.

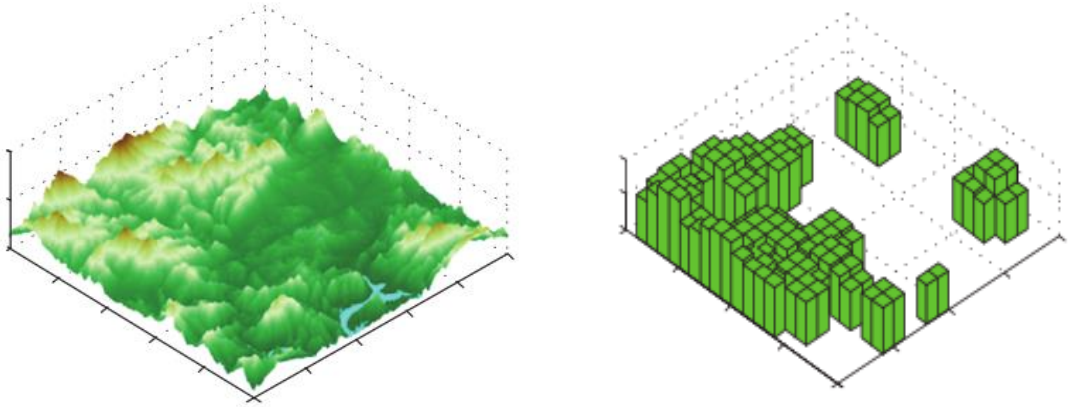
BAB 3
PENGHALUSAN JALUR BERDASARKAN KEMAMPUAN MANUVER FIXED-WING

Bab ini akan membahas mengenai langkah kerja penelitian yang akan dilakukan dalam perancangan algoritma yang dapat dilalui oleh *Fixed-wing UAV* pada lingkungan perkotaan yang akan direpresentasikan sebagai blok-blok persegi pada sebuah *roadmap* yang akan dibuat. Diberikan titik awal dan titik target pada *roadmap* yang sudah dilengkapi dengan *nodes* dan *edge*, lalu algoritma pencarian jalur digunakan untuk mendapatkan jalur terpendek pada *nodes* yang telah disediakan. Jalur yang didapatkan oleh algoritma pencarian perlu dihaluskan karena jalur yang dihasilkan dapat berupa pembelokan yang tajam. Metode yang digunakan untuk penghalusan jalur yaitu metode Bézier *curve* dengan mempertimbangkan kemampuan *maneuver* dari *fixed-wing*. Selanjutnya untuk memastikan jalur yang sudah dihaluskan sudah sesuai dengan kemampuan manuver *fixed-wing*, maka *fixed-wing* diterbangkan untuk melewati jalur tersebut. *Fixed-wing* yang digunakan pada penelitian ini adalah *fixed-wing* ultrastick-25.

3.1 Pemodelan Lingkungan 3D

3.1.1 Pemodelan *Digital Map*

Pembuatan lingkungan 3D yang merepresentasikan lingkungan perkotaan dimodelkan seperti balok-balok kubus menggunakan data matriks pada Matlab yang dibuat pada area dengan luas tertentu tergantung dari banyaknya bangunan yang akan dibuat, balok-balok kubus yang merepresentasikan lingkungan perkotaan ini disebut sebagai *map M* dan menjadi *input* untuk membuat algoritma *visibility graph* untuk pembuatan *roadmap* yang menyajikan *nodes* dan *edges.* Balok – balok kubus yang merepresentasikan lingkungan perkotaan ini dapat dilihat seperti Gambar 3.1.



(a) (b)

Gambar 3.1 Pemodelan lingkungan 3D. (a) Contoh *digital map*. *(*b) *Convex obstacle* 3D

3.1.2 Algoritma pembuatan *roadmap*

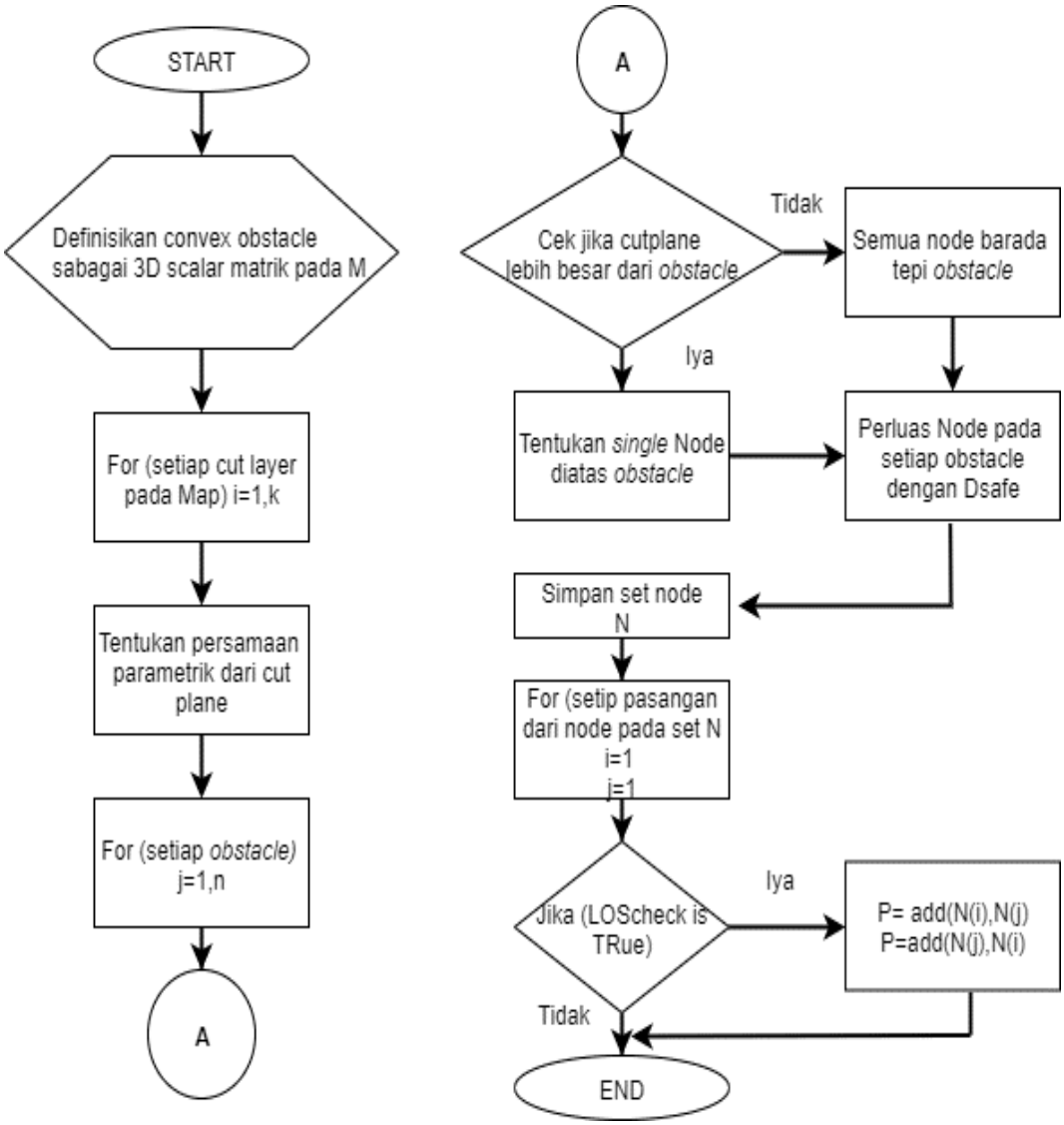
Subbab ini menyajikan tentang deskripsi *roadmap* dan algoritma untuk pembuatan *roadmap*. *Digital map M* yang dipresentasikan oleh nilai ketinggian dalam sebuah matriks skalar 3D. Roadmap Γ adalah kumpulan *node* (*node*; *N*) dan koneksi jalur (*edge*; *P*) antar *node*. Secara matematis, ini adalah grafik tepi ganda terarah dimana semua *nodes* yang terlihat saling berhubungan.

Untuk membangun *roadmap*, langkah pertama melibatkan pengambilan sampel 3D untuk *node* yang memungkinkan. *Map* ini dibagi menjadi *k-layer* oleh bidang horizontal *k* dengan jarak potong yang sama dari H_{min} ke H_{max}. Membuat *node* di dalam batasan halangan yaitu dengan menghitung titik-titik persimpangan bidang potong dan tepi halangan. Jika bidang potong lebih tinggi dari halangan, maka tentukan *single node* yang terletak di atas halangan yang memiliki ketinggian yang sama dengan bidang potong.

Setelah itu, perluas pembuatan *node* pada halangan dengan jarak yang aman. Koordinat *nodes* yang diperluas harus disimpan dalam set *N* (*node*). Setelah menemukan set *node N*, selanjutnya membuat set *edge P* dengan pemeriksaan visibilitas. Algoritma ini menyajikan semua *node* yang memiliki *line of sight* (LOS) memiliki *edge* masing-masing. Kompleksitas waktu dari algoritma sangat

tergantung pada fungsi pemeriksaan LOS, sehingga metode untuk menghitung LOS dapat diperoleh dalam waktu linier.

Secara umum perancangan algoritma *roadmap* dapat digambarkan dalam bentuk *flowchart* pada Gambar 3.2.



Gambar 3.2 *Flowchart* perancangan algoritma *roadmap*

3.2 Theta* dan Pencarian Jalur Terpendek

Dari 3D *visibility roadmap* yang telah dibuat, tahap selanjutnya adalah menemukan jalur terpendek yang menghubungkan titik awal ke titik akhir. Pencarian dimulai setelah memuat konfigurasi *map* dengan set *node*, set *edge* pada halangan 3D. Algoritma heuristik Theta* akan digunakan untuk menemukan jalur terpendek. Algoritma Theta* ini merupakan algoritma turunan yang dikembangkan dari Algoritma A* yang telah umum digunakan pada permasalahan *pathfinding*. Cara kerja algoritma ini hampir sama dengan algoritma A*, namun hal yang membedakannya adalah jika algoritma A* memperkirakan jalur melalui *node* yang bersebelahan dan berurutan. Sedangkan Theta* dapat menentukan jalur berdasarkan *node* yang masih dalam jangkauan LOS dan tidak harus bersebelahan. Theta* memperluas titik pencariannya menggunakan LOS yang nantinya akan digunakan untuk melihat *edge* dalam jangkauan yang sudah disediakan oleh *visibility roadmap*.

Dalam perancangan algoritma juga memelihara dua struktur data global yaitu: (1) *Open List*, antrian prioritas yang berisi *node-node* yang harus dipertimbangkan untuk ekspansi. (2) *Close List*, yang berisi *node-node* yang sudah diperluas dan dipastikan bahwa masing-masing *node* sudah diperluas satu kali. Theta* mengevaluasi *node* dengan menggabungkan $g(n)$, yaitu *cost* untuk mencapai *node*, dan $h(n)$, yaitu *cost* yang diperlukan dari *node* untuk mencapai tujuan, dalam notasi matematika dituliskan pada (3.1).

$$f(n) = g(n) + h(n) \tag{3.1}$$

dimana:

$f(n)$ = Biaya evaluasi

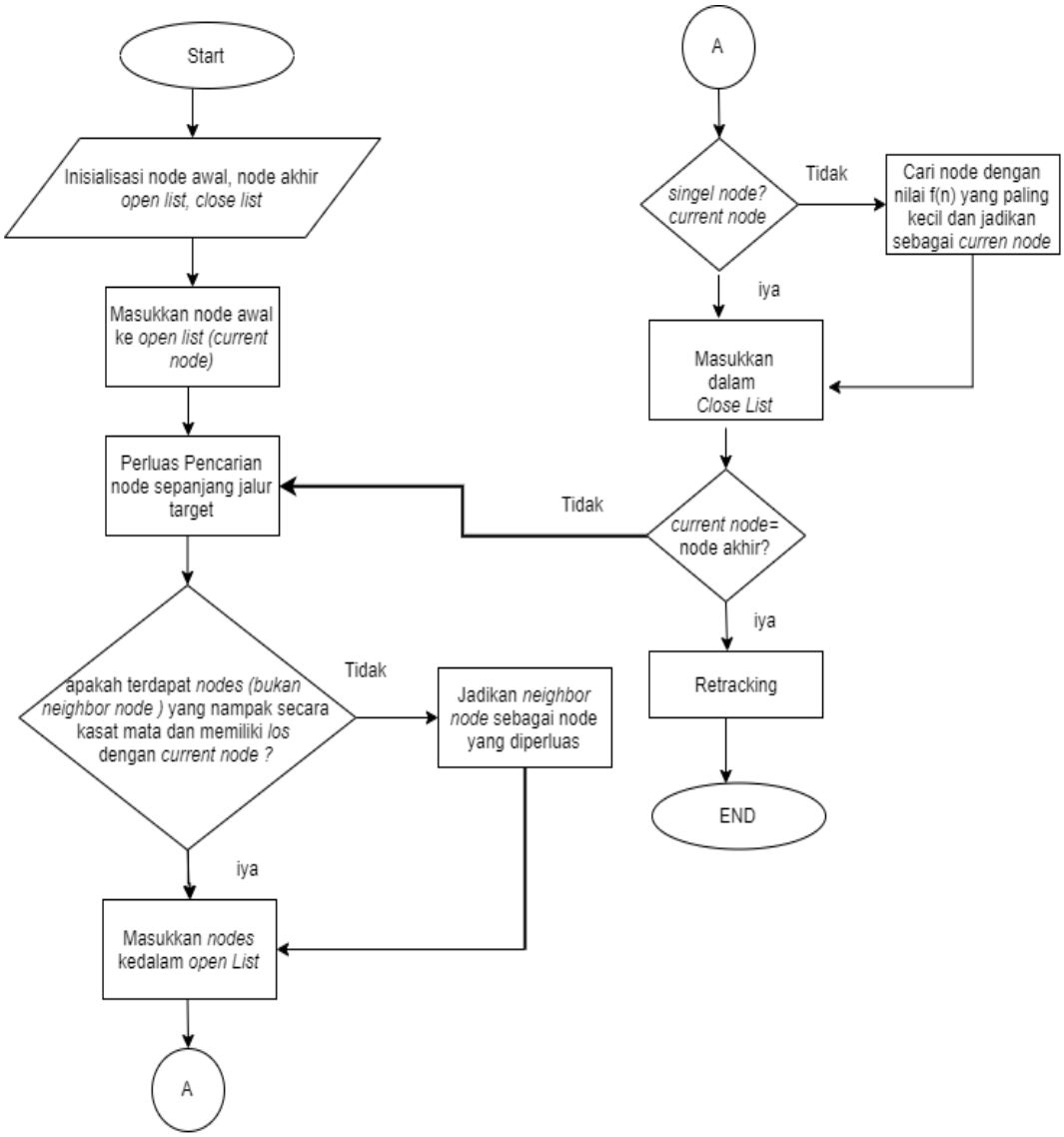
$g(n)$ = Biaya yang sudah dikeluarkan dari keadaan awal sampai keadaan n

$h(n)$ = Estimasi biaya untuk sampai pada suatu tujuan mulai dari n

Biaya dari fungsi pada dapat mengunakan metode jarak *Euclidean* untuk menentukan jarak antara dua *node*. Jarak *Euclidean* ini berkaitan dengan teorema *Phytagoras* dan biasanya dapat diterapkan pada 1, 2 dan 3 dimensi. Pada penelitian ini, posisi *node* berada dalam ruang 3D sehingga rumus jarak *Euclidean* menjadi seperti pada (3.2).

$$j = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2} \quad (3.2)$$

Secara garis besar rancangan pembuatan algoritma Theta* sebagai pencari jalur terpendek pada lingkungan perkotaan yang sudah disajikan dapat dirangkum dalam bentuk sebuah flowchart pada Gambar 3.3.



Gambar 3.3 *Flowchart* algoritma Theta*

Untuk *pseudocode* pembuatan algoritma Theta* pada *visibility roadmap* adalah sebagai berikut:

Input: $\Gamma = [N, P]$, start node, goal node.

Output: A shortest path between start and goal nodes.

1: Openlist O_L.
2: Closelist C_L.
3: **repeat**
4: Pick n_{best} from O_L such that $f(n_{best}) \leq f(n), \forall n \in O_L$.
5: Remove n_{best} from O_L and add to C_L.
6: If $n_{best} = q_{goal}$, EXIT.
7: Expand n_{best}: $\forall n_P$ that are not in C_L.
8: calculate n_{best} from n_P and add to C_L.
9: **until** O_L is empty

Gambar 3.4 *Pseudocode* pemilihan jalur terpendek menggunakan algoritma Theta* di *visibility graph*

Penjelasan simbol-simbol yang digunakan di *pseudocode* pada Gambar 3.4 yaitu:

O_L : *Openlist*

C_L : *Closelist*

$f(n)$: $g(n) + h(n)$, kriteria utama untuk evaluasi pencarian dan seleksi

$g(n)$: Biaya yang sudah dikeluarkan dari awal sampai mencapai *node* n

$h(n)$: Estimasi biaya dari node n sampai ke tujuan

3.3 *Fixed-wing* Ultrastick-25 (*Manuever)*

Pada penelitian ini *fixed-wing* ultrastick-25 digunakan sebagai *plant* yang akan melintasi jalur terpendek yang sudah didapatkan oleh algoritma Theta*. Algoritma Theta * tidak menganggap kinematika kendaraan sebagai bagian dari pembuatan jalur. Sehingga permasalahan ini menjadi masalah utama untuk *nonholonomic vehicles* seperti *fixed-wing*, yang membutuhkan proses perataan untuk merealokasi urutan *node* untuk mendapatkan jalur yang dapat diterbangkan. Solusi yang akan digunakan pada penelitian ini adalah menggunakan metode *Bézier curve* untuk penghalusan jalur.

Fixed-wing merupakan UAV yang memiliki kemampuan belok yang terbatas. Oleh karena itu pada perancangan Bézier *curve* akan mempertimbangkan

kemampuan manuver dari *fixed-wing* ultrastick-25, sehingga jalur yang dihasilkan oleh Bézier *curve* merupakan jalur yang dapat dilewati oleh *fixed-wing*. Kemampuan manuver dari *Fixed-wing* ultrastick-25 akan didapatkan dengan mensimulasikan persamaan dari model dinamika *fixed-wing* pada (3.3) - (3.14).

$$\dot{p}_n = (\cos\theta\cos\psi)u + (\sin\phi\sin\theta\cos\psi - \cos\phi\sin\psi)v + (\cos\phi\sin\theta\cos\psi + \sin\phi\sin\psi)w \tag{3.3}$$

$$\dot{p}_e = (\cos\theta\sin\psi)u + (\sin\phi\sin\theta\sin\psi - \cos\phi\sin\psi)v + (\cos\phi\sin\theta\sin\psi + \sin\phi\cos\psi)w \tag{3.4}$$

$$\dot{h} = u\,sin\,\theta - v\,sin\,\phi\,cos\,\theta - w\,cos\,\phi\,cos\,\theta \tag{3.5}$$

$$\dot{u} = rv - qw - g\sin\theta + \frac{\rho V_a^2 S}{2m}\left[C_X(\alpha)\frac{cq}{2V_a} + C_{X_{\delta e}}(\alpha)\delta_e\right] + \frac{\rho s_{prop} c_{prop}}{2m}\left[\left(k_{moto}\delta_t\right)^2 - V_a^2\right] \tag{3.6}$$

$$\dot{v} = pw - ru - g\cos\theta\sin\phi + \frac{\rho V_a^2 S}{2m}\left[\begin{array}{l} C_{Y_0} + C_{Y_\beta}\tan^{-1}\left(\frac{v}{\sqrt{u^2+w^2}}\right) + C_{Y_p}p \\ + C_{Y_r}r\frac{br}{2V_a} + C_{Y_{\delta a}}\delta_a + C_{Y_{\delta r}}\delta_r \end{array}\right] \tag{3.7}$$

$$\dot{w} = qu - pv - g\cos\theta\cos\phi + \frac{\rho V_a^2 S}{2m}\left[C_Z(\alpha) + C_{Z_q}(\alpha)\frac{cq}{2V_a} + C_{Z_{\delta e}}(\alpha)\delta_e\right] \tag{3.8}$$

$$\dot{p} = \Gamma_1 pq - \Gamma_2 qr + \frac{\rho V_a^2 Sb}{2m}\left[\begin{array}{l} C_{p_0} + C_{p_\beta}\tan^{-1}\left(\frac{v}{\sqrt{u^2+w^2}}\right) + C_{p_p}\frac{bp}{2V_a} \\ + C_{p_r}\frac{br}{2V_a} + C_{p_{\delta a}}\delta_a + C_{Y_{\delta r}}\delta_a \end{array}\right] \tag{3.9}$$

$$\dot{q} = \Gamma_5 pr - \Gamma_6(p^2 - r^2) + \frac{\rho V_a^2 sc}{2J_y}\left[C_{m_0} + C_{m_\alpha} + C_{m_q}\frac{cq}{2V_a} + C_{m_{\delta e}}\delta_e\right] \tag{3.10}$$

$$\dot{r} = \Gamma_7 pq - \Gamma_1 qr + \frac{\rho V_a^2 Sb}{2}\left[C_{r_0} + C_{r_\beta}\beta + C_{r_p}\frac{bp}{2V_a} + C_{r_r}\frac{br}{2V_a} + C_{r_{\delta a}}\delta_a + C_{r_{\delta r}}\delta_r\right] \tag{3.11}$$

$$\dot{\phi} = p + q\,sin\,\phi\,tan\,\theta + r\,cos\,\phi\,tan\,\theta \tag{3.12}$$

$$\dot{\psi} = q\,sin\,\phi\,sec\,\theta + r\,cos\,\phi\,sec\,\theta \tag{3.13}$$

$$\dot{\theta} = q\,cos\,\phi - r\,sin\,\phi \tag{3.14}$$

Keterangan dari parameter dan variabel yang digunakan pada model dinamika *fixed-wing* dapat dilihat pada Tabel 3.1.

Tabel 3.1 Parameter pada dinamika *Fixed-wing*

Parameter	Keterangan
α	*Angle of attack*
β	*Side slip angel*
J_x	Inersia pada sumbu-x
J_y	Inersia pada sumbu-y
J_z	Inersia pada sumbu-z
J_{xz}	Inersia pada sumbu dibentang sumbu x dan z
u, v, w	Kecepatan inersia
K_p	Konstanta *thrust*
K_d	Konstanta *drag*
m	Massa total UAV
g	Gravitasi
l_1	Jarak antara motor 1 ke motor 5
l_2	Jarak antara motor 1 ke pusat massa UAV
l_3	Jarak antara motor 3 ke pusat massa UAV
Γ	Produk dari Matrix Inersia
V_a	Vektor *airspeed*
V_g	Vektor kecepatan *ground*
V_w	Vektor kecepatan angin
C_L	Koefisien angkat (*fixed-wing*)
C_D	Koefisien *drag* (*fixed-wing*)
C_m	Koefisien momen $pitch$
C_l	Koefisien momen $roll$
C_n	Koefisien momen yaw
C_X	Koefisien gaya pada X_B
C_Y	Koefisien gaya pada Y_B
C_Z	Koefisien gaya pada Z_B
k_{motor}	konstanta efisiensi motor
S_{prop}	luas yang dihempas propeller
C_{prop}	Koefisien aerodinamis propeller
S	Area penampang sayap pesawat
c	*Maincord* sayap

Parameter	Keterangan
b	Bentang sayap
ρ	Densitas atmosfir
δ_e	Defleksi *elevator*
δ_a	Defleksi *aileron*
δ_r	Defleksi *rudder*
δ_t	Defleksi *thrust*

Parameter *Fixed-wing*

Parameter yang digunakan pada penelitian ini adalah parameter pada *fixed-wing* UltraStick-25e (Thor). Tabel 3.2 menampilkan parameter dari jenis *fixed-wing* yang digunakan.

Tabel 3.2 Parameter *fixed-wing* [16]

Parameter	Simbol	Nilai	Satuan
Wing span	B	1.72	m
Wing surface area	S	0.3097	m^2
Maincord	C	0.25	m
Massa	M	1.959	kg
Inersia	J_x	0.07151	$kg.m^2$
	J_y	0.08636	$kg.m^2$
	J_z	0.15364	$kg.m^2$
	J_{xz}	0.014	$kg.m^2$

3.4 Strategi Penghalusan Jalur menggunakan *Bézier curve*

Permasalahan penghalusan jalur dirumuskan sebagai masalah optimisasi dengan batasan-batasan yang diberikan. Tujuannya adalah untuk menghitung jalur yang diperkirakan pada jalur asli yang memungkinkan *fixed-wing* membuat manuver yang paling sedikit. Perhatikan bahwa ada hubungan antara kelayakan jalur dan optimalisasinya. Dengan penghalusan jalur, optimalisasi jalur dalam hal keselamatan dan panjang jalur dapat menurun. Gagasan yang diusulkan di sini adalah untuk menetapkan jalur baru dari penghalusan jalur yang dibuat di sekitar jalur asli berada dalam batasan koridor keselamatan dan sesuai kemampuan

manuver (*curvature – turn radius*). Oleh karena itu dua fungsi objektif pada (3.15)-(3.16) dipakai sebagai batasan dalam proses penghalusan jalur.

1. Batasan 1: Koridor keselamatan (*safety corridor*)

$$(Q_j - p(t_j)) \leq (D_j - \delta) \tag{3.15}$$

dimana:

Q_j : Data asli

$p(t_j)$: Nilai data yang diperkirakan dihitung oleh interpolasi

D_j : Jarak *Euclidean* ke *obstacle* terdekat di setiap titik jalur asli.

2. Batasan 2: Kemampuan manuver

   Koefisien kelengkungan kurva:

   a. $\frac{d^2y}{dx^2} = \frac{d^2y/dt^2}{\left(d^2x/dt^2\right)^2} < \frac{1}{R^2}$

   b. $\frac{d^2z}{dp^2} = \frac{d^2z/dt^2}{(dx/dt)(dy/dt)} < \frac{1}{R^2}$ (3.16)

dimana:

$p = f(x, y)$

R^2 = Manuver minimum *fixed-wing*

Jalur yang mulus harus berada dalam batasan koridor keselamatan. Sehingga harus dapat menentukan (dengan memecahkan masalah optimasi) titik kontrol yang menentukan bentuk jalur yang mulus yang dibentuk oleh *patches* Bézier *curve* yang terhubung. Properti interpolasi titik akhir dari *Bézier curve* memastikan bahwa lintasan dimulai pada P_s dan berakhir pada P_T. Fungsi objektif yang akan diminimalkan adalah rata-rata jarak kuadrat antara *real* data dengan data yang dihaluskan.

Misalkan ada m *node* Q_j (j = 1: m) dalam data asli. Untuk data yang dihaluskan $p(t_j)$, diketahui bahwa $0 \leq t_j \leq 1$ dengan pembagian yang sama, maka $Q_1 = p(0) = P_S$ dan $Q_m = p(1) = P_T$. Maka fungsi objektifnya menjadi:

$$\min_{p(t_j)} \left(\frac{1}{m-2} \sqrt{\sum_{j=2}^{m-1} (Q_j - p(t_j))^2} \right) \tag{3.17}$$

3.5 Perancangan Kontrol *Fixed-wing*

Kontroler *proportional integral derivative* (PID) yang digunakan pada penelitian ini berfungsi untuk mengatur gerak *fixed-wing* agar dapat melewati lintasan Theta* yang telah dihaluskan dengan algoritma Bézier curve menggunakan batasan *safety corridor* dan tambahan batasan baru yang diusulkan yaitu kemampuan manuver dari *fixed-wing*. Dalam kontrol *fixed-wing* terdapat *inner loop* dan *outer loop* dimana *inner loop* sebagai kontrol *attitude* dan *outer loop* sebagai kontrol posisi. Masukan dari kontrol posisi yaitu (X_e, Y_e, Z_e) dan keluaran nya berupa $(\phi_d, \theta_d, \psi_d)$ yang akan menjadi masukan untuk kontrol *attitude* seperti yang dijelaskan pada Gambar 3.4.

$$u_x = k_{px}(x_r - x) + K_{ix}\int(x_r - x)dt + k_{dx}(\dot{x}_r - \dot{x}) \tag{3.18}$$

$$u_y = k_{py}(y_r - y) + K_{iy}\int(y_r - h)dt + k_{dy}(\dot{y}_r - \dot{y}) \tag{3.19}$$

$$\delta_t = k_{pz}(z_r - h) + K_{iz}\int(z_r - z)dt + k_{dz}(\dot{z}_r - \dot{z}) \tag{3.20}$$

$$\delta_a = K_{p\phi}(\phi_d - \phi) + K_{i\phi}\int(\phi_d - \phi)dt + K_{d\phi}(p_d - p) \tag{3.21}$$

$$\delta_e = K_{p\theta}(\theta_d - \theta) + K_{i\theta}\int(\theta_d - \theta)dt + K_{d\theta}(q_d - q) \tag{3.22}$$

$$\delta_r = K_{p\psi}(\psi_r - \psi) + K_{i\psi}\int(\psi_r - \psi)dt + K_{d\psi}(r_r - r) \tag{3.23}$$

Nilai $\delta_t, \delta_a, \delta_e$ dan δ_r merupakan sinyal kontrol yang akan menjadi masukan untuk dinamika dari *fixed-wing*. Dari hasil tuning berdasarkan eksperimen, parameter kontrol PID yang digunakan pada penelitian ini adalah

$$k_{px} = 3, k_{ix} = 5.5, k_{dx} = 4$$

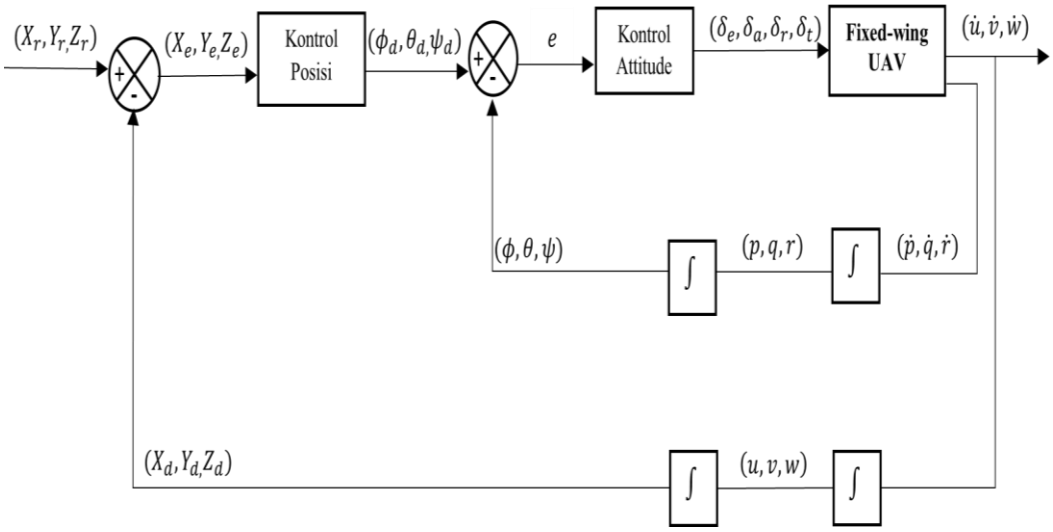
$$k_{py} = 3, k_{iy} = 5.5, k_{dy} = 4$$

$$k_{pz} = 1.5, k_{iz} = 2, k_{dz} = 2.5$$

$$k_{p\phi} = k_{p\theta} = k_{p\psi} = 2.5$$

$$k_{i\phi} = k_{i\theta} = k_{i\psi} = 2.75$$

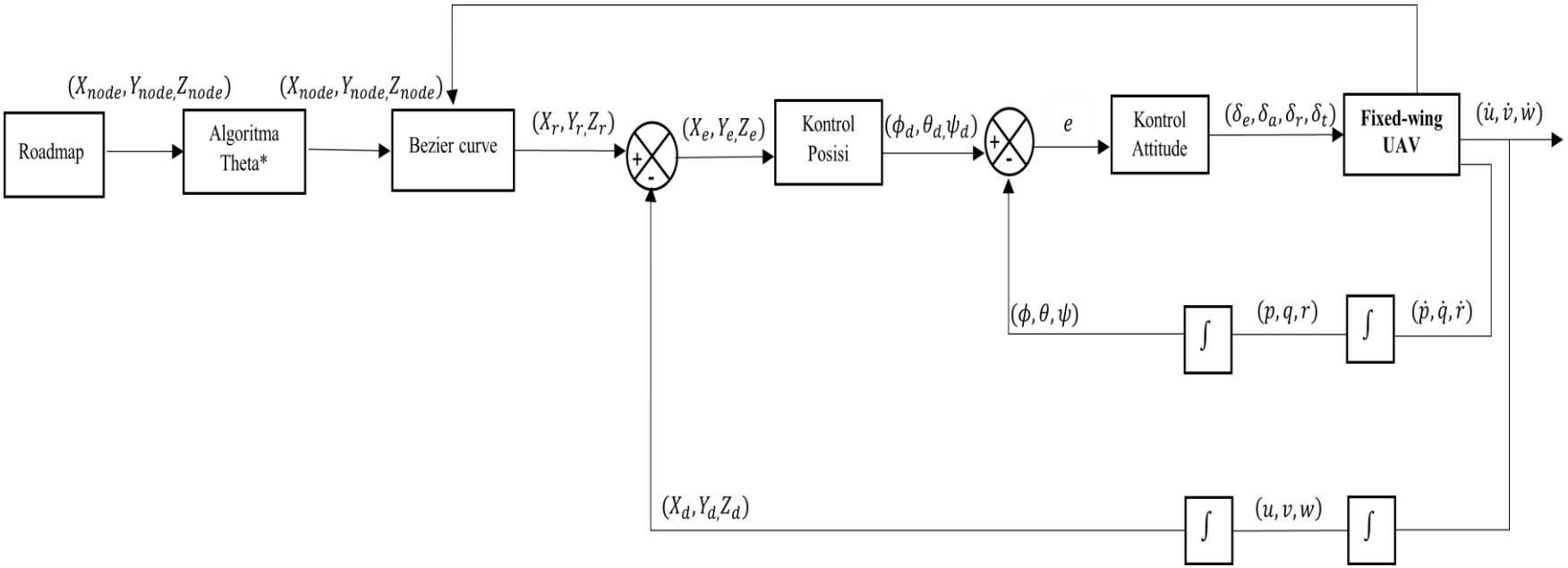
$$k_{d\phi} = k_{i\theta} = k_{d\psi} = 1.075$$



Gambar 3.5 Skema kontrol *fixed-wing*

3.6 Diagram Blok Keseluruhan

Pada tahapan perancangan diatas dapat diketahui tahapan-tahapan yang akan dikerjakan pada penelitian ini. Mulai dari perancangan lingkungan perkotaan 3D, pencarian jalur terpendek, penghalusan jalur serta pengujian jalur dengan membuat sistem *fixed-wing* terbang melewati jalur yang telah dihaluskan. Perancangan di atas juga dapat dilihat dalam bentuk blok diagram sistem pada.



Gambar 3.6 Blok diagram sistem keseluruhan

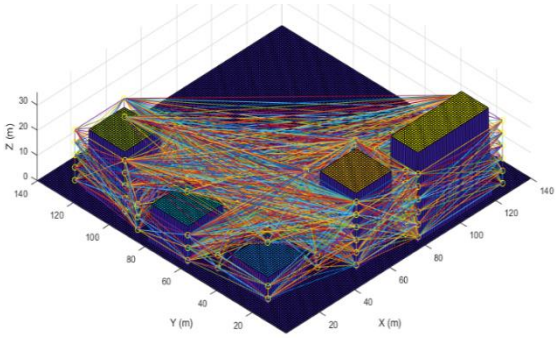
Halaman ini sengaja dikosongkan

BAB 4
HASIL DAN PEMBAHASAN

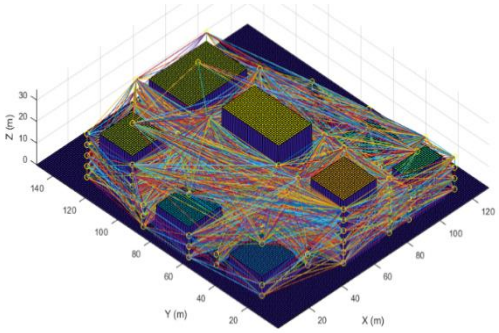
Pada bab ini akan dibahas tentang hasil algoritma yang digunakan untuk Menyelesaikan permasalahan *path planning* dan penghalusan jalur pada lingkungan 3D yang merepresentasikan lingkungan perkotaan. Untuk menguji hipotesis pada bab sebelumnya, eksperimen akan dilakukan pada setiap tahapan. Pengujian yang dilakukan adalah pengujian algoritma *visibility roadmap* sebagai pembuat lingkungan 3D serta pengujian algoritma Theta* sebagai algoritma pencari jalur terpendek. Algoritma-algoritma ini di simulasikan pada lingkungan yang memiliki jumlah gedung berbeda. Selanjutnya pengujian untuk mendapatkan manuver *fixed-wing*, nilai manuver yang didapatkan akan dimasukkan ke (3.16) sebagai syarat pembuatan jalur yang halus (*smoothing path*) yang diujikan dengan jumlah gedung berbeda-beda. Algoritma tersebut akan dibuat menggunakan fasilitas *script* pada perangkat lunak Matlab R2021a.

4.1 Pengujian Algoritma *Visibility Roadmap*

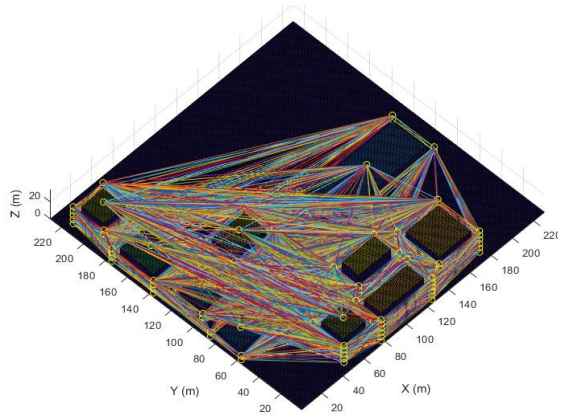
Pengujian pertama ini dilakukan untuk beberapa jumlah gedung. Tujuan dari pengujian ini yaitu untuk membuktikan bahwa algoritma yang dibuat mampu menyediakan *waypoint* dan lintasan pada lingkungan 3D dengan jumlah halangan yang beragam. Gambar 4.1 menunjukkan visualisasi *roadmap* dengan jumlah gedung.



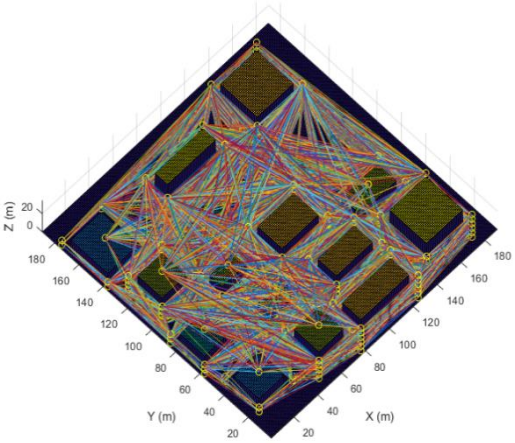
(a) 5 Gedung



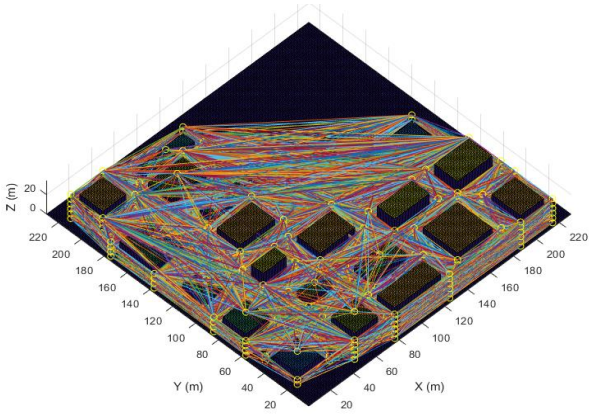
(b) 9 Gedung



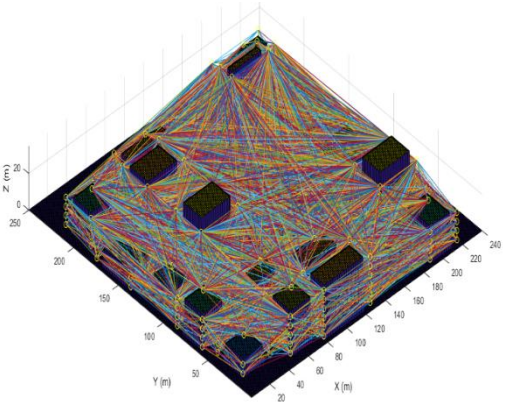
(c) 10 Gedung



(d) 15 Gedung



(e) 20 Gedung



(f) 25 Gedung

Gambar 4.1 Visualisasi roadmap dengan jumlah gedung berbeda-beda

Pada Gambar 4.1 dapat dilihat visualisasi dari *node* dan *edge* yang terhubung yang menghasilkan *traversable path,* yang kemudian akan diseleksi oleh

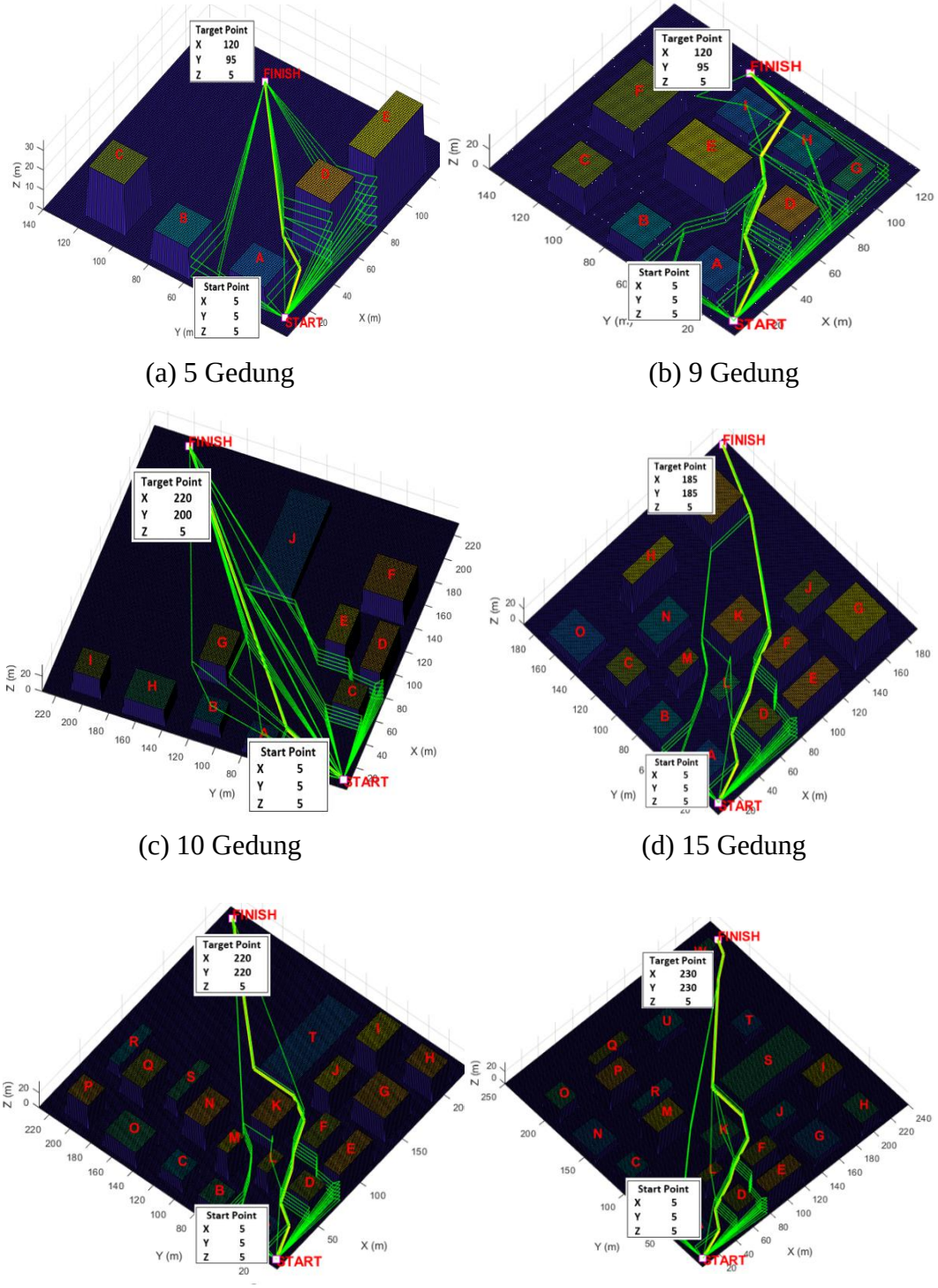
Theta* dalam pencarian jalur terpendek untuk menuju titik akhir yang diinginkan. Hasil dari algoritma *visibility roadmap* ini dapat dilihat pada Tabel 4.1 . Semakin banyak halangan atau gedung yang dibuat, maka waktu yang dibutuhkan juga semakin lama. Jumlah lintasan juga semakin banyak karena terjadinya peningkatan jumlah *node* yang dibuat.

Tabel 4.1 *Processing time* dari konstruksi *roadmap*

Jumlah gedung	*Processing time* (s)	Jumlah lintasan
5	12,46	1.622
9	37,46	3.968
10	73,90	5.123
15	215,71	9.358
20	703,20	12.906
25	1028,53	14.159

4.2 Pengujian Algoritma Theta*

Pada pengujian algoritma Theta* ini hasil yang diinginkan merupakan jalur yang terpendek. Saat Theta* memperluas *node* pencarian, jalur yang dibuat mungkin memiliki jumlah *node* yang lebih sedikit atau memiliki waktu komputasi yang lebih kecil, tetapi jalur yang diseleksi untuk eksperimen ini yaitu jalur yang menghasilkan jarak terpendek dari *node* awal menuju *node* tujuan. Pengujian algoritma ini juga dilakukan pada lingkungan 3D seperti pengujian *visibility roadmap* sebelumnya. Jalur hijau merupakan jalur perluasan pencarian dari algoritma Theta* dan jalur kuning merupakan jalur terpendek dari semua jalur perluasan tersebut, untuk visualisasi pencarian jalur terpendek menggunakan algoritma Theta* ditunjukkan oleh Gambar 4.2.



(a) 5 Gedung (b) 9 Gedung

(c) 10 Gedung (d) 15 Gedung

(e) 20 Gedung (f) 25 Gedung

Gambar 4.2 Visualisasi pencarian jalur terpendek dengan algoritma Theta* pada jumlah gedung berbeda-beda

Pengujian algoritma Theta* dilakukan dengan titik awal yang sama dan titik akhir yang berbeda-beda tetapi masih dalam satu arah. Hasil pengujian dapat dilihat pada Tabel 4.2, pada tabel menunjukkan bahwa waktu komputasi untuk algoritma Theta* relatif sangat kecil. Hal ini dikarenakan banyaknya informasi yang sudah tersedia dari algoritma pembuatan *map*, sehingga waktu komputasi pada saat pencarian jalur terpendek dari titik awal menuju titik tujuan menjadi kecil. Jumlah jalur yang ditampilkan diseleksi berdasarkan jalur terpendek. Sehingga dari banyaknya jalur yang diperluas, jalur kuning di setiap gedung yang berbeda pada Gambar 4.2 merupakan jalur yang dipilih.

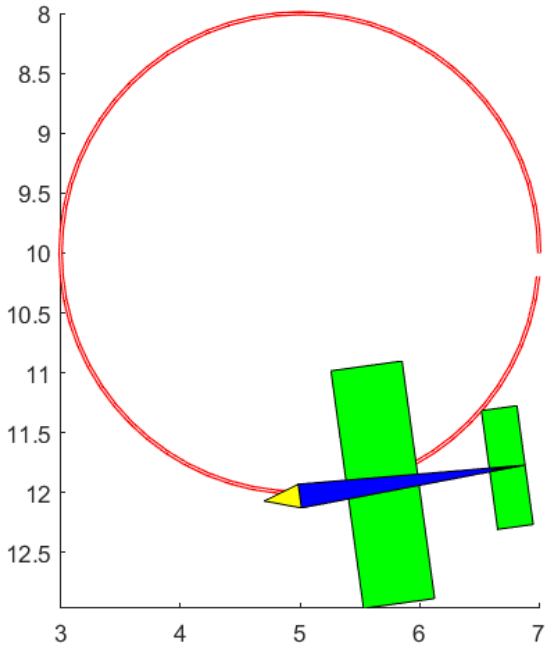
Tabel 4.2 Perbandingan jalur terpendek pada jumlah gedung berbeda-beda

Jumlah gedung	Jumlah jalur	Jalur terpendek (m)	Waktu komputasi (s)
5	29	39,01	0,010977
9	31	148,27	0,001965
10	37	54,06	0,003325
15	26	268,08	0,005492
20	27	127,01	0,002125
25	27	311,34	0,002385

Tabel 4.2 juga menunjukkan hasil dari jalur terpendek di beberapa banyak bangunan bervariasi. Jumlah gedung yang lebih sedikit tidak berarti memiliki jarak terpendek yang lebih kecil. Pada jumlah gedung 10, jarak terpendeknya sebesar 54,05 m sedangkan pada gedung 9 jarak terpendeknya 148,27 m. Hal ini dikarenakan berbedanya struktur dan letak gedung pada lingkungan 3D.

4.3 Pengujian Manuver *Fixed-wing*

Pengujian manuver *fixed-wing* ini dilakukan untuk mendapatkan nilai manuver. Nilai tersebut di masukkan ke persamaan (3.16) sebagai syarat pembuatan jalur yang halus. Pada pengujian manuver ini *fixed-wing* mampu membuat lingkaran berdiameter 4 m yang dapat dilihat pada Gambar 4.3. Jadi nilai R pada persamaan (3.16) merupakan nilai kurvatur dari lingkaran yang dibuat oleh *fixed-wing* tersebut.



Gambar 4.3 Batas maksimum manuver (*curvature-turn radius*) dari *fixed-wing*

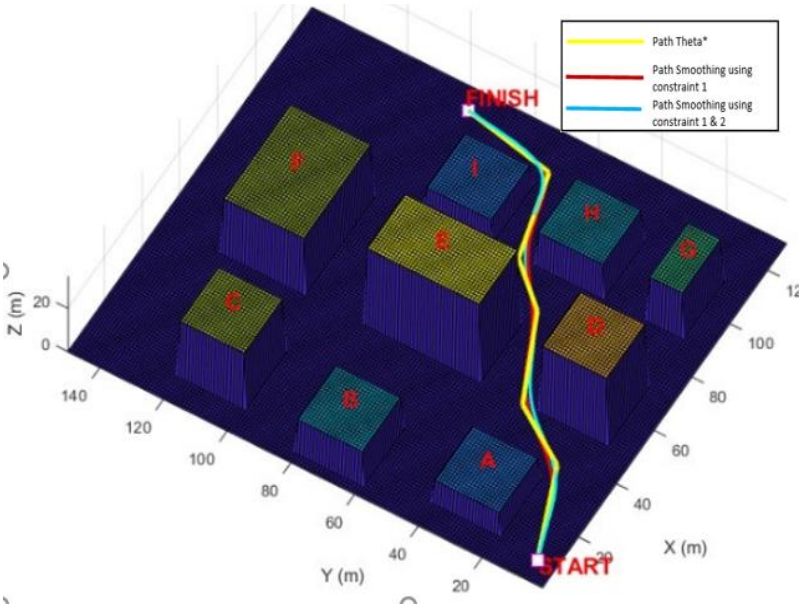
4.4 Pengujian Strategi Penghalusan Jalur berdasarkan Bézier Curve

Pengujian penghalusan jalur dilakukan pada dua jenis jumlah gedung yang berbeda yaitu pada lingkungan 3D dengan jumlah gedung 9 dan 15. Hal ini dilakukan karena melihat pada jalur asli atau jalur yang dipilih Theta* dapat dilihat bahwa lingkungan 3D pada Gambar 4.2 dengan 9 gedung memiliki belokan jalur yang tajam pada setiap titik yang dilalui. Sedangkan lingkungan dengan jumlah gedung 15 terlihat lebih halus. Oleh karena itu, pengujian untuk penghalusan jalur ini dilakukan pada 2 jenis lingkungan tersebut.

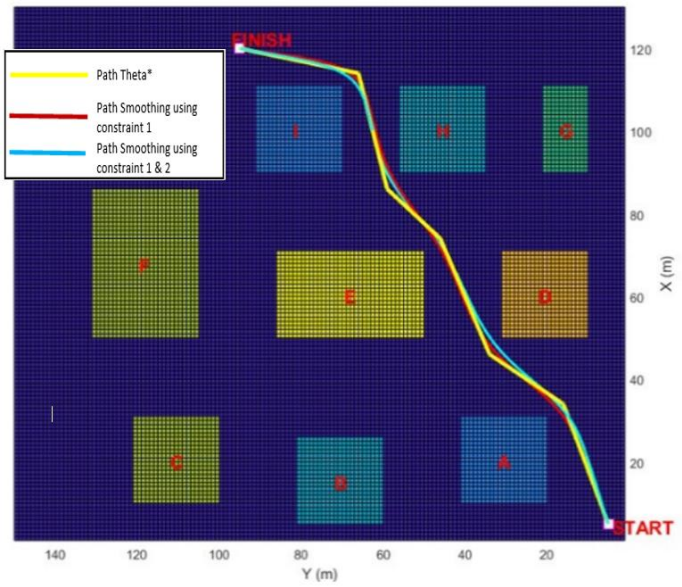
Pengujian penghalusan jalur dengan metode yang diusulkan ini juga akan dibandingkan langsung dengan strategi penghalusan jalur yang diusulkan pada penelitian sebelumnya. Beberapa parameter pembanding di berikan untuk melihat performa dari metode penghalusan jalur yang diusulkan.

4.4.1 Pengujian Lingkungan 3D 9 Gedung

Pengujian pertama dilakukan pada lingkungan 3D dengan 9 gedung. Visualisasi perbandingan jalur dari metode penghalusan jalur yang diusulkan dengan metode penghalusan pada penelitian sebelumnya dapat dilihat pada Gambar 4.4 dan Gambar 4.5.



Gambar 4.4 Tampilan 3D visualisasi penghalusan jalur pada lingkungan 3D 9 gedung



Gambar 4.5 Tampilan 2D visualisasi penghalusan jalur pada lingkungan 3D 9 gedung

Pada Gambar 4.4 dan Gambar 4.5 dapat dilihat jalur yang berwarna kuning merupakan jalur asli sebelum di haluskan. Jalur yang berwarna merah merupakan hasil penghalusan jalur menggunakan metode dari penelitian sebelumnya, dimana perkiraan penghalusan jalur di sekitar jalur asli harus berada di dalam koridor yang aman. Jalur yang berwarna biru merupakan hasil penghalusan jalur yang sudah menambahkan syarat manuver (*curvature - maximum turn radius*) pada saat pemilihan jalur Bézier yang dibuat.

Tabel 4.3 Hasil perbandingan penghalusan jalur pada lingkungan 3D 9 gedung

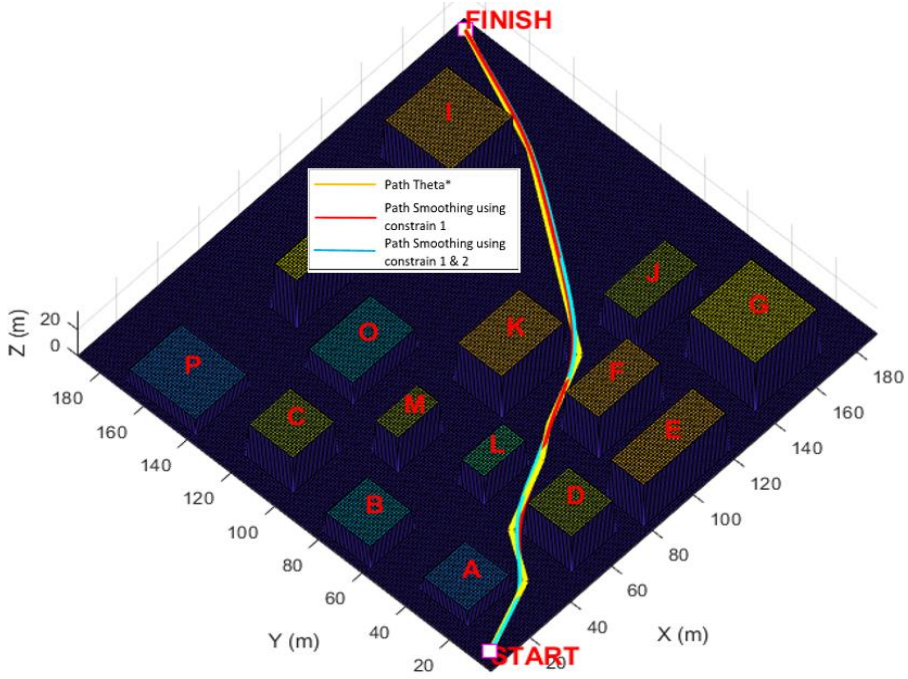
Perbandingan	Bézier *Curve*: Batasan 1	Bézier *Curve*: Batasan 1 dan 2
Panjang Jalur (m)	142,36	140,54
Jarak terkecil ke halangan (m)	3,51	4,73
Rata-rata jarak ke halangan (m)	7,25	7,84
Maksimum *smoothing* gap (cm)	338,53	380,44
Rata-rata *smoothing* gap (cm)	88,99	114,93
Perubahan nilai ketinggian maksimum (cm)	34,56	73,59
Perubahan nilai sudut *heading* maksimum (deg)	20,45	27,29

Berdasarkan

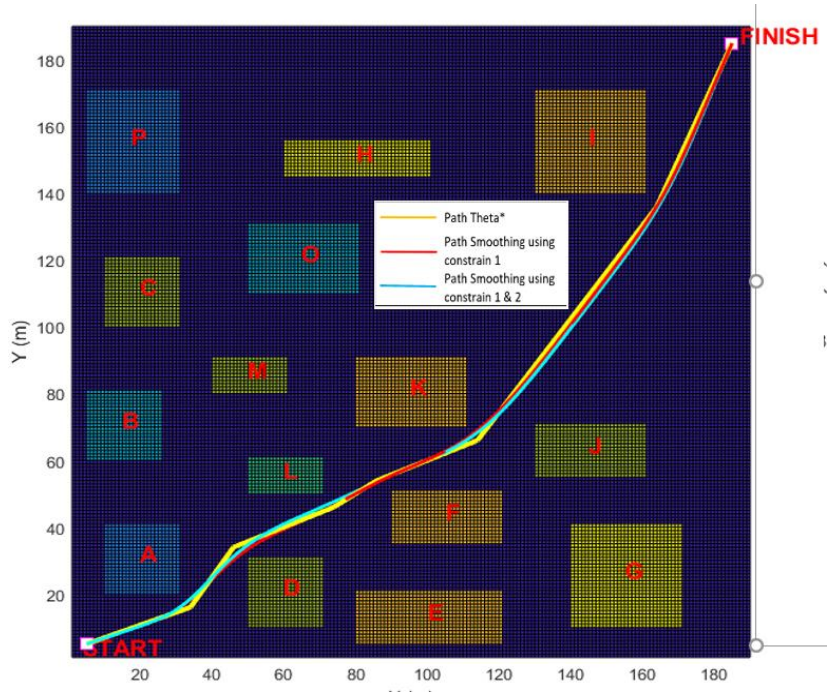
Tabel 4.3 dapat dilihat bahwa penghalusan jalur dengan penambahan syarat manuver (Bézier *curve*: batasan 1 dan 2) menghasilkan jalur yang lebih pendek dengan pengurangan sebesar 5,37 % dari panjang jalur asli. Jarak terkecil ke halangan juga lebih besar dengan rata-rata jarak ke halangan relatif lebih jauh dari pada mengunakan metode penelitian sebelumnya (Bézier *curve*: batasan 1), sehingga memberi lintasan yang lebih aman untuk *fixed-wing*. Jalur asli dari lingkungan 3D 9 gedung ini terlihat tajam, sehingga dengan memasukkan syarat manuver pada pembuatan penghalusan jalur menghasilkan perubahan nilai sudut *heading* maksimum $\Delta\psi_{max}$ yang lebih besar dibandingkan metode penelitian sebelumnya yaitu sebesar 27,29 deg.

4.4.2 Pengujian Lingkungan 3D 15 Gedung

Pengujian kedua dilakukan pada lingkungan 15 gedung. Visualisasi hasil perbandingan jalur dapat dilihat pada Gambar 4.6 dan Gambar 4.7.



Gambar 4.6 Tampilan 3D visualisasi penghalusan jalur pada lingkungan 3D 15 gedung



Gambar 4.7 Tampilan 2D visualisasi penghalusan jalur pada lingkungan 3D 15 gedung

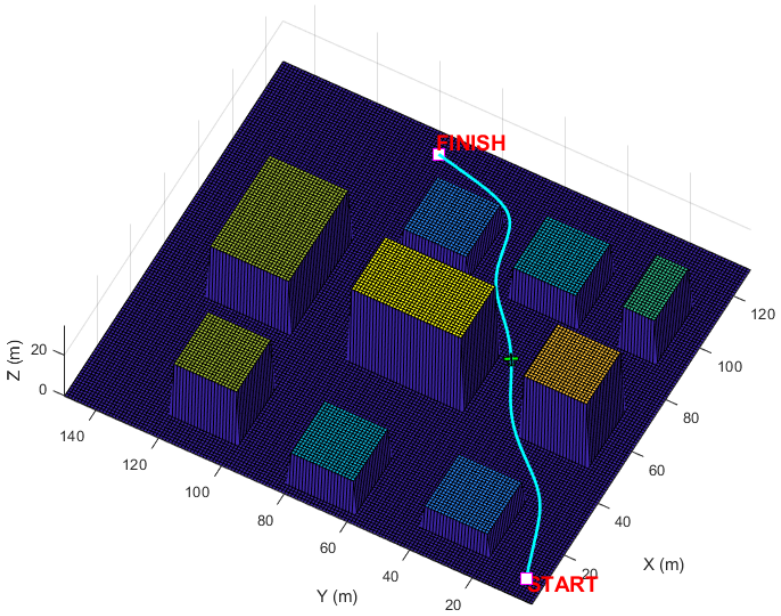
Pada pengujian di lingkungan 3D dengan 15 gedung ini, nilai perbandingan dari kedua metode memiliki perbedaan yang tidak terlalu signifikan. Hal ini dikarenakan jalur asli pada lingkungan ini terlihat lebih halus, dengan perubahan *heading* yang kecil. Namun, hasil dari metode penghalusan jalur yang diusulkan tetap memberikan performa yang lebih baik seperti pada pengujian di lingkungan 3D 9 gedung, hasil perbandingan antara kedua metode yang digunakan dapat dilihat pada Tabel 4.4.

Tabel 4.4 Hasil perbandingan penghalusan jalur pada lingkungan 3D 15 gedung

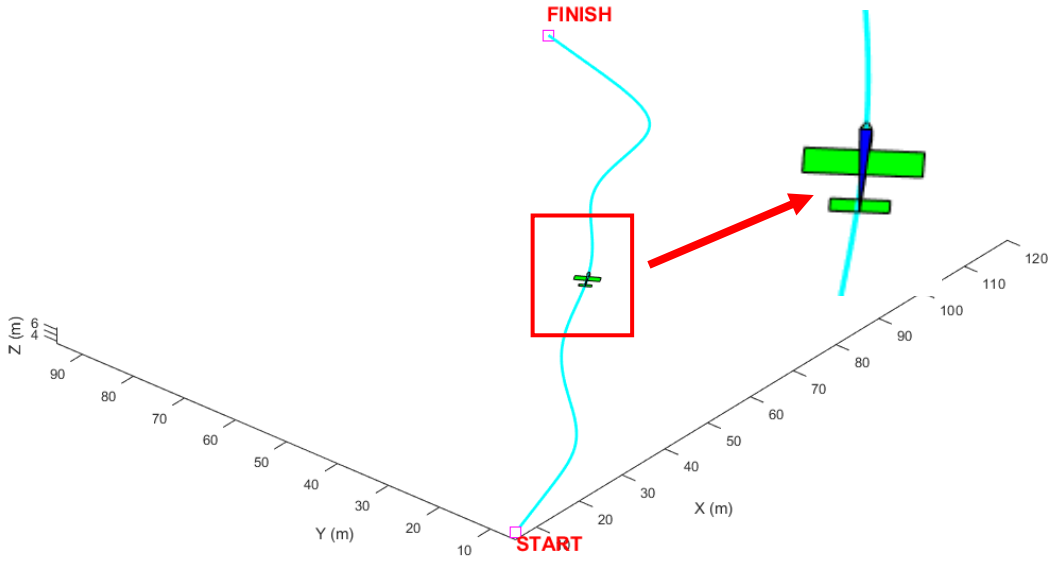
Perbandingan	Bézier *Curve*: Batasan 1	Bézier *Curve*: Batasan 1 dan 2
Panjang Jalur (m)	262,55	261,17
Jarak terkecil ke halangan (m)	4,34	5,50
Rata-rata jarak ke halangan (m)	12,06	12,11
Maksimum *smoothing* gap (cm)	466,55	509,26
Rata-rata *smoothing* gap (cm)	100,64	108,16
Perubahan nilai ketinggian maksimum (cm)	12,78	13,75
Perubahan nilai sudut *heading* maksimum (deg)	5,72	6,10

4.5 *Tracking* Jalur

Pada Gambar 4.8 terlihat bahwa jalur Theta* yang telah dihaluskan dengan metode penghalusan jalur yang diusulkan yaitu Bézier *curve* dengan batasan 1 dan 2 akan diuji. Caranya dengan menerbangkan *fixed-wing* pada jalur tersebut menggunakan kontroler PID sederhana sebagai penggerak. Gambar 4.8 (a) menunjukkan visualisasi *fixed-wing* yang terbang di antara gedung-gedung. Sedangkan Gambar 4.8 (b) hanya menampilkan jalur yang telah dihaluskan dan *fixed-wing* saja.



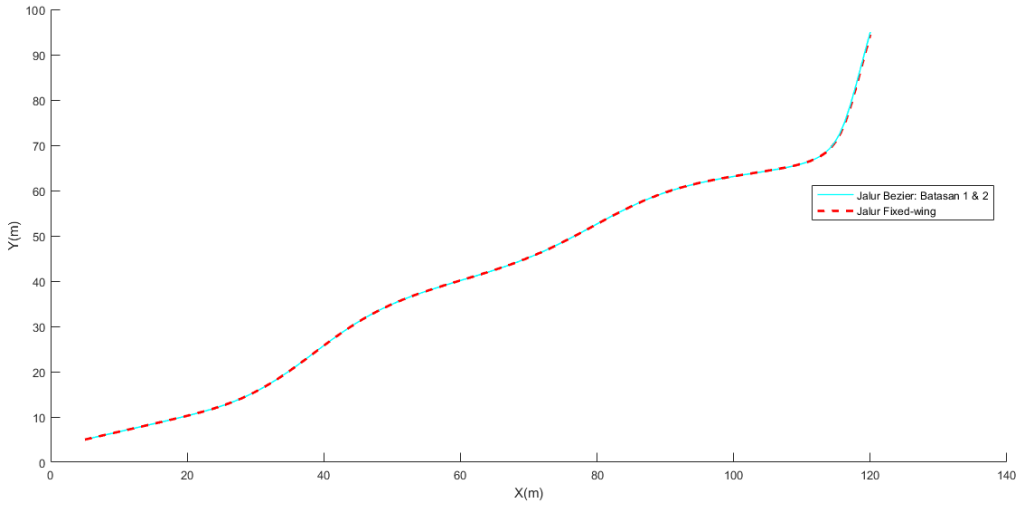
(a)



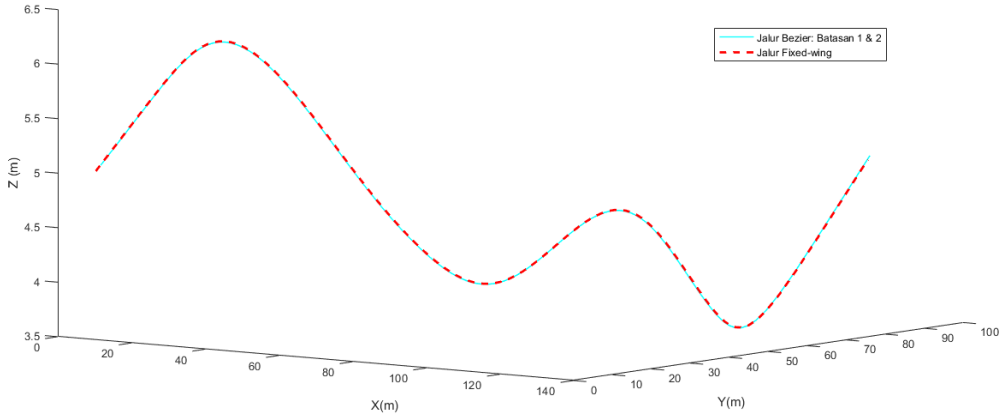
(b)

Gambar 4.8 Visualisasi *tracking* jalur oleh *fixed-wing*. (a) dengan gedung. (b) tanpa gedung.

Pada Gambar 4.9 dapat dilihat bahwa *fixed-wing* dengan akurat dapat mengikuti jalur yang dihasilkan. Hal ini membuktikan bahwa jalur yang dihaluskan dengan metode Bézier *curve* yang diusulkan kongruen dengan gerakan *fixed-wing*.



(a)



(b)

Gambar 4.9 Hasil simulasi *tracking* jalur oleh *fixed-wing*. (a) 2D View. (b) 3D View

4.6 Pengujian Bézier *Curve* dengan Variasi Kurvatur

Pengujian ini dilakukan untuk melihat bagaimana hasil perbandingan dari jalur yang dihasilkan jika kurvatur dibuat bervariasi yang dapat dilihat pada Tabel 4.5.

Tabel 4.5 Pengujian Bézier Curve dengan Variasi Kurvatur

Perbandingan	Bézier *Curve*: Batasan 1	Bézier *Curve*: Batasan 1 dan 2			
		$D = 3$ $r = 1.5$ $R = \frac{1}{1.5}$	$D = 4$ $r = 2$ $R = \frac{1}{2}$	$D = 5$ $r = 2.5$ $R = \frac{1}{2.5}$	$D = 6$ $r = 3$ $R = \frac{1}{3}$
Panjang Jalur (m)	142,36	144.17	140,54	141.31	141.86
Jarak terkecil ke halangan (m)	3,51	4.27	4,73	4.01	3.85
Rata-rata jarak ke halangan (m)	7,25	8.02	7,84	7.51	7.47
Maksimum *smoothing* gap (cm)	338,53	470.51	380,44	493.78	331.40
Rata-rata *smoothing* gap (cm)	88,99	126.52	114,93	116.94	95.17
Perubahan nilai ketinggian maksimum (cm)	34,56	30.76	73,59	69.73	24.14
Perubahan nilai sudut *heading* maksimum (deg)	20,45	21.44	27,29	24.90	25.96

Tabel 4.5, menampilkan hasil perbandingan dari nilai kurvatur yang bervariasi. $R = \frac{1}{2}$ merupakan nilai kurvatur yang didapatkan dari pengujian fixed wing ultrastick-25. Pada saat nilai kurvatur diubah menjadi $R = \frac{1}{1.5}$ panjang jalur yang dihasilkan menjadi sedikit lebih panjang dibandingkan dengan jalur Bezier dengan batasan 1 yaitu 144.17 m, tetapi nilai jarak terkecil ke halangan dan rata-rata jarak ke halangan memberikan nilai jarak yang lebih besar. Sedangkan untuk $R = \frac{1}{2.5}$ dan $R = \frac{1}{3}$ hasil keseluruhan dari perbandingan tidak memberikan perubahan yang signifikan dengan nilai curvature $R = \frac{1}{2}$ yang merupakan hasil pengujian dari pengujian sebelumnya.

Halaman ini sengaja dikosongkan

BAB 5
PENUTUP

5.1 Kesimpulan

Tesis ini membahas tentang perencanaan penghalusan jalur untuk *fixed-wing* pada lingkungan 3D. Algoritma *visibility roadmap* digunakan sebagai algoritma pembuat lingkungan 3D untuk merepresentasikan lingkungan perkotaan. Lingkungan 3D ini menyediakan *node* dan semua kemungkinan *traversable path*. Setelah ditentukan titik awal dan titik tujuan pada *roadmap,* Algoritma Theta* diadaptasi untuk menentukan jalur terpendek dari titik awal ke tujuan. Penghalusan jalur pada tesis ini menggunakan metode Bézier *curve* dengan memberikan beberapa batasan pada saat *pre-smoothing*. Batasan pertama diadaptasi dari penelitian sebelumnya, yaitu jalur yang dihaluskan di sekitar jalur asli harus berada dalam koridor keselamatan. Kemudian pada tesis ini menambahkan satu batasan lagi yaitu jalur yang dihaluskan harus memenuhi batasan manuver *fixed-wing*.

Batasan manuver *(curvature-maximum turn radius*) ini didapatkan dari pengujian simulasi *fixed-wing*. Setelah mendapatkan jalur yang telah dihaluskan, *fixed-wing* akan diterbangkan atau melacak jalur tersebut dengan menggunakan kontrol PID sederhana. Simulasi pada penelitian ini menunjukkan bahwa semua algoritma yang digunakan berhasil dijalankan. Metode penghalusan jalur yang diusulkan memberikan performa yang lebih baik dilihat dari segi keamanan dan jalur yang lebih halus. Hal ini dibuktikan bahwa pada pengujian 9 gedung memiliki jarak terkecil ke halangan yang lebih besar dibandingkan metode sebelumnya yaitu 4,73 m. Jika dilihat dari segi jalur yang lebih halus, pada 9 gedung metode yang diusulkan menghasilkan nilai perubahan sudut *heading* maksimum yang lebih besar yaitu 27,29 deg yang menyebabkan jalur yang dihasilkan lebih halus dibandingkan metode pada penelitian sebelumnya.

5.2 Saran

Pada penelitian selanjutnya, penulis menyarankan untuk menggunakan metode pada tesis ini secara *realtime* di lingkungan yang dinamis. Metode disimulasikan secara *realtime* agar jalur yang dihasilkan dapat terus diperbarui sesuai dengan perubahan lingkungan, namun harus tetap memenuhi batasan-batasan yang digunakan pada tesis ini.

DAFTAR PUSTAKA

[1] A. H. Qureshi and Y. Ayaz, "Intelligent bidirectional rapidly-exploring random trees for optimal motion planning in complex cluttered environments," *Robotics and Autonomous Systems*, vol. 68, 2015, doi: 10.1016/j.robot.2015.02.007.

[2] T. Agustinah, A. Jazidie, M. Fuad, and F. A. Setiawan, "Evading of Pedestrian Pursuer and Avoiding Obstacles Using Path-Velocity Planner," *IEEE Access*, vol. 9, 2021, doi: 10.1109/access.2021.3133383.

[3] I. K. Nikolos, K. P. Valavanis, N. C. Tsourveloudis, and A. N. Kostaras, "Evolutionary Algorithm Based Offline/Online Path Planner for UAV Navigation," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 33, no. 6, 2003, doi: 10.1109/TSMCB.2002.804370.

[4] B. Ranjbar-Sahraei *et al.*, "Bio-inspired multi-robot systems," in *Biomimetic Technologies: Principles and Applications*, 2015. doi: 10.1016/B978-0-08-100249-0.00013-6.

[5] E. S. Kaur, "Shortest Path Finding Algorithm Using Ant Colony Optimization," vol. 2, no. 6, pp. 419–427, 2013.

[6] B. Dasgupta, "Robot Motion Planning Methods : An Overview," no. April, 2007.

[7] R. Oliva and N. Pelechano, "NEOGEN: Near optimal generator of navigation meshes for 3D multi-layered environments," *Computers and Graphics (Pergamon)*, vol. 37, no. 5, 2013, doi: 10.1016/j.cag.2013.03.004.

[8] M. de Berg, O. Cheong, M. van Kreveld, and M. Overmars, *Computational geometry: Algorithms and applications*. 2008. doi: 10.1007/978-3-540-77974-2.

[9] L. de Filippis, G. Guglieri, and F. Quagliotti, "Path planning strategies for UAVS in 3D environments," *Journal of Intelligent and Robotic Systems: Theory and Applications*, vol. 65, no. 1–4, 2012, doi: 10.1007/s10846-011-9568-2.

[10] P. E. Hart, N. J. Nilsson, and B. Raphael, "A Formal Basis for the Heuristic Determination of Minimum Cost Paths," *IEEE Transactions on Systems Science and Cybernetics*, vol. 4, no. 2, 1968, doi: 10.1109/TSSC.1968.300136.

[11] Z. Ahmad, F. Ullah, C. Tran, and S. Lee, "Efficient Energy Flight Path Planning Algorithm Using 3-D Visibility Roadmap for Small Unmanned Aerial Vehicle," *International Journal of Aerospace Engineering*, vol. 2017, 2017, doi: 10.1155/2017/2849745.

[12] K. Benzaid, R. Marie, N. Mansouri, and O. Labbani-Igbida, “Filtered Medial Surface Based Approach for 3D Collision-Free Path Planning Problem,” *Journal of Robotics*, vol. 2018, 2018, doi: 10.1155/2018/4676720.

[13] “Modelling of a Small Unmanned Aerial Vehicle,” *Advances in Robotics & Automation*, vol. 04, no. 01, 2015, doi: 10.4172/2168-9695.1000126.

[14] K. Daniel, A. Nash, S. Koenig, and A. Felner, “Theta*: Any-angle path planning on grids,” *Journal of Artificial Intelligence Research*, vol. 39, 2010, doi: 10.1613/jair.2994.

[15] K. H. Ang, G. Chong, and Y. Li, “PID control system analysis, design, and technology,” *IEEE Transactions on Control Systems Technology*, vol. 13, no. 4, 2005, doi: 10.1109/TCST.2005.847331.

[16] A. Manual, “Ultra Stick 25e”.

LAMPIRAN

LAMPIRAN A - Visibility Roadmap

```
clc;
clear all;
close all;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
wing=2;

%Nama Gedung
gdg=['A','B','C','D','E','F','G','H','I','J','K','L','M','O','P'
,'Q','R','S','T','U','V','W','X','Y','Z'];

% number slice
nsl=5;
hmin=5;
hmax=30;

%membuat titik pojok gedung
nb=[5]; % Jumlah Gedung

time_making_LOS=[];
for i=1:length(nb)
    n=nb(i);
    [xy,h]=build_coord(n);


[xyz_edge,zh,area_building,corner_all_building]=building3d_v4(xy
,h,nsl,wing,hmin,hmax);

    save([num2str(n) 'building_corner'],'corner_all_building');
    save([num2str(n) 'edge_coord_v2'],'xyz_edge');
    save([num2str(n) 'high_coord_v2'],'zh');
    save([num2str(n) 'building_coord_v2'],'area_building');

    figure,surf(zh);hold on; %membuat gambar gedung
    axis tight
    axis equal
    view(45*7,20);

    % menggambar titik-titik way point

x_edge=xyz_edge(:,1);y_edge=xyz_edge(:,2);z_edge=xyz_edge(:,3);
    plot3(x_edge,y_edge,z_edge,'yo');
    xlabel('X (m)')
    ylabel('Y (m)')
    zlabel('Z (m)')

    %membuat seluruh edge pada waypoint
    s=0;

    %garis antar node
    tic
    dd=0.5;
    lines_d=LOSline_map3(xyz_edge(:,1:3),s,area_building,1,dd);
```

```
    num_lines=size(lines_d,1);
    elapsed_time=toc;

    %Building LOS
    save([num2str(n) 'building_LOS'],'lines_d');

    time_making_LOS=[time_making_LOS;[elapsed_time num_lines
dd]];
end
disp('=============================================================')
disp(' Jumlah gedung  |  Waktu (s)  | Jumlah lintasan');
disp('=============================================================')
for i=1:length(nb)
    spasi=['      '];
    disp([spasi num2str(nb(i)) spasi spasi
num2str(time_making_LOS(i,:))])
end
disp('=============================================================')
```

```
function
[xyz_edge,zh,area_building,corner_all_building]=building3d_v4(xy
,h,nsl,wing,hmin,hmax)

%input
%xy base building coodinat
%h building height
%nsl number of slize level
%dsl distance of slice
%wing eff radius uav

%output
%zh building map 3d
%xyz_edge edge slice coordinat
duav=wing*2;

x=xy(:,1);
n=length(h);
xh=reshape(x,4,n);
xh=xh';

y=xy(:,2);
yh=reshape(y,4,n);
yh=yh';

for i=1:n
    xhmin(i)=min(xh(i,:));
    xhmax(i)=max(xh(i,:));
    yhmin(i)=min(yh(i,:));
    yhmax(i)=max(yh(i,:));
end

%Area 100x100
x=1:0.2:max(x)+20;
y=1:0.2:max(y)+20;
zh=zeros(max(x),max(y));
% figure;
```

```
for k=1:n
    %     k
    xl=xhmin(k);
    xr=xhmax(k);
    yf=yhmax(k);
    yr=yhmin(k);
    xh=[];yh=[];
    for i=1:length(x)
        if i>=xl&i<=xr
            xh=i;
            for j=1:max(x)
                if j>=yr&j<=yf
                    yh=j;
                    zh(xh,yh)=h(k);
%                     zh=h(k);
                end
            end

        end
    end
end

% number slice
nsl=5;%number slice
hmin=5;% min high uav fly
hmax=30;% max high uav fly
hslice=[];
dslice=floor((hmax-hmin)/nsl);
for i=1:nsl+1
    hs=i*dslice;
    hslice=[hslice hs];
end
hslice;

xsl=[]; ysl=[]; zsl=[];

edges_all_building=[];
corner_all_building=[];
area_building=[];
for k=1:n
    %     k
    xl=xhmin(k)-duav;
    xr=xhmax(k)+duav;
    yf=yhmax(k)+duav;
    yr=yhmin(k)-duav;
    high=h(k)+wing;
    area_building=[area_building;[yf yr xl xr high]];
    edges_all=[];
    corner_all=[];
    nsl_b=floor(h(k)/dslice)+1;%number slice per building
    if nsl_b>6
        nsl_b=6;
    end

    for l=1:nsl_b
        hsl=hslice(l);
        edgesl=[xl yf hsl k
```

```
            xl yr hsl k
            xr yf hsl k
            xr yr hsl k];
        corner=[xhmin(k) yhmax(k) hsl k
                xhmin(k) yhmin(k) hsl k
                xhmax(k) yhmax(k) hsl k
                xhmax(k) yhmin(k) hsl k];

        if l==nsl_b
            if (hsl-h(k))>duav
                xsl=round((xl+xr)/2);
                ysl=round((yf+yr)/2);
                edgesl=[xsl ysl hsl k];
                corner=[xsl ysl h(k) k];
            elseif hsl<=h(k)
                xsl=round((xl+xr)/2);
                ysl=round((yf+yr)/2);
            end
        end
        edges_all=[edges_all;edgesl];
        corner_all=[corner_all;corner];
    end
    edges_all_building=[edges_all_building;edges_all];
    corner_all_building=[corner_all_building;corner_all];
end
corner_all_building=[corner_all_building(:,2)
corner_all_building(:,1) corner_all_building(:,3)];
edges_slice=edges_all_building;

x_edge=edges_slice(:,2);
y_edge=edges_slice(:,1);
z_edge=edges_slice(:,3);
bldk=edges_slice(:,4);
xyz_edge=[x_edge y_edge z_edge bldk];
```

LAMPIRAN B – ALGORITMA THETA*

```
clc;
clear all;
close all;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%path parameters
%Starting point
x0=5;y0=5;z0=5;

%Arrival point
xend=120;yend=95;zend=5; %%5 gedung
% xend=120;yend=95;zend=5; %%9 gedung
% xend=220;yend=200;zend=5; %%10 gedung
% xend=185;yend=185;zend=5; %%15 gedung
% xend=220;yend=220;zend=5; %%20 gedung
% xend=230;yend=230;zend=5; %25 gedung
wing=2;

%Nama Gedung
```

```
gdg=['A','B','C','D','E','F','G','H','I','J','K','L','M','N','O'
,'P','Q','R','S','T','U','V','W','X','Y','Z'];

% number slice
nsl=5;
hmin=5;
hmax=30;

%membuat titik pojok gedung
%edge of n-building and high of buildings
n=5; % Jumlah Gedung

load([num2str(n) 'building_corner'],'corner_all_building');
load([num2str(n) 'edge_coord_v2'],'xyz_edge');
load([num2str(n) 'high_coord_v2'],'zh');
load([num2str(n) 'building_coord_v2'],'area_building');

figure,surf(zh);hold on; %membuat gambar gedung
axis tight
axis equal
view(45*7,20);

% menggambar titik-titik way point
x_edge=xyz_edge(:,1);y_edge=xyz_edge(:,2);z_edge=xyz_edge(:,3);

%index building
[xy,h]=build_coord(n)
nb=size(xy,1)/4;
for i=1:nb
    xsl=mean(xy((i-1)*4+1:(i-1)*4+4,2));
    ysl=mean(xy((i-1)*4+1:(i-1)*4+4,1));
    zsl=h(i);

text(xsl,ysl+2,zsl+3,gdg(i),'FontWeight','bold','color','red','F
ontSize',14);
end

%panggil data LOS jika sudah ada
load([num2str(n) 'building_LOS'],'lines_d');

%garis LOS dari start ke node pertama
%draft start-finish point
start=[x0 y0 z0];
finish=[xend yend zend];
edges_choice1=[finish;xyz_edge(:,1:3)];
lines_q=find_qstart(start,edges_choice1,area_building);

%garis dari node terakhir ke finish
edges_choice2=[start;xyz_edge(:,1:3)];
lines_p=find_pfinish(finish,edges_choice2,area_building);
lines=[lines_q;lines_d;lines_p];

plot3(x0,y0,z0,'ms','MarkerSize',10,'MarkerFaceColor','auto');
plot3(xend,yend,zend,'ms','MarkerSize',10,'MarkerFaceColor','aut
o');
text(x0-1,y0-1,z0-
1,'START','FontWeight','bold','color','red','FontSize',14);
```

```
text(xend+2,yend+2,zend+2,'FINISH','FontWeight','bold','color','
red','FontSize',14);
xlabel('X (m)')
ylabel('Y (m)')
zlabel('Z (m)')

pqLOS=lines;
pa=pqLOS(:,1:3);
qa=pqLOS(:,4:6);
pqLOSa=[[pa qa];[qa pa]];
ps=pqLOSa(:,1:3);
qs=pqLOSa(:,4:6);

%determine q from start poin
starts=lines_q;
num_path=size(starts,1)
all_path=zeros(17,3*num_path);
all_range=[];
time=[];
num_node=[];
z_diff=[];
for k=1:num_path

    step=10;
    p=starts(k,4:6);
    p_all=[];
    r0=find_range(start,p);
    [x,y,z]=makeline3d(start,p);

    range_total=r0;
    zd=0;
    for i=1:step
        %         p
        tic;
        p_all=[p_all;p];
        nq=find_matrix(ps,p);
        lnq=length(nq);
        range_q=[];
        for j=1:length(nq)
            q=qs(nq(j),:);
            np=find_matrix2(p_all,[q(1) q(2)]);
            lnp=length(np);
            if length(np)==0
                range1=find_range(p,q);
                range2=find_range(q,finish);
                range=range1+range2;

                P1=p;
                P2=finish;
                P0=q;
                ang=find_angle3d(P0,P1,P2);
                data_q=[q range1 range2 range 180-ang];
                range_q=[range_q;data_q];
            end
            range_q;

        end
```

```
        sortir_q=sortrows(range_q,6);
        n=1;
        q_shortest=sortir_q(1:n,1:3);
        z1=p(3);
        z2=q_shortest(3);
        if z1~=z2
            zd=zd+1;
        end
        if q_shortest==finish
             time_searching=toc;


            p_all=[p_all;finish];
            break
        end

        range_shortest=sortir_q(1:n,4:7);
        range_total=range_total+range_shortest(1);
        sudut=180-range_shortest(4);
        p=q_shortest(1,:);
    end

    nodes=[start;p_all];
    all_path(1:size(nodes,1),(k-1)*3+1:k*3)=nodes;
    all_range=[all_range;[k range_total]];
    time=[time;time_searching];
    num_node=[num_node;size(nodes,1)];
    z_diff=[z_diff;zd];
end

perf_allpath=[all_range time num_node-2 z_diff num_node];

disp(' ');
disp('            Peringkat Jarak Start-Finish       ');
disp('-------------------------------------------------------');
disp(' No.  | Panjang   |  Waktu          |  heading    |
perubahan ');
disp('      | jalur     |  komputasi      |             |
ketinggian');
disp('-------------------------------------------------------');

perf_allpath_sortrange=sortrows(perf_allpath,2);
disp(num2str(perf_allpath_sortrange))

% plot line from start to finish
for i=1:num_path
    path_i=all_path(:,(i-1)*3+1:(i-1)*3+3);
    for j=1:10
        p=path_i(j,:);
        q=path_i(j+1,:);
        X=[p(1) q(1)];
        Y=[p(2) q(2)];
        Z=[p(3) q(3)];

        if q==[0 0 0]
            break;
```

```
        else
            if i==1
                line(X,Y,Z,'Color','yellow','LineWidth',3);
            else
                line(X,Y,Z,'Color','green');
        end
    end
    perf_allpath_sortrange(i,1:2)
    pause
end
display(time_searching)
```

LAMPIRAN C – BÉZIER CURVE

```
clc;
clear all;
clf;
close all;
warning off;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%path parameters
%Starting point
x0=5;y0=5;z0=5;

%Arrival point
% xend=120;yend=95;zend=5; %%5 gedung
xend=120;yend=95;zend=5; %%9 gedung
% xend=220;yend=200;zend=5; %%10 gedung
% xend=185;yend=185;zend=5; %%15 gedung
% xend=110;yend=120;zend=5; %%15 gedung
% xend=220;yend=220;zend=5; %%20 gedung
% xend=230;yend=230;zend=5; %25 gedung
wing=2;

%membuat titik pojok gedung
%edge of n-building and high of buildings
n=[9]; % Jumlah Gedung
n_buildings=n;
[xy,h]=build_coord(n);
gdg=['A','B','C','D','E','F','G','H','I','J','K','L','M','O','P'
,'Q','R','S','T','U','V','W','X','Y','Z'];

% number slice
nsl=5;
hmin=5;
hmax=30;

%membuat titik pojok gedung
load([num2str(n) 'building_corner'],'corner_all_building');
load([num2str(n) 'edge_coord_v2'],'xyz_edge');
load([num2str(n) 'high_coord_v2'],'zh');
load([num2str(n) 'building_coord_v2'],'area_building');
load([num2str(n) 'all_path_range3'],'all_path','all_range');

start=[x0 y0 z0];
finish=[xend yend zend];
```

```
[xy,h]=build_coord(n);
fig_building2v1(xy,h,zh,xyz_edge,n,gdg,start,finish);

num_path=size(all_path,2)/3;
step=size(all_path,1);

for k=1:num_path
    path=all_path(:,(k-1)*3+1:(k-1)*3+3);

    for i=1:step

        p=path(i,:);
        q=path(i+1,:);

        if q==[0 0 0]
            break;
        else
            [x,y,z]=makeline3d(p,q);
            plot3(x,y,z,'y','LineWidth',2);hold on;%shortest
path LOS
        end
    end
end

min_range=min(all_range);

np=1:length(all_range);
np_range=[np' all_range];
np_range=sortrows(np_range,2);

min_path_num=np_range(1,1);
min_path=all_path(:,(min_path_num-1)*3+1:(min_path_num-1)*3+3);
path=min_path;


%membangun lintasan bazier yang terbaik zero error
fig_building2v1(xy,h,zh,xyz_edge,n,gdg,start,finish);

%tanpa gambar
[path0,path1,path_param0,path_param1,baz0,baz1]=make_OK_bazier3(
path,area_building,h,zh,xyz_edge,n,gdg,start,finish,xy,corner_al
l_building,wing);
gap0=[];
dz0=[];
agl0=[];
theta_star_range=0;
bazier0_range=0;
for i=1:size(path0,2)
    pbz=path0(1:3,i);
    ths=path0(4:6,i);
    gapi=find_range(pbz,ths);
    gap0=[gap0 gapi];

    if i>1
        dzi=abs(path0(3,i)-path0(3,i-1));
        dz0=[dz0 dzi];
        pqrange=find_range(path0(4:6,i),path0(4:6,i-1));
```

```
        bzrange=find_range(path0(1:3,i),path0(1:3,i-1));
        theta_star_range=theta_star_range+pqrange;
        bazier0_range=bazier0_range+bzrange;
    end
    if i>2
        ai=path0(1:3,i-2);
        bi=path0(1:3,i-1);
        ci=path0(1:3,i);
        agli=find_angle3d2(bi,ai,ci);
        agl0=[agl0 agli];
    end
end

param_jalur2=[path_param0;max(gap0)*100;mean(gap0)*100;max(dz0)*
100;max(agl0)];

gap1=[];
dz1=[];
agl1=[];
bazier1_range=0;

for i=1:size(path1,2)
    pbz=path1(1:3,i);
    ths=path1(4:6,i);
    gapi=find_range(pbz,ths);
    gap1=[gap1 gapi];

    if i>1
        dzi=abs(path1(3,i)-path1(3,i-1));
        dz1=[dz1 dzi];
        bzrange=find_range(path1(1:3,i),path1(1:3,i-1));
        bazier1_range=bazier1_range+bzrange;
    end
    if i>2
        ai=path1(1:3,i-2);
        bi=path1(1:3,i-1);
        ci=path1(1:3,i);
        agli=find_angle3d2(bi,ai,ci);
        agl1=[agl1 agli];
    end
end

param_jalur1=[path_param1;max(gap1)*100;mean(gap1)*100;max(dz1)*
100;max(agl1)];

if param_jalur1==param_jalur2
    space=['  ';'  ';'  ';'  ';'  ';'  ';'  ';'  '];

    disp('Jalur 2 tidak ada');
    disp(' ');

disp('=====================================================');
    disp('        Perbandingan         | Jalur 1    ');
    disp('                             |(syarat 1) )');

disp('=====================================================');
    prb=['Panjang jalur (m)                          '
```

```
        'Jumlah titik jalur                          '
        'Jarak terkecil ke halangan (m)              '
        'Rata-rata jarak ke halangan (m)             '
        'Maksimum smoothing gap (cm)                 '
        'Rata smoothing gap (cm)                     '
        'Perubahan nilai ketinggian maks (cm)        '
        'Perubahan nilai sudut heading maks (deg)'];
    disp([prb space space
num2str([bazier1_range;size(baz1,2);param_jalur1])]);

else
    space=['  ';'  ';'  ';'  ';'  ';'  ';'  '];

    disp(' ');

disp('=====================================================');
disp('        Perbandngan       | Jalur 1   | Jalur 2  ');
    disp('                          |(syarat 1) | (syarat 1-2)');

disp('=====================================================');
    prb=['Panjang jalur (m)                           '
        'Jumlah titik jalur                          '
        'Jarak terkecil ke halangan (m)              '
        'Rata-rata jarak ke halangan (m)             '
        'Maksimum smoothing gap (cm)                 '
        'Rata smoothing gap (cm)                     '
        'Perubahan nilai ketinggian maks (cm)        '
        'Perubahan nilai sudut heading maks (deg)'];
disp([prb space space
num2str([bazier1_range;size(baz1,2);param_jalur1])...
        space space space
num2str([bazier0_range;size(baz0,2);param_jalur2])]);

end

disp(' ');
disp(['Jarak_Jalur_Theta_Star = ' num2str(theta_star_range) '
m']);

figure
plot3(baz0(1,:),baz0(2,:),baz0(3,:),'c','LineWidth',2);hold on;
axis equal
uav_tracking2(baz0);
```

```
function
[path0,path1,path_param0,path_param1,baz0,baz1]=make_OK_bazier3(
path,area_building,h,zh,xyz_edge,n,gdg,start,finish,xy,corner_al
l_building,wing)
step=size(path,1);

for i=1:step
    p=path(i,:);
    q=path(i+1,:);
    if q==[0 0 0]
        break;
    else
        [x,y,z]=makeline3d(p,q);
```

```
    end
end
path_ok=path(1:i,:);

%Optimasi path
[nodes2,rt1,rt2]=optim_link2(lines,path_ok,area_building);
% plot line from start to finish
for i=1:size(nodes2,1)-1
    p=nodes2(i,:);
    q=nodes2(i+1,:);
    [x,y,z]=makeline3d(p,q);
    plot3(x,y,z,'y','LineWidth',3);hold on
end

cor=corner_all_building;
%Cari nomor gedung
rand_path=[];
for i=2:size(nodes2,1)-1
    num_node=find_matrix_v2(xyz_edge(:,1:3),nodes2(i,:));
    num_gdg=xyz_edge(num_node,4);
    num_node2=find(xyz_edge(:,4)==num_gdg);
    gdg_node=xyz_edge(num_node2,:);
    gdg_cor=cor(num_node2,:);
    rem_node=rem(size(gdg_node,1),4);
    if rem_node==0

node_gdg_pos=find_matrix_v2(gdg_node(:,1:3),nodes2(i,:));
        slize_pos=ceil(node_gdg_pos/4);
        slize_node=gdg_node((slize_pos-1)*4+1:(slize_pos-
1)*4+4,:);
        slize_cor=gdg_cor((slize_pos-1)*4+1:(slize_pos-
1)*4+4,:);
    else

node_gdg_pos=find_matrix_v2(gdg_node(:,1:3),nodes2(i,:));
        if node_gdg_pos==size(gdg_node,1)
            puncak=nodes2(i,:);
        else
            slize_pos=ceil(node_gdg_pos/4);
            slize_node=gdg_node((slize_pos-1)*4+1:(slize_pos-
1)*4+4,:);
            slize_cor=gdg_cor((slize_pos-1)*4+1:(slize_pos-
1)*4+4,:);
        end
    end
    [xyz_rnd,edg_rnd]=random_node2(slize_cor,slize_node);
    xyz_node=edg_rnd;
    nodes2i=nodes2(i,:);
    rand_node_pos=find_matrix_v3(xyz_node,nodes2i);%temukan
posisi2 yang sesuai matrik mynode
    rand_path=[rand_path xyz_rnd(rand_node_pos,1:3)];
end

gdg_line=xy;
n_gdg=size(xy,1)/4;
xy_gdg=[];
for i=1:n
```

```
    xy_gdg=[xy_gdg; [xy((i-1)*4+1:(i-1)*4+4,:) i*ones(4,1)]];
end
xy_gdg=[xy_gdg(:,2) xy_gdg(:,1) xy_gdg(:,3)];

start=nodes2(1,:);
finish=nodes2(size(nodes2,1),:);
n=size(nodes2,1)-1;

%Tentukan gedung yang akan dilewati berdasarkan node yang
dipilih
nodes_gdg=nodes2(2:size(nodes2,1)-1,:);
ng=[];
for i=1:size(nodes_gdg,1)
    nodei=nodes_gdg(i,:);
    num_node=find_matrix_v2(xyz_edge(:,1:3),nodei);
    num_gdg=xyz_edge(num_node,4);
    ng=[ng;[nodei num_gdg]];
end

%====================
%membuat jalur bazier
%====================
test_error=17;
i1=0;
i2=0;
i3=0;
for ii=1:test_error
    %%memilih titik randong di sekitar node yang akan dilewati
    nodes_rand=[];
    for i=1:size(nodes2,1)-2
        %memilih titik tujuan
        nq=randi(20,1,1);
        q_rand=rand_path(:,(i-1)*3+1:(i-1)*3+3);
        q=q_rand(nq,:);
        nodes_rand=[nodes_rand;[q ng(i,4)]];
    end
    nodes_rand=[start 0
        nodes_rand
        finish 0];

    %%membuat garis LOS dari node acak yang dipilih
    %     nodes_rand
    line_rand=[];
    nn=size(nodes_rand,1)-1;
    for i=1:nn
        p=nodes_rand(i,1:3);
        q=nodes_rand(i+1,1:3);
        spc=3;
        [x,y,z]=makeline3d_v2(p,q,spc);
        line_rand=[line_rand
[x;y;z;ones(1,length(x))*nodes_rand(i,4);ones(1,length(x))*nodes
_rand(i+1,4)]];
    end
    line_rand;

    P=line_rand(1:3,:);%lintasan lurus asli
    nt=size(P,2);
```

```
    m=3;
    t=linspace(0,1,nt*m);
    xyz_bazc=Bezier(P,t);%lintasan kurva bazier
    xyz_bazc=come2finish(xyz_bazc',finish,start);
    xyz_bazc=xyz_bazc';

    %menentukan titik bazier terdekat dengan titik asli
    xyz_baz=[];
    theta_star=[];
    for i=1:nt
        baz=xyz_bazc(:,(i-1)*m+1:(i-1)*m+m);
        pi=P(:,i);
        r=[];
        for j=1:m
            rj=find_range(baz(:,j),pi);
            r=[r rj];
        end
        minr=min(r);
        bazi=find(r==minr);
        xyz_baz=[xyz_baz baz(:,bazi)];
    end

    %pengujian tahap-1
    Qmin_p=sqrt(sum((P-xyz_baz).^2,1));

    %Pencarian nilai Dj Harus diketahui dulu mana obstacle
terdekat
    n_gdg=size(nodes_rand,1);
    err1_all=[];
    line_gdg_all=[];
    for i=2:n_gdg-1
        gd=nodes_rand(i,4);
        num_gd_line=find_matrix_v3(xy_gdg(:,3),gd);
        xy_gdgi=xy_gdg(num_gd_line,1:2);
        line_gdg=[];
        for k=1:4
            if k==4
                a=[xy_gdgi(k,:) 0];
                b=[xy_gdgi(1,:) 0];
                [xg,yg,zg]=makeline3d(a,b);% x y z mendatar
            else
                a=[xy_gdgi(k,:) 0];
                b=[xy_gdgi(k+1,:) 0];
                [xg,yg,zg]=makeline3d(a,b);% x y z mendatar
            end
            line_gdg=[line_gdg [xg;yg;zg]];%koordinat garis
keliling gedung
        end

        line_gdg_all=[line_gdg_all line_gdg];
    end

    Dj=[];
    for j=1:nt
        pj=[P(1:2,j);0];
        robs_all=[];
        for k=1:size(line_gdg_all,2)
```

```
            robs=find_range(pj,line_gdg_all(:,k));
            robs_all=[robs_all robs];
        end
        Dj=[Dj min(robs_all)];%Jarak euclidiean terdekat
kegedung
    end

    Djmin_del=Dj-wing;
    error1=sum(Qmin_p>Djmin_del);
    err1_all=[err1_all;error1];

    error1_all=sum(err1_all);

    %pengujian tahap-2
    curvature=1.5;
    R=(1/curvature);
    xc3=xyz_baz(1,:);
    yc3=xyz_baz(2,:);
    zc3=xyz_baz(3,:);
    dd=[];
    for i=1:length(xc3)-2
        y0=yc3(i);
        y1=yc3(i+1);
        y2=yc3(i+2);
        dely0=y1-y0;
        dely1=y2-y1;

        x0=xc3(i);
        x1=xc3(i+1);
        delx=x1-x0;
        if delx>0
            dydx1=dely0/delx;
            dydx2=dely1/delx;
            d2ydx2=(dydx2-dydx1)/delx;
            dd=[dd;d2ydx2];
        end
    end

    error21=sum(dd>=(1/R^2));

    %curvature-2
    ddzp=[];
    dz=[];
    da=[];
    for i=1:length(xc3)-2
        z0=zc3(i);
        z1=zc3(i+1);
        z2=zc3(i+2);
        delz0=z1-z0;
        delz1=z2-z1;
        dz=[dz;delz0];
        x0=xc3(i);
        x1=xc3(i+1);
        y0=yc3(i);
        y1=yc3(i+1);
        delx=x1-x0;
        dely=y1-y0;
```

```
        if delx~=0&dely~=0
            alpha=atan(delz0/(delx^2+dely^2));
            beta=atan(dely/delx);
            da=[da;[alpha beta sin(alpha)*sin(beta)]];
        end
        t0=t(i);
        t1=t(i+1);
        delt=t1-t0;

        if delx~=0&dely~=0&delt~=0
            dydt1=dely/delt;
            dxdt1=delx/delt;
            dzdt1=delz0/delt;
            dzdt2=delz1/delt;
            d2zdxt=(dzdt2-dzdt1)/delt;
            d2zdp2=d2zdxt/(dydt1*dxdt1);
            ddzp=[ddzp;d2zdp2];
        end
    end

    error22=sum(ddzp>=(1/R^2));

    max_high_diff=max(dz);
    angle_sort=flip(sortrows(da,3));
    max_az=angle_sort(1,2);
    max_el=angle_sort(1,1);

    disp(['Pengujian ke ' num2str(ii)]);
    err_all=[error1_all error21 error22];
    if err_all==[0 0 0]
        if i3<1
            disp('Tidak ada error');
            path_zero_error=xyz_baz;
            baz0=xyz_bazc;
            path_theta_star=P;
            path0=[path_zero_error;path_theta_star];
            num_spot=nt;
            DJmin=min(Dj);
            DJave=mean(Dj);

plot3(xyz_bazc(1,:),xyz_bazc(2,:),xyz_bazc(3,:),'cyan','LineWidt
h',2);hold on;

            i3=i3+1;

        end
    end
    if err_all(1)==0
        if sum(err_all(1:2))>0
            if i2<1
                disp('Syarat 1 memenuhi, syarat 2 tidak
memenuhi');

plot3(xyz_bazc(1,:),xyz_bazc(2,:),xyz_bazc(3,:),'r','LineWidth',
2);hold on;
                i2=i2+1;
                path_zero_error1=xyz_baz;
```

```
                baz1=xyz_bazc;
                path_theta_star1=P;
                path1=[path_zero_error1;path_theta_star1];
                num_spot1=nt;
                DJmin1=min(Dj);
                DJave1=mean(Dj);
            end
        end
    end
    if i3+i2==2
        break;
    end
    disp('--------------------------------');
    disp(' ');

end
    disp(' ');
    disp('--------------------------------');
    disp('--------------------------------');

if i3==0
    disp('Tidak ada lintasan yang memenuhi semua syarat');
    disp('Ulangi lagi simulasi');
elseif i3==1
    path_zero_error0=path_zero_error';
    path_zero_error=come2finish(path_zero_error0,finish,start);

    path_param0=[DJmin; DJave];
    if i2==0
        path_param1=path_param0;
        path1=path0;
        baz1=baz0;
            disp('Semua lintasan yang memenuhi syarat 1 dan 2');
    else
        path_param1=[DJmin1; DJave1];
            disp('Terdapat lintasan yang memenuhi syarat 1 tapi
tidak memenuhi syarat 2');
    end
end
```

Halaman ini sengaja dikosongkan

RIWAYAT PENULIS

Dila Marta Putri lahir di Tapan pada tanggal 1 Maret 1995 sebagai anak kedua dari pasangan Syamsul Afri dan Yasnar. Penulis adalah alumni dari MAN/MAKN 1 Koto Baru Padang Panjang yang lulus pada tahun 2013. Pendidikan sarjana di tempuh di Program Studi S1 Teknik Elektro Konsentrasi Elektronika dan Instrumentasi Universitas Islam Negeri Sultan Syarif Kasim, lulus tahun 2017. Pada tahun Agustus 2018, penulis diterima di Program Magister di Jurusan Teknik Elektro, Institut Teknologi Sepuluh Nopember dengan bidang keahlian Teknik Sistem Pengaturan.

(dilamartaputri@yahoo.com / 085334186844)