TESIS – EE185401

DESAIN SISTEM NAVIGASI QUADCOPTER UNTUK MENGHINDARI HALANGAN DENGAN EFISIENSI ENERGI DAN JARAK SIMPANGAN MENGGUNAKAN *MACHINE LEARNING*

IRFIN SANDRA ASTI
07111750022004

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
2020

TESIS – EE185401

DESAIN SISTEM NAVIGASI QUADCOPTER UNTUK MENGHINDARI HALANGAN DENGAN EFISIENSI ENERGI DAN JARAK SIMPANGAN MENGGUNAKAN *MACHINE LEARNING*

IRFIN SANDRA ASTI
07111750022004

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
2020

LEMBAR PENGESAHAN TESIS

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

IRFIN SANDRA ASTI
NRP: 07111750022004

Tanggal Ujian: 3 Januari 2020
Periode Wisuda: Maret 2020

Disetujui oleh:
Pembimbing:

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

Penguji:

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

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

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

Kepala Departemen Teknik Elektro
Fakultas Teknologi Elektro dan Informatika Cerdas

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

Halaman ini sengaja dikosongkan

PERNYATAAN KEASLIAN TESIS

Dengan ini saya menyatakan bahwa isi keseluruhan Tesis saya dengan judul "**DESAIN SISTEM NAVIGASI QUADCOPTER UNTUK MENGHINDARI HALANGAN DENGAN EFISIENSI ENERGI DAN JARAK SIMPANGAN MENGGUNAKAN MACHINE LEARNING**" adalah benar-benar hasil karya intelektual mandiri, diselesaikan tanpa menggunakan bahan-bahan yang tidak diijinkan dan bukan merupakan karya pihak lain yang saya akui sebagai karya sendiri.

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

Surabaya, 5 Januari 2020

Irfin Sandra Asti

NRP. 07111750022004

Halaman ini sengaja dikosongkan

DESAIN SISTEM NAVIGASI QUADCOPTER UNTUK MENGHINDARI HALANGAN DENGAN EFISIENSI ENERGI DAN JARAK SIMPANGAN MENGGUNAKAN *MACHINE LEARNING*

Nama mahasiswa : Irfin Sandra Asti
NRP : 07111750022004
Pembimbing : Dr. Trihastuti Agustinah, ST., MT.

ABSTRAK

Penelitian ini membahas tentang sistem navigasi quadcopter untuk mencapai titik tujuan tanpa menabrak halangan. Sistem penghindaran halangan dibuat pada lingkungan 3D. Quadcopter menggunakan proportional-derivative untuk mencapai titik tujuan. Untuk menghindari halangan, waypoint switching diatur dengan cara menambahkan titik hindar sebelum menuju titik target. Terdapat 3 titik hindar yaitu ke atas, kiri dan kanan. Titik hindar dipilih dengan mengklasifikasikan berdasarkan jarak simpangan quadcopter dan energi yang efisien ke titik hindar menggunakan machine learning KNN (K-Nearest-Neighbour). Simulasi menunjukkan bahwa desain sistem yang dirancang dapat membuat quadcopter mencapai titik tujuan tanpa menabrak halangan yang memiliki ukuran bervariasi. Quadcopter juga berhasil menghindari halangan dengan arah hindar yang efisien.

Kata kunci: Penghindaran Halangan, Navigasi 3D, Efisiensi Energi, Quadcopter, Machine Learning, KNN.

Halaman ini sengaja dikosongkan

DESIGN OF QUADCOPTER NAVIGATION SYSTEM TO AVOID OBSTACLES WITH ENERGY EFFICIENCY AND DEVIATION DISTANCE USING MACHINE LEARNING

By : Irfin Sandra Asti
Student Identity Number : 07111750022004
Supervisor(s) : Dr. Trihastuti Agustinah, ST., MT.

ABSTRACT

This research discussed about the quadcopter navigation system to reach the destination point without collide with obstacles. Navigation system created in 3D environment. Quadcopter is controlled to reach the target point using proportional-derivative control method. Obstacle avoidance based on the classification of the deviation distance between quadcopter with obstacle and energy consumption using KNN machine learning. The simulations show that the design of the system can make the quadcopter reach the destination point without collide with obstacles that have varying sizes. The quadcopter can also avoid obstacles with efficient avoidance direction.

Key words: Obstacle Avoidance, 3D Navigation, avoidance direction, Energy Efficient, Quadcopter, Machine Learning, KNN.

Halaman ini sengaja dikosongkan

KATA PENGANTAR

Alhamdulillah, segala puji syukur pada kehadirat Allah SWT, atas segala karunia dan ridho-NYA, sehingga tesis dengan judul "**Desain Sistem Quadcopter Untuk Menghindari Halangan dengan Efisiensi Energi dan Jarak Simpangan Menggunakan Machine Learning**" 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 Akhmad Alwi dan Ibunda Kholisatul Khotimah 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 Menghianati Hasil"**. Maka dari itu, penyelesaian Tesis ini juga tak lepas dari segala usaha dan do'a penulis serta bantuan dari berbagai pihak. Semoga Tesis ini bermanfaat bagi pembaca terutama untuk penulis

Surabaya, 20 Desember 2019
Penulis

Halaman ini sengaja dikosongkan

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

Halaman ini sengaja dikosongkan

BAB 1
PENDAHULUAN

1.1 Latar Belakang

Saat ini, teknologi *Unmanned Aerial Vehicle* (UAV) telah banyak digunakan dalam keperluan pengawasan, pencarian, dan penyelamatan. Salah satu tipe UAV yang banyak digunakan tersebut adalah quadcopter. Telah banyak metode kontrol yang dikembangkan saat ini tentang *path following, waypoint tracking,* VTOL (*Vertical Take-Off and Landing*), dan lain-lain.

Seiring berkembangnya aplikasi quadcopter, diperlukan suatu mekanisme agar quadcopter lebih adaptif terhadap berbagai halangan. Situasi ini menuntut quadcopter harus dilengkapi dengan instrumen dan sistem penghindaran halangan yang *robust* untuk mencegah quadcopter menabrak halangan seperti bangunan, gunung, pohon, tiang listrik, dan lain-lain. Hal tersebut dapat meminimalkan pengawasan manusia terhadap quadcopter yang ditugaskan.

Terdapat beberapa penelitian tentang sistem kontrol quadcopter untuk bergerak dari posisi awal menuju titik target pada lingkungan yang terdapat halangan. Sistem penghindaran halangan dengan menggunakan *visibility graph* dibahas pada [1]. Metode *visibility graph* efektif dalam menghindari halangan dengan memperhitungkan posisi halangan dengan quadcopter sehingga arah hindar yang dipilih (kanan atau kiri) adalah yang terdekat dengan titik target. Kontroler PID-Fuzzy digunakan untuk mengatur gerak quadcopter. Namun, pada sistem penghindaran halangan tersebut belum memperhitungkan arah hindar ke atas. Padahal arah hindar ke atas juga dapat menghasilkan *path* yang terdekat dengan titik target.

Sistem penghindaran halangan dengan mempertimbangkan arah hindar ke atas pada UAV dapat dilakukan dengan *hybrid* 3D *path planning* [2]. *Hybrid* 3D *path planning* adalah perpaduan antara kontroler *fuzzy* yang bekerja bersama algoritma *lazy theta** untuk mencari jalur terpendek dalam menghindari halangan. Halangan diklasifikasikan menjadi 2, yaitu *semi-obstacle* untuk tinggi kurang dari 150 m dan *full-obstacle* dengan tinggi lebih dari 150 m. *Hybrid* 3D *path planning*

dapat mengatur UAV terbang pada ketinggian yang tetap mengikuti jalur yang dihasilkan algoritma *lazy theta** dan hanya mengubah ketinggian apabila terdapat *semi-obstacle* menggunakan kontroler *fuzzy*. Metode *hybrid* 3D *path planning* sederhana, cepat dan adaptif [2]. Namun, klasifikasi yang dilakukan untuk *decision making* hanya berdasarkan tinggi halangan. Apabila memperhitungkan energi yang dibutuhkan, maka keputusan arah hindar ke atas belum tentu menghasilkan jalur yang efisien karena UAV membutuhkan energi yang lebih banyak untuk menghindar ke atas daripada terbang disekitar halangan tanpa mengubah *altitude* [3]. Klasifikasi dan *decision making* dapat ditingkatkan dengan mengevaluasi jalur yang terbaik apakah dengan terbang disekitar halangan atau ke atas tergantung posisi UAV saat ini dan halangan untuk menghasilkan jalur yang efisien.

Klasifikasi halangan berdasarkan dimensi halangan telah dibahas pada [4] untuk navigasi *mobile robot*. Klasifikasi yang dilakukan menggunakan metode *machine learning.* Data fitur-fitur untuk klasifikasi adalah data pengukuran lebar, tinggi dan kedalaman yang dihasilkan oleh sensor kamera. Dari fitur-fitur tersebut diklasifikasikan menjadi 3 kelas, yaitu batuan kecil, batuan besar dan lereng. Jadi, navigasi yang dihasilkan bisa berbeda-beda tergantung dari hasil klasifikasi. Metode *machine learning* cocok untuk klasifikasi dengan fitur lebih dari satu. Hal ini dapat mempermudah dalam menentukan keputusan saat halangan terdeteksi.

Berdasarkan berbagai penelitian diatas, ide yang dihasilkan untuk Tesis ini adalah desain sistem navigasi quadcopter dalam menghindari halangan dengan arah hindar yang diklasifikasikan menjadi 3, yaitu kanan, kiri dan atas. Keputusan dalam menentukan arah hindar dihasilkan dari klasifikasi menggunakan *machine learning*, berdasarkan efisiensi jarak simpangan quadcopter terhadap halangan dan energi.

1.2 Rumusan Masalah

Rumusan masalah dari penelitian ini adalah bagaimana merancang sistem navigasi pada quadcopter untuk menghindari halangan, dengan efisiensi jarak simpangan antara quadcopter terhadap halangan dan energi yang dibutuhkan dalam menentukan arah hindar (kanan, kiri dan atas).

1.3 Tujuan

Tujuan dari penelitian ini adalah menghasilkan desain sistem navigasi quadcopter untuk menghindari halangan pada lingkungan 3D menggunakan metode *machine learning* dalam menentukan keputusan arah hindar (kanan, kiri dan atas), berdasarkan efisiensi jarak simpangan antara quadcopter dengan halangan dan energi yang dibutuhkan.

1.4 Batasan Masalah

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

1. Quadcopter yang digunakan adalah tipe Quanser-Qdrone.
2. Tidak memperhitungkan *noise* pengukuran.
3. Kecepatan quadcopter konstan.
4. Halangan dinamis dengan kecepatan tertentu.
5. Pemilihan jalur berdasarkan optimisasi lokal.

1.5 Kontribusi

Kontribusi penelitian ini adalah desain sistem navigasi quadcopter dalam menghindari halangan pada lingkungan 3D dengan dengan arah hindar yang diklasifikasikan menjadi 3 yaitu kanan, kiri dan atas, berdasarkan efisiensi jarak simpangan (quadcopter dengan halangan) dan energi. Keputusan dalam menentukan arah hindar dihasilkan dari klasifikasi menggunakan *machine learning.*

Halaman ini sengaja dikosongkan

BAB 2
KAJIAN PUSTAKA

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

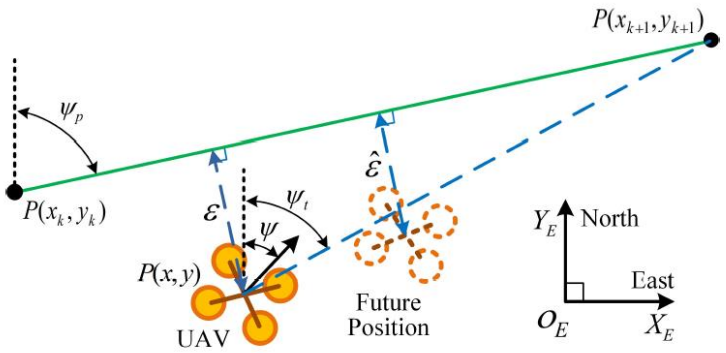
2.1 Kajian Penelitian Terkait

2.1.1 *Path Following Control of Unmanned Quadrotor Helicopter with Obstacle Avoidance Capability* [1]

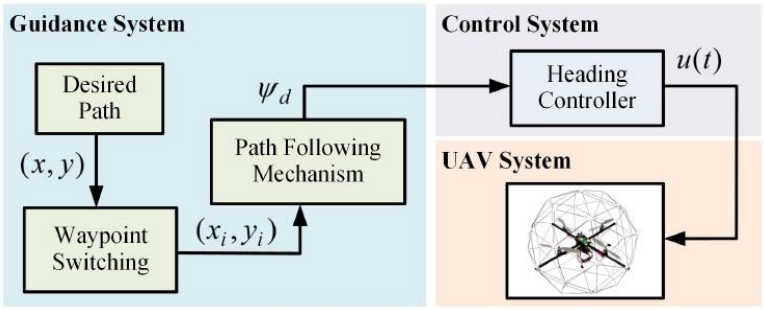
Paper ini membahas tentang kontrol *path following* dikombinasikan dengan sistem penghindaran halangan pada *Unmanned Quadrotor Helicopter* (UQH). Kontrol *path following* yang digunakan adalah *cross track error* berbasis prediksi eror. Berikut adalah tahapan atau proses metode *cross track error.*

1. Pertama, menentukan *desired path* yang dihasilkan garis lurus antara titik *start* $P(x_k, y_k)$ dan akhir $P(x_{k+1}, y_{k+1})$.
2. Selanjutnya menghitung nilai *cross-track error* atau jarak ortogonal menuju tangensial *desired path* referensi (sesuai Gambar 2.1).

$$\varepsilon = \frac{(y_{k+1} - y_k)x + (x_k - x_{k+1})y + y_k x_{k+1} - x_k y_{k+1}}{\sqrt{(y_{k+1} - y_k)^2 + (x_k - x_{k+1})^2}} \qquad (2.1)$$



Gambar 2.1 Geometri Path Following [1]



Gambar 2.2 Ilustrasi Skema *Path Following* [1]

3. Terakhir, menentukan nilai prediksi c*ross-track error* dengan persamaan berikut:

$$\hat{\varepsilon} = \varepsilon + U_f \times T_s \times \sin(\Delta\psi) \quad (2.2)$$

dimana:

$\hat{\varepsilon}$: *Prediksi Cross-Track error*

ε : *Cross-track error*

U_f : Kecepatan forward quadcopter

T_s : *Time-sampling*, untuk menentukan $\hat{\varepsilon}$ pada periode selanjutnya

$\Delta\psi$: $(\psi_{ref} - \psi)$

Halangan yang digunakan pada paper ini bersifat statis dan tidak diketahui sebelumnya. Berdasarkan Gambar 2.2, sistem *waypoint switching* mengubah *heading* UQH yang sebelumnya mengikuti *desired path* menjadi arah hindar saat halangan terdeteksi. Dengan menggunakan nilai prediksi $\hat{\varepsilon}$, *heading* ditentukan dengan persamaan (2.3) saat halangan terdeteksi.

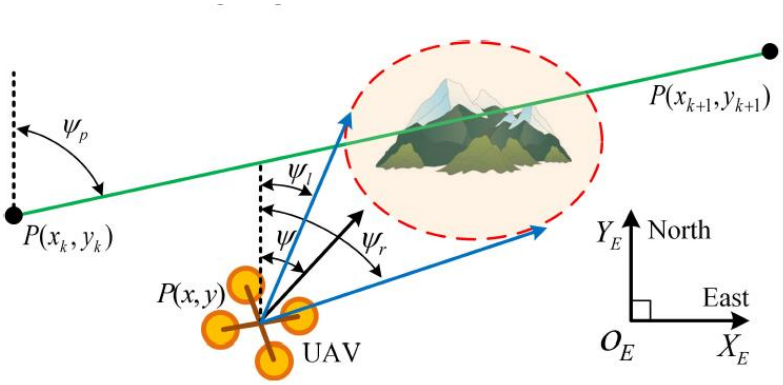
$$\psi_d = \psi_p + K_c\hat{\varepsilon} \quad (2.3)$$

dimana:

ψ_d : *Heading* yang diinginkan

ψ_p : *Heading* saat ini

K_c : Koefisien tuning dari *cross track error* dan *desired path*



Gambar 2.3 Geometri *Obstacle Avoidance* [1]

Pada Gambar 2.3, terdapat 2 *heading* untuk menghindari halangan ke kiri dan ke kanan. Sudut *heading* yang dipilih berdasarkan persamaan (2.4).

$$\psi_d = \begin{cases} \psi_l, & if\ \delta_l \leq \delta_r \\ \psi_r, & if \delta_l > \delta_r \end{cases} \tag{2.4}$$

dengan

$$\delta_l = |\psi_l - \psi| \tag{2.5}$$

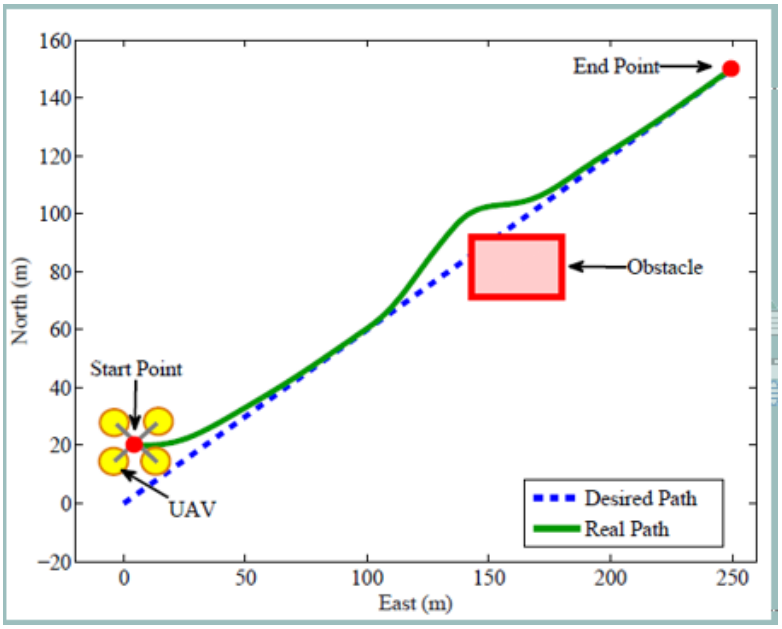
$$\delta_r = |\psi_r - \psi| \tag{2.6}$$

dimana:

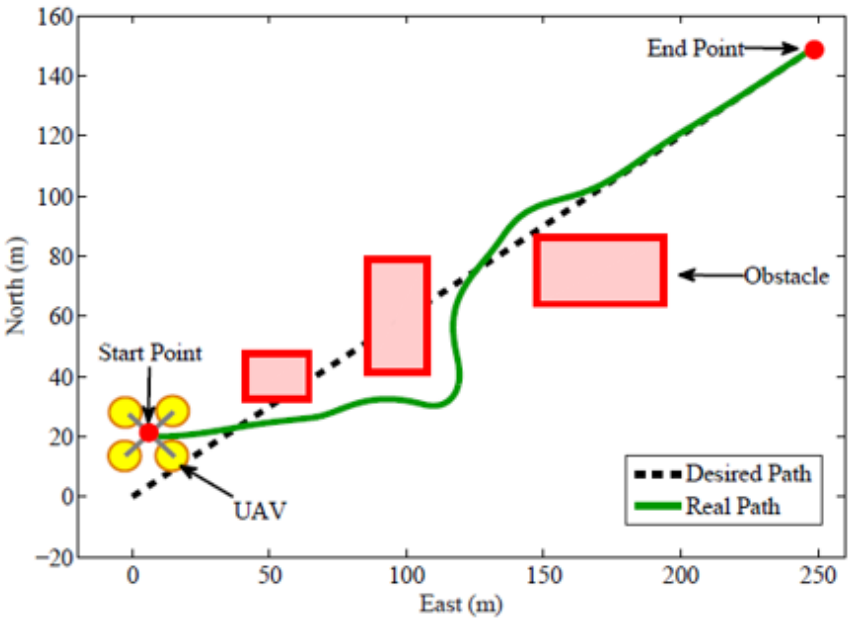
δ_l : Nilai simpangan kiri

δ_r : Nilai simpangan kanan

Selain mekanisme *path following* dan sistem penghindaran halangan, UQH juga memerlukan sistem kontrol untuk mengatur pergerakan sesuai dengan yang diinginkan. Kontroler PID-Fuzzy digunakan pada paper ini untuk mengatur *attitude* dan *altitude* UQH. Keseluruhan hasil simulasi ditunjukkan pada Gambar 2.4 dan 2.5.



Gambar 2.4 Simulasi Skenario 1 [1]



Gambar 2.5 Simulasi Skenario 2 [1]

Dari pembahasan *paper* ini, metode yang digunakan untuk menghindari halangan sangat efektif karena memperhitungkan posisi antara halangan dan quadcopter sehingga sudut *heading* dalam menghindari juga dipilih yang terdekat dengan titik target. Pada *paper* ini juga memperhitungkan dinamika dari quadcopter dan memberi batas bahaya atau jarak antara halangan dan quadcopter.

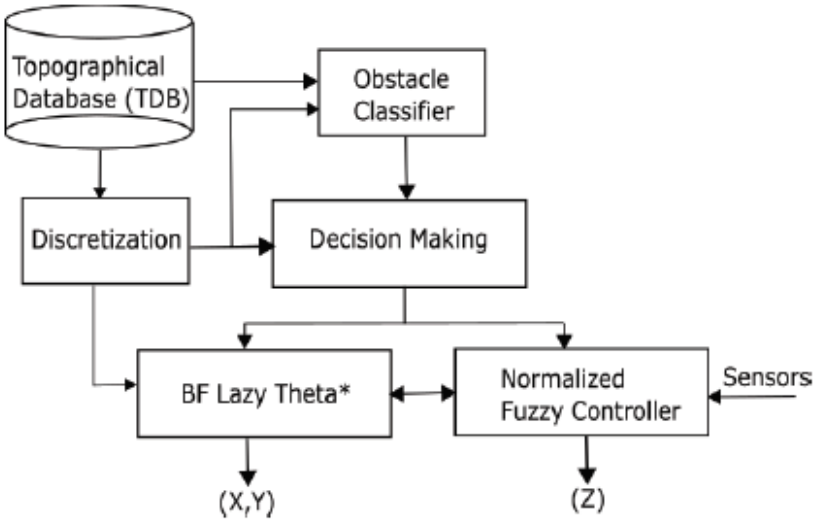
Namun, belum memperhitungkan pergerakan ke atas untuk menghindari halangan. Menghindar ke atas juga dapat menghasilkan arah hindar yang terdekat dengan titik target.

2.1.2 *A Hybrid 3D Path Planning Method for UAVs* [2]

Paper ini menjelaskan tentang metode *path planning* UAV untuk menghindari halangan pada lingkungan 3D. Metode yang digunakan adalah perpaduan antara algoritma 2D *offline near-optimal* dan kontroler *online fuzzy* untuk mencari rute yang terpendek. Skema sistem *path planning* ditunjukkan pada Gambar 2.6. Halangan diklasifikasikan menjadi 2 berdasarkan ketinggiannya [2], yaitu:

- *Semi-obstacle* jika tinggi *obstacle* $<$ 150 m.
- *Full-obstacle* jika tinggi obstacle $\geq$150 m.

Saat yang terdeteksi adalah *full-obstacle*, algoritma yang digunakan adalah *Backward Forward lazy theta** (BF *lazy theta*), yaitu salah satu algoritma *near-optimal* 2D *off-line* path *planning lazy theta**. Algoritma *lazy theta** ditunjukkan pada Gambar 2.7.



Gambar 2.6 Skema *Sistem Planning*

Algorithm 1 Lazy Theta*

```
1:  procedure SETVERTEX(s)
2:      if NOT lineofsight(parent(s), s) then                    ▷ Path 1
3:          parent(s) :=
    argmin_{s' ∈ nghbr_vis(s) ∩ closed(g(s')+c(s',s))};
4:          g(s) := min_{s' ∈ nghbr_vis(s) ∩ closed(g(s')+c(s',s))};
5:      end if
6:  end procedure
7:  procedure COMPUTECOSTS(s, s')
8:      if g(parent(s)) + c(parent(s), s') < g(s') then           ▷ Path 2
9:          parent(s') := parent(s);
10:         g(s') = g(parent(s) + c(parent(s), s');
11:     end if
12: end procedure
```

Gambar 2.7 Algoritma *Lazy Theta** [2]

BF *lazy theta** adalah evaluasi *path* yang dihasilan dari *forward* (start ke *target*) atau *backward* (target ke *start*). Persamaan BF *lazy theta**:

$$sp(s,d) = \begin{cases} fp(s,d) & if\ len\ (fp(s,d)) \le len(bp(d,s)) \\ bp(d,s) & if\ len\ (bp(s,d)) < len(fp(d,s)) \end{cases}$$
$$len\ (fp(p,s) = \sum_{i=0}^{N-2} d_e(p(i), p(i+1)) \qquad (2.7)$$
$$p(i) = p(N-i)\ for\ i = 0 \ldots . N$$

dimana:

$sp(s,d)$: *Path* yang dihasilkan BF *lazy theta*

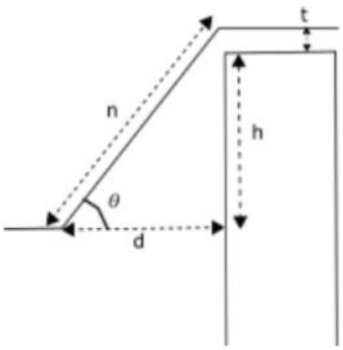
$fp(s,d)$: *Path* yang dihasilkan dari *forward*

$bp(d,s)$: *Path* yang dihasilkan dari *backward*

len : Panjang *path*

$p(i)$: Posisi *vertex*

d_e : Panjang jarak antar *vertex*



Gambar 2.8 Path saat *Semi-Obstacle* [2]

Saat halangan yang dideteksi adalah *semi-obstacle*, maka *decision making* akan mengirimkan perintah pada kontroler fuzzy yang dapat menggerakkan UAV untuk menghindari halangan dengan arah ke atas. Kontroler fuzzy menghasilkan *path* yang diilustrasikan pada Gambar 2.8. *Path* dihasilkan terbentuk dengan menghitung tinggi halangan yang dibaca sensor UAV, yaitu:

$$h = |h_t - c_a| \tag{2.8}$$

dimana:

h : Tinggi halangan yang dibaca sensor UAV
h_t : Tinggi halangan dari *database*
c_a : *Altitude* UAV

dan sudut yang terbentuk antara posisi UAV terhadap titik hindar, θ, berikut:

$$\theta = tan^{-1}\left(\frac{h+t}{d}\right) \tag{2.9}$$

dimana:

t : Jarak aman.
d : Jarak deteksi.

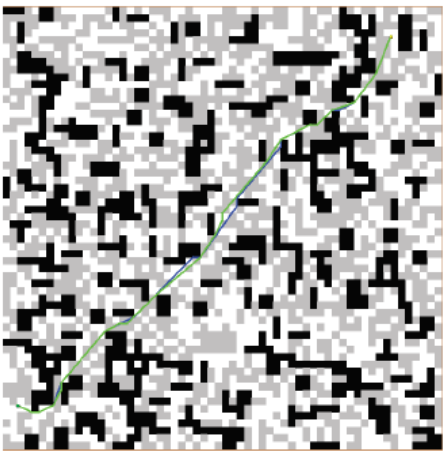
Sehingga terbentuk *path* sepanjang *n* untuk menghindar ke atas dengan persamaan:

$$n = \tan\theta \,.\, d \tag{2.10}$$

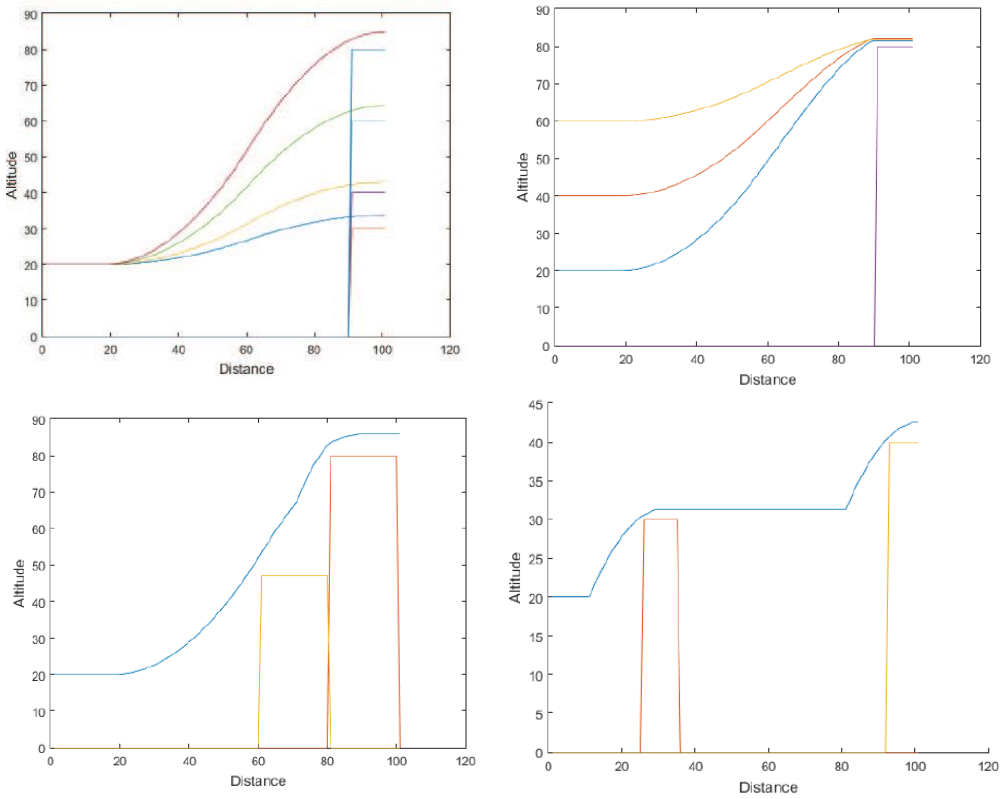
Input kontroler Fuzzy adalah variabel lingustik dari jarak dari UAV terhadap halangan dan tinggi halangan (*close, far)*. Sedangkan *output* adalah kontrol ketinggian (*keep, increase*) pada UAV. Semua *input* dan *output* direpresentasikan menggunakan *membership function* yang berbentuk *trapezoid. Rule base* kontroler fuzzy adalah sebagai berikut:

If (DistanceFromObject is Close) and (HighestPointObject is Close) then (Altitude is Increase)
If (DistanceFromObject is Close) and (HighestPointObject is Far) then (Altitude is Increase)
If (DistanceFromObject is Far) and (HighestPointObject is Far) then (Altitude is Keep)
If (DistanceFromObject is Far) and (HighestPointObject is Close) then (Altitude is Keep)
If (HighestPointObject is Close) then (Altitude is Increase)
If (HighestPointObject is Far) then (Altitude is Increase)

Simulasi dilakukan dengan menguji algoritma BF *lazy theta** bersama dengan kontroler fuzzy pada lingkungan 3D. Terdapat beberapa kubus dengan tinggi yang berbeda. Hasil simulasi terdapat pada gambar 2.9 dan 2.10.



Gambar 2.9 Hasil simulasi BF Lazy Theta* [2]



Gambar 2.10 Hasil Simulasi Kontroler Fuzzy [2]

Kontroler fuzzy yang bekerja bersama dengan algoritma BF *lazy theta** dapat menghasilkan jalur terpendek untuk menghindari halangan dengan mengubah ketinggian UAV secara *real-time* menggunakan data dari sensor dan *database* topografi.

Namun, klasifikasi untuk menghindari halangan hanya dengan ketentuan tinggi halangan. Untuk menghasilkan rute terpendek, tidak hanya dilihat dari tinggi halangan saja, tetapi juga perlu memperhatikan posisi UAV terhadap halangan. *Decision making* yang dibuat pada [2] juga belum memperhitungkan energi yang dibutuhkan. karena meskipun menghindar ke atas (z) menghasilkan rute terpedek, belum tentu akan membutuhkan energi yang lebih kecil dibandingkan dengan menghindar di sekitar halangan (x, y). Ide yang dapat diambil dari paper ini yaitu klasifikasi halangan dilakukan dengan ketentuan variasi tinggi, lebar dan posisi UAV terhadap halangan. Selain itu juga mempertimbangkan energi yang dibutuhkan dalam memilih keputusan arah hindar.

2.1.3 *Efficient Energy Flight Path Planning Algoritm Using 3-D Visibility Roadmap for Small Unmanned Aerial Vehicle* [3]

Metode *visibility roadmap* digunakan pada [3] untuk *path planning* pada UAV. *Path planning* juga memperhitungkan konsumsi energi yang dibutuhkan dalam pemilihan jalur. Halangan yang digunakan dalam penelitian ini adalah halangan statis. *Roadmap* pada Gambar 2.11 terdiri dari titik hindar yang mungkin dapat dipilih untuk menghindari halangan dan membentuk *waypoint* jika digabungkan dengan halangan lainnya. Titik hindar tersebut juga digunakan untuk mengecek *line of sight* terhadap titik selanjutnya sampai titik target dengan tujuan untuk menghitung energi yang dibutuhkan.

Nilai total energi yang dibutuhkan dihitung dengan penjumlahan dari energi potensial, energi kinetik dan energi belok. Setiap 2 *waypoint* terhubung dapat dihitung total energi menggunakan:

$$\Delta E_{i,j} = \Delta E_p + \Delta E_k + \Delta E_{turn} \quad (2.11)$$

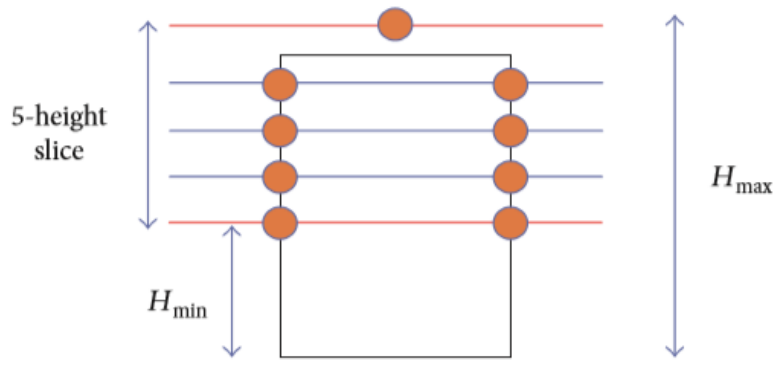
dimana:

$\Delta E_{i,j}$: Total energi dari *waypoint* i sampai j

ΔE_p : Total energi potensial

ΔE_k : Total energi kinetik

ΔE_{turn} : Total energi belok



Gambar 2.11 *Roadmap* pada *Single Obstacle* [3]

Energi potensial ΔE_p akan bertambah bila terdapat perubahan *altitude* dengan persamaan (2.12).

$$\Delta E_p = \max(W\Delta h, 0) = \max\left(\text{mg}\left(h_i - h_j\right), 0\right) \tag{2.12}$$

dimana:

h_i : Tinggi *waypoint i*

h_j : Tinggi *waypoint j*

m : Massa UAV

g : Gravitasi

Energi kinetik dihitung dengan:

$$\Delta E_k = AV^2 + \frac{B}{V^2} \tag{2.13}$$

$$A = \frac{\rho f}{2mg}$$

$$B = \frac{2mg}{\rho b^2 \pi e} \tag{2.14}$$

dimana V adalah kecepatan selama terbang sedangkan A dan B adalah parameter dengan berat jenis udara. Sedangkan energi untuk belok adalah sebagai berikut:

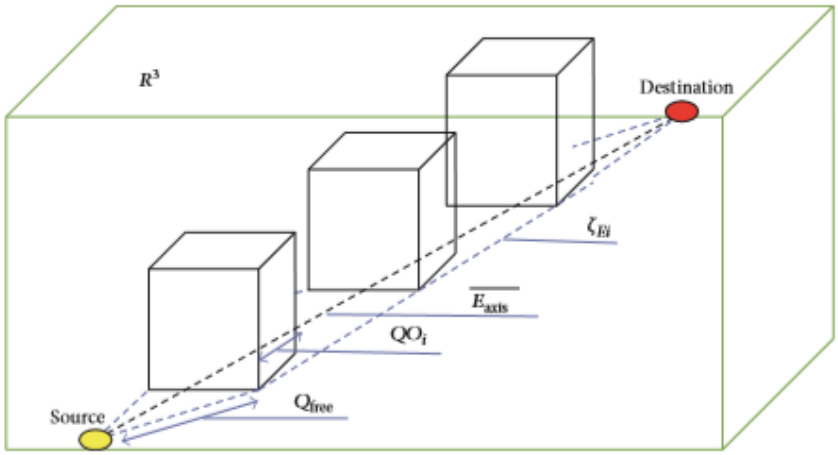
$$\Delta E_{turn} = m\, \Delta E_k\, sin\sigma |V|^2 \tag{2.15}$$

dimana:

σ : Sudut maksimal belok pada UAV

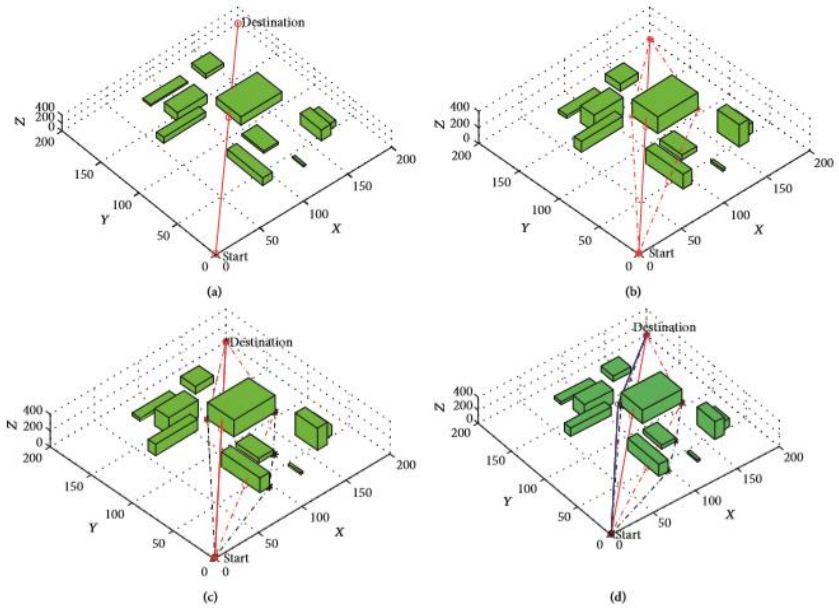
m : Massa UAV

Dalam menghitung total energi yang terbentuk pada setiap *path* memerlukan waktu komputasi yang lama bila terdapat banyak halangan. Diperlukan batasan energi, yaitu hanya membandingkan 3 *path* yang terdekat dengan garis lurus antara titik *start* dan target seperti Gambar 2.12.

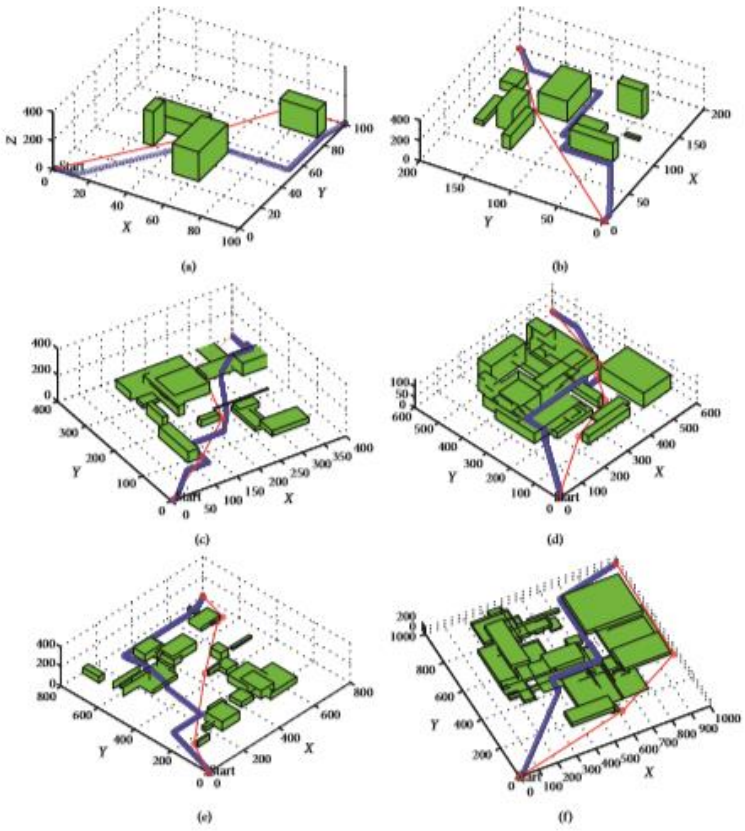


Gambar 2.12 Ilustrasi Batasan Energi [3]

Path yang terpilih adalah *path* optimal. Untuk menghitung *path* optimal yaitu dengan meminimalkan jumlah nilai jarak dan energi pada setiap *path.* Misalnya pada Gambar 2.13, *path* yang terpilih adalah *path* warna biru.



Gambar 2.13 Penjelasan Space Energi [3]



Gambar 2.14 Hasil Simulasi [3]

Hasil simulasi dapat dilihat pada Gambar 2.14. Simulasi dilakukan dengan membandingkan dengan metode *grid.* Perbandingan kedua metode dapat dilihat pada Tabel 2.1.

Penentuan *path* yang efisien pada *paper* ini tidak hanya memperhitungkan jarak terpendek saja, melainkan juga mempertimbangkan energi yang dibutuhkan dalam pemilihan *path.* Dari hasil penelitian ini, dapat dilihat bahwa jarak yang dekat belum tentu menghasilkan energi yang minimal.

Tabel 2.1 Perbandingan Metode Roadmap dan Grid [3]

Obsacle	***Proposed visibility Roadmap***			***Grid Method***		
	Energy	***Distance***	***Heading***	***Energy***	***Distance***	***Heading***
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

Hal ini terjadi karena UAV akan memerlukan energi yang lebih banyak saat menambah ketinggian (bergerak ke atas) daripada menghindar disekitar halangan tanpa merubah *altitude* pada UAV. Arah hindar yang efisien juga harus memperhitungkan energi yang dibutuhkan dalam *decision making.*

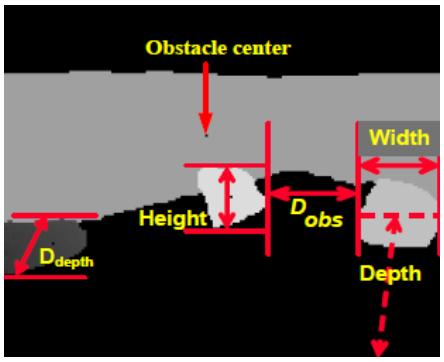
2.1.4 *Obstacle Classification and 3D Measurement in Unstructured Environments Based on ToF Cameras* [4]

Pada *paper* ini dibahas tentang deteksi dan klasifikasi halangan berdasarkan sensor camera ToF (*Time-of-flight*) yang digunakan untuk navigasi pada *mobile robot*. Kamera ToF digunakan untuk memberikan informasi tentang ukuran dari halangan 3D yaitu lebar, tinggi dan kedalaman.

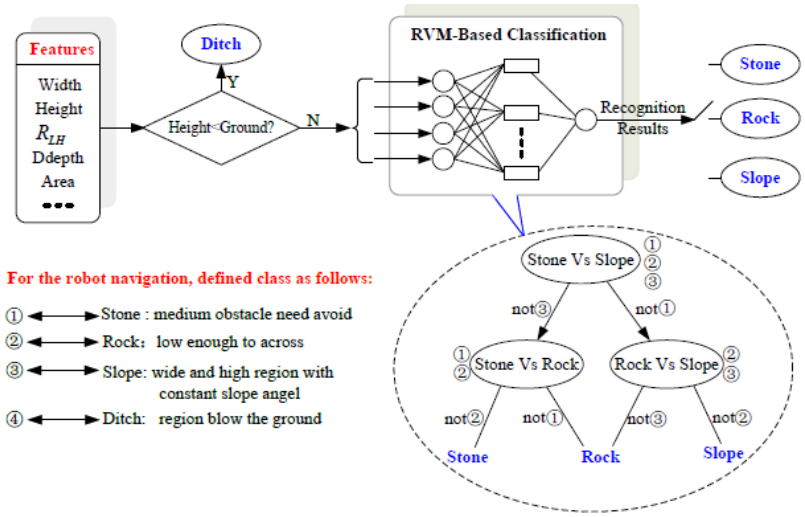
Saat ada halangan yang menghalangi jalan, secara otomatis akan mencari jalan lain yang bebas dari halangan. Hasil pengukuran oleh kamera yang berupa lebar, tinggi dan kedalaman akan menjadi fitur dalam klasifikasi.

Sedangkan kelas dalam klasifikasi ada 4, yaitu:

1. Batuan: Hambatan kecil dapat dilalui oleh robot tanpa menghindarinya.
2. Batu: Batu sedang seperti batu, pilar dan pohon harus dihindari oleh robot.
3. Lereng: rintangan besar, seperti kemiringan, dinding atau megalit. Robot tidak perlu menghindari, melainkan dilewati dengan menambahkan kecepatan.
4. Ditch: rintangan di bawah bidang tanah, seperti parit atau jalur air kecil yang harus dihindari oleh robot.



Gambar 2.15 Fitur Geometri Halangan



Gambar 2.16 Skema Klasifikasi Halangan Berdasarkan RVM [4]

Berdasarkan data dari fitur-fitur yang ada, selanjutnya diklasifikasikan menggunakan metode RVM (*relevance vector machine*). Skema rinci ditampilkan pada Gambar 2.16. Pada simulasi, sebanyak 500 skenario dengan halangan yang berbeda sebagai data *training*. Setiap halangan dan keterangan fiturmya dideteksi pada setiap skenario. Contoh hasil tabel fitur dan kelasnya dengan metode RVM dapat dilihat pada Tabel 2.2. Data *training* yang telah ditentukan juga diuji akurasinya dengan data *testing* pada Tabel 2.3 yang menunjukkan akurasi diatas 90%.

Tabel 2.2 Contoh *Training Data* [4]

Lebar (cm)	Tinggi (cm)	Kedalaman (cm)	Kelas
40	41	230	*Stone*
39	41	273	*Stone*
18	13	297	*Rock*
46	25	376	*Stone*
52	19	350	*Rock*
29	18	253	*Rock*
19	25	215	*Rock*
105	57	376	*Stone*
31	379	435	*Stone*
418	132	574	*Slope*

Tabel 2.3 Jumlah *Training* Set dan Testing Set dalam 500 Skenario [4]

Kelas Halangan	**Jumlah *Training***	**Jumlah *Testing***	**Akurasi**
Rock	238	163	92.7 %
Stone	282	201	90.4 %
Slope	167	114	95.5 %
Total	687	478	92.86 %

Pada *paper* ini berhasil melakukan pengukuran pada halangan 3D (lebar, tinggi dan kedalaman) dan mengklasifikasikannya menjadi 3 kelas menggunakan salah satu metode machine learning yaitu RVM.

Pada Tesis ini, klasifikasi dilakukan untuk menghasilkan arah hindar yang efisien (kanan, kiri atau atas). Fitur-fitur yang digunakan adalah jarak simpangan antara quadcopter dengan halangan dan energi yang dibutuhkan pada setiap arah hindar.

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 sistem navigasi quadcopter dalam menghindari halangam, konsep quadcopter secara umum dan konsep gerak dari quadcopter. Bagian selanjutnya membahas tentang teori metode klasifikasi machine learning KNN (*K-nearest-Neighbour*) dan metode kontrol gerak quadcopter.

2.2.1 Sistem Navigasi dalam Menghindari Halangan pada UAV

UAV adalah kendaraan udara yang mampu bergerak secara otomatis tanpa adanya kontrol langsung dari manusia. Salah satu jenis UAV yang sering digunakan quadcopter. Permasalahan yang mendasar pada quadcopter adalah tentang sistem navigasi, dimana navigasi adalah suatu perpindahan dari suatu titik *waypoint* ke titik yang lain yang meliputi koordinasi dari perencanaan, penginderaan dan pengendalian [5]. Tujuan dari navigasi disini agar quadcopter sampai pada titik target yang ditentukan tanpa menabrak halangan.

Selama menuju titik target dan quadcopter mendeteksi halangan, maka quadcopter perlu diberikan perintah untuk menghindarinya. Dalam menghindari

halangan, quadcopter harus berpindah ke titik hindar terlebih dahulu sebelum menuju titik target. Titik hindar secara otomatis akan ditentukan oleh suatu algoritma yang dimasukkan pada sistem navigasi.

2.2.2 Konsep Dasar Quadcopter

Quadcopter merupakan salah satu jenis kendaraan udara tanpa awak (*unmanned aerial vehicle*), quadcopter memiliki model mekanik yang terdiri dari empat rotor yang dipasang pada sumbu silang simetris. Secara umum quadrotor memiliki pengaturan motor depan dan motor belakang yang berputar searah jarum jam, sedangkan motor kiri dan motor kanan berputar berlawanan arah jarum jam Dengan mengatur kecepatan dari setiap motor, maka pergerakan quadrotor dapat berubah.

Quadcopter memiliki 6 derajat kebebasan (*degree of freedom*), enam variabel tersebut digunakan untuk mengekspresikan posisi dan orientasi dalam bidang ($x, y, z, \phi, \theta,$ dan ψ). Jarak pusat massa quadcopter sepanjang sumbu x, y dan z dari frame bumi dinotasikan oleh x, y dan z. Sudut Euler ($\phi, \theta,$ dan ψ) merepresentasikan orientasi dari quadrotor. Sudut *roll* yang dinotasikan oleh ϕ merupakan sudut disekitar sumbu x. Sudut *pitch* dinotasikan oleh θ yang merupakan sudut pada sumbu y, dan sudut *yaw* dinotasikan oleh ψ yang merupakan sudut pada sumbu z.

Gambar 2.17 merupakan ilustrasi quadrotor dengan acuan frame bumi dan frame *body*. Sudut *roll* dan *pitch* biasanya disebut dengan *attitude* untuk quadrotor, sedangkan sudut *yaw* ditunjukkan sebagai *heading* quadcopter. Untuk jarak quadcopter dari tanah dikenal sebagai *altitude* atau ketinggian, sedangkan posisi quadcopter dalam bidang X, Y biasa disebut dengan istilah posisi quadcopter.

Quadcopter dapat terbang dengan syarat gaya angkat pada quadcopter lebih besar daripada gaya gravitasi. Pada kondisi titik berat yang seimbang dan karakteristik motor yang sama, kondisi melayang (*hover*) tercapai saat semua motor memiliki kecepatan yang besarnya sama. Untuk mencapai posisi tertentu, quadcopter memiliki 4 pergerakan dasar yang digunakan berdasarkan pengaturan

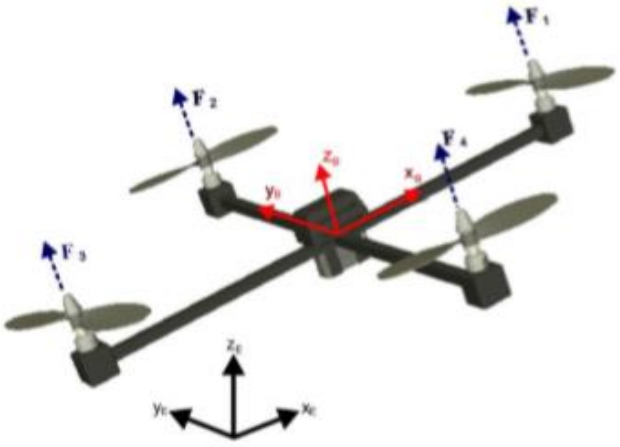
kecepatan dan arah putaran keempat motor, yaitu gerak *thrust* (U1), *roll* (U2), *pitch* (U3) dan *yaw* (U4).

Gerakan *thrust* yaitu gerakan menaikan dan menurunkan posisi quadcopter pada sumbu vertikal. Pergerakan ini terjadi ketika kecepatan dari keempat motor dipercepat atau diperlambat dengan nilai penambahan yang sama. Semakin besar penambahan, maka semakin cepat pergerakan quadcopter ke atas. Jika pergerakan thrust diatur pada nilai tertentu, maka quadcopter dapat melakukan gerak hover.

Gerakan *roll* yaitu gerakan perputaran sudut quadcopter pada sumbu x. Pergerakan ini terjadi ketika kecepatan motor kiri, dipercepat atau diperlambat, dan kecepatan motor kanan diubah sebaliknya.

Gerakan *pitch* yaitu gerakan perputaran sudut quadcopter pada sumbu y. Pergerakan ini terjadi ketika kecepatan motor depan, dipercepat atau diperlambat, dan kecepatan motor belakang diubah sebaliknya. Semakin besar percepatan sudut *pitch* maka semakin cepat juga quadrotor berputar.

Gerakan *yaw* merupakan gerakan perputaran sudut quadrotor pada sumbu z. Pergerakan ini terjadi dengan cara mempercepat atau memperlambat motor depan dan belakang sedangkan motor kiri dan kanan diubah sebaliknya.



Gambar 2.17 Quadcopter UAV dan Gaya Angkat Pada Setiap Rotor [6]

2.2.2.1 Kinematika Quadcopter [6]

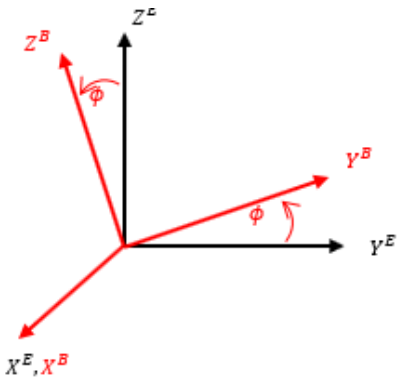
Kinematika merupakan cabang ilmu mekanika yang membahas gerak sebuah benda dengan mengabaikan gaya penyebab gerakan. Pada quadcopter analisa gerak kinematika meliputi gerak *roll*, *pitch*, dan *yaw* yang mengacu pada koordinat *B-frame*, sehingga untuk menentukan posisi linear terhadap bumi diperlukan sebuah matriks transformasi dari koordinat *B-frame* ke koodinat *E-frame*. Sehingga posisi linear maupun posisi *angular* dapat didefinisikan sebagai berikut:

$$\xi^E = [X \quad Y \quad Z]^T \qquad (2.15)$$
$$\Theta^E = [\varphi \quad \theta \quad \psi]^T$$

Sebuah matriks transformasi harus digunakan untuk mentransformasi vektor state dari *E-frame* ke *B-frame*. Matriks tersebut didapat berdasarkan kaidah dari pertama rotasi terhadap sumbu X, kemudian rotasi terhadap sumbu Y dan terakhir rotasi terhadap sumbu Z.

- Gerak *Roll*

Gerak *roll* adalah gerak quadcopter yang berotasi pada sumbu *x* sebesar φ atau dilambangkan $R(x, \varphi)$. Gerak tersebut dapat dilihat pada Gambar 2.18.



Gambar 2.18 Diagram Cartesius Rotasi terhadap Sumbu *X* [6]

Berdasarkan gerak *roll* pada quadcopter didapatkan persamaan kinematika seperti pada:

$$\begin{aligned} X^E &= X^B \cos 0^o + Y^B \cos 90^o + Z^B \cos 90^o \\ Y^E &= X^B \cos 90^o + Y^B \cos\varphi + Z^B \cos(90^o + \varphi) \\ Z^E &= X^B \cos 90^o + Y^B \cos(90^o - \varphi) + Z^B \cos\varphi \end{aligned} \tag{2.16}$$

Persamaan (2.16) dapat diubah dalam bentuk matriks seperti pada persamaan (2.17) dan (2.18)

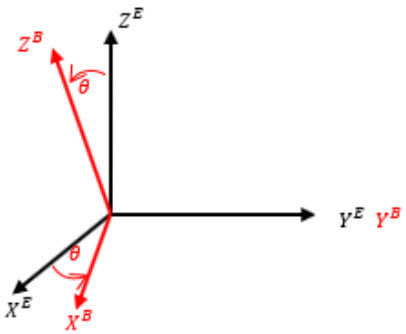
$$\begin{bmatrix} X^E \\ Y^E \\ Z^E \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\varphi & -\sin\varphi \\ 0 & \sin\varphi & \cos\varphi \end{bmatrix} \begin{bmatrix} X^B \\ Y^B \\ Z^B \end{bmatrix} \tag{2.17}$$

$$R^{EB}(x,\varphi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\varphi & -\sin\varphi \\ 0 & \sin\varphi & \cos\varphi \end{bmatrix} \tag{2.18}$$

- Gerak *Pitch*

Gerak *pitch* adalah gerak quadcopter yang berotasi pada sumbu *y* sebesar θ atau dilambangkan $R(y,\theta)$. Gerak tersebut dapat dilihat pada Gambar 2.19. Berdasarkan gerak *pitch* pada quadcopter didapatkan persamaan kinematika seperti pada persamaan (2.19).

$$\begin{aligned} X^E &= X^B \cos\theta + Y^B \cos 90^o + Z^B \cos(90^o - \theta) \\ Y^E &= X^B \cos 90^o + Y^B \cos 0^o + Z^B \cos 90^o \\ Z^E &= X^B \cos(90^o + \theta) + Y^B \cos 90^o + Z^B \cos\theta \end{aligned} \tag{2.19}$$



Gambar 2.19 Diagram Cartesius Rotasi terhadap Sumbu *Y* [6]

Persamaan (2.19) dapat diubah dalam bentuk matriks seperti persamaan (2.20) dan (2.21).

$$\begin{bmatrix} X^E \\ Y^E \\ Z^E \end{bmatrix} = \begin{bmatrix} cos\,\theta & 0 & sin\,\theta \\ 0 & 1 & 0 \\ -sin\,\theta & 0 & cos\,\theta \end{bmatrix} \begin{bmatrix} X^B \\ Y^B \\ Z^B \end{bmatrix} \tag{2.20}$$

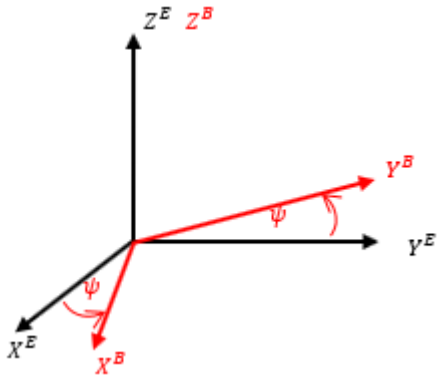
$$R^{EB}(y,\theta) = \begin{bmatrix} cos\,\theta & 0 & sin\,\theta \\ 0 & 1 & 0 \\ -sin\,\theta & 0 & cos\,\theta \end{bmatrix} \tag{2.21}$$

- Gerak *Yaw*

Gerak *yaw* adalah gerak quadcopter yang berotasi pada sumbu *z* sebesar ψ atau dilambangkan $R(z,\psi)$. Gerak tersebut dapat dilihat pada Gambar 2.20. Berdasarkan gerak *yaw* pada quadcopter didapatkan persamaan kinematika seperti pada persamaan (2.22).

$$\begin{aligned} X^E &= X^B \cos\psi + Y^B \cos(90^o + \psi) + Z^B \cos 90^o \\ Y^E &= X^B \cos(90^o - \psi) + Y^B \cos\psi + Z^B \cos 90^o \\ Z^E &= X^B \cos 90^o + Y^B \cos 90^o + Z^B \cos 0^o \end{aligned} \tag{2.22}$$

Persamaan (2.22) dapat diubah dalam bentuk matriks seperti persamaan (2.23) dan (2.24).



Gambar 2.20 Diagram Cartesius Rotasi terhadap Sumbu Z [6]

$$\begin{bmatrix} X^E \\ Y^E \\ Z^E \end{bmatrix} = \begin{bmatrix} cos\,\psi & -sin\,\psi & 0 \\ sin\,\psi & cos\,\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X^B \\ Y^B \\ Z^B \end{bmatrix} \quad (2.23)$$

$$R^{EB}(z,\psi) = \begin{bmatrix} cos\,\psi & -sin\,\psi & 0 \\ sin\,\psi & cos\,\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (2.24)$$

Persamaan (2.18), (2.21) dan (2.24) digunakan untuk memperoleh bentuk matriks transformasi R_{ξ}^{EB} seperti pada persamaan berikut:

$$R_{\xi}^{\ EB} = R^{EB}(x,\varphi).R^{EB}(y,\theta).R^{EB}(z,\psi)$$

$$R_{\xi}^{\ EB} = \begin{bmatrix} c_\psi c_\theta & s_\psi c_\theta & -s_\theta \\ -s_\psi c_\varphi + c_\psi s_\theta s_\varphi & c_\psi c_\varphi + s_\psi s_\theta c_\varphi & c_\theta s_\varphi \\ s_\psi s_\varphi + c_\psi s_\theta c_\varphi & -c_\psi s_\varphi + s_\psi s_\theta c_\varphi & c_\theta c_\varphi \end{bmatrix} \quad (2.25)$$

dengan s(.) menotasikan sin(.) dan c(.) menotasikan cos(.).

Adapun matriks transformasi dari *B-frame* ke koordinat *E-frame* ditunjukkan persamaan berikut:

$$R_{\xi}^{\ BE} = \begin{bmatrix} c_\psi c_\theta & -s_\psi c_\varphi + c_\psi s_\theta s_\varphi & s_\psi s_\varphi + c_\psi s_\theta c_\varphi \\ s_\psi c_\theta & c_\psi c_\varphi + s_\psi s_\theta c_\varphi & -c_\psi s_\varphi + s_\psi s_\theta c_\varphi \\ -s_\theta & c_\theta s_\varphi & c_\theta c_\varphi \end{bmatrix} \quad (2.26)$$

Selain matriks tranformasi R_{ξ}^{BE}, ada pula matriks transformasi T_{Θ}^{BE} untuk mengubah besaran *angular* dari koordinat *B-frame* ke koordinat *E-frame*. Matriks transformasi T_{Θ}^{BE} mengacu pada kecepatan *Euler* dalam *B-frame* dengan membalik pola perputaran sudut *roll*, *pitch*, dan *yaw*.

$$\begin{bmatrix} p \\ q \\ r \end{bmatrix} = I \begin{bmatrix} \dot{\varphi} \\ 0 \\ 0 \end{bmatrix} + R(\varphi,x)^{-1} \begin{bmatrix} 0 \\ \dot{\theta} \\ 0 \end{bmatrix} + R(\varphi,x)^{-1} R(\theta,y)^{-1} \begin{bmatrix} 0 \\ 0 \\ \dot{\psi} \end{bmatrix} \quad (2.27)$$

$$\begin{bmatrix} p \\ q \\ r \end{bmatrix} = T_{\Theta}^{\ BE^{-1}} \begin{bmatrix} \dot{\varphi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} \quad (2.28)$$

Dengan menyelesaikan persamaan (2.28) maka didapatkan matriks transformasi T_{Θ}^{BE}.

$$T_{\Theta}^{BE^{-1}} = \begin{bmatrix} 1 & 0 & -\sin\theta \\ 0 & \cos\varphi & \cos\theta\sin\varphi \\ 0 & -\sin\varphi & \cos\varphi\cos\theta \end{bmatrix}$$

$$T_{\Theta}^{BE} = \begin{bmatrix} 1 & \sin\varphi\tan\theta & \cos\varphi\tan\theta \\ 0 & \cos\varphi & -\sin\varphi \\ 0 & \sin\varphi/\cos\theta & \cos\varphi/\cos\theta \end{bmatrix} \quad (2.29)$$

2.2.2.2 Dinamika Quadcopter [6]

Dinamika merupakan analisis gerak suatu benda dengan memperhatikan gaya yang berpengaruh pada benda tersebut. Analisis dinamika dilakukan dengan menggunakan hukum Newton *Euler* tentang gerak suatu benda. Gaya yang muncul akibat gerak dari quadcopter dapat ditulis sebagai berikut:

$$\begin{bmatrix} F\ trust \\ F\ roll \\ F\ pitch \\ F\ yaw \end{bmatrix} = \begin{bmatrix} U_1 \\ U_2 \\ U_3 \\ U_4 \end{bmatrix} = \begin{bmatrix} F_1 + F_2 + F_3 + F_4 \\ F_2 - F_4 \\ F_1 - F_3 \\ F_1 - F_2 + F_3 - F_4 \end{bmatrix} \quad (2.30)$$

dimana gaya F_i merupakan gaya angkat yang dihasilkan oleh tiap baling-baling motor yang didefinisikan sebagai berikut:

$$F_i = K\frac{\omega}{s+\omega}u_i \quad (2.31)$$

dengan konstanta *K* merupakan gaya dorong, *ω* bandwith motor dan *u* adalah sinyal kontrol dari kontroler. Untuk memperoleh konstanta *K* maka dicari dengan menerbangkan quadcopter pada posisi *hover*. Nilai *K* akan sebanding dengan total gaya angkat dari keempat motor quadcopter saat *hover*.

Pemodelan quadcopter dilakukan dengan kombinasi koordinat *frame* atau disebut *Hibrid–Frame* (*H-frame*). Penurunan model gerak translasi akan diturunkan terhadap koordinat bumi (*E-frame*) karena berkaitan dengan posisi dan kecepatan quadcopter terhadap bumi. Sedangkan penurunan gerak rotasi diturunkan terhadap koordinat *body* (*B-frame*) karena gerak rotasi mempengaruhi gerakan quadcopter itu sendiri.

- ***Penurunan Gerak Translasi***

Berdasarkan aksioma pertama *Euler* dari hukum II Newton didapatkan persamaan gerak translasi sebagai berikut:

$$\sum F^E = \ddot{\xi}^E m \tag{2.32}$$

dengan ΣF^E merupakan resultan gaya yang bekerja pada quadcopter, m merupakan total massa quadcopter dan $\ddot{\xi}^E$ merupakan percepatan gerak quadcopter. Persamaan (2.32) dinyatakan sebagai berikut:

$$F_f{}^E + F_g = \ddot{\xi} m \tag{2.33}$$

dengan $\xi = [x\ y\ z]^T$ merupakan posisi quadcopter terhadap bumi, F_f^E merupakan resultan gaya pada tiap sumbu koordinat, dan $F_g = [0\ 0\ -mg]^T$ merupakan gaya gravitasi. Jika didefinisikan $F_f^B = [F_x\ F_y\ F_z]^T$ merupakan resultan gaya pada koordinat *B-frame,* maka harus ditransformasikan ke dalam koordinat bumi *E-frame*. Karena gaya yang bekerja hanya pada sumbu *z* maka resultan gaya yang muncul hanya F_z atau gaya thrust (U_1).

$$F_f{}^E = R_\xi{}^{BE} F_f{}^B \tag{2.34}$$

$$F_f{}^E = \begin{bmatrix} F_z(s_\psi s_\varphi + c_\psi s_\theta c_\varphi) \\ F_z(-c_\psi s_\varphi + s_\psi s_\theta c_\varphi) \\ F_z(c_\theta c_\varphi) \end{bmatrix} = \begin{bmatrix} U_1(s_\psi s_\varphi + c_\psi s_\theta c_\varphi) \\ U_1(-c_\psi s_\varphi + s_\psi s_\theta c_\varphi) \\ U_1(c_\theta c_\varphi) \end{bmatrix} \tag{2.35}$$

Dengan memasukkan persamaan (2.35) ke (2.33) maka diperoleh:

$$\begin{bmatrix} U_1(s_\psi s_\varphi + c_\psi s_\theta c_\varphi) \\ U_1(-c_\psi s_\varphi + s_\psi s_\theta c_\varphi) \\ U_1(c_\theta c_\varphi) \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ -mg \end{bmatrix} = \begin{bmatrix} \ddot{X} \\ \ddot{Y} \\ \ddot{Z} \end{bmatrix} m \tag{2.36}$$

Setelah persamaan (2.36) diselesaikan maka diperoleh persamaan gerak translasi sebagai berikut

$$\begin{cases} \ddot{X} = (sin\,\psi\,sin\,\varphi + cos\,\psi\,sin\,\theta\,cos\,\varphi)\dfrac{U_1}{m} \\ \ddot{Y} = (-cos\,\psi\,sin\,\varphi + sin\,\psi\,sin\,\theta\,cos\,\varphi)\dfrac{U_1}{m} \\ \ddot{Z} = -g + (cos\,\theta\,cos\,\varphi)\dfrac{U_1}{m} \end{cases} \tag{2.37}$$

- ***Penurunan Gerak Rotasi***

Berdasarkan aksioma kedua *Euler* pada hukum II Newton didapatkan persamaan gerak rotasi sebagai berikut:

$$\tau^E = J\ddot{\Theta}^E \tag{2.38}$$

Persamaan (2.38) merupakan persamaan di koordinat *E-frame*, sehingga perlu ditransformasi ke dalam koordinat *B-frame*. Jika persamaan tersebut dibawa ke dalam koordinat *B-frame* menjadi:

$$\begin{aligned} T_\Theta \tau^B &= J(T_\Theta \dot{\omega}^B) \\ T_\Theta \tau^B &= J(\dot{T}_\Theta \omega^B + T_\Theta \dot{\omega}^B) \end{aligned} \tag{2.39}$$

Jika turunan dari matriks transformasi T_Θ adalah $T_\Theta S(\omega^B)$, dengan $S(\omega^B)$ adalah matriks *skew-symmetric*, maka persamaan (2.39) dapat diubah menjadi (2.40).

$$\begin{aligned} T_\Theta \tau^B &= J(T_\Theta S(\omega^B)\omega^B + T_\Theta \dot{\omega}^B) \\ T_\Theta \tau^B &= T_\Theta(S(\omega^B)J\omega^B + J\dot{\omega}^B) \\ T_\Theta \tau^B &= T_\Theta(\omega^B \times J\omega^B + J\dot{\omega}^B) \end{aligned} \tag{2.40}$$

Karena pada kedua ruas terdapat matriks tranformasi T_Θ, maka dapat dihilangkan atau persamaan (2.40) dapat dianggap sudah ditransformasikan ke dalam koordinat *B-frame* berikut ini:

$$\begin{aligned} \tau^B &= \omega^B \times J\omega^B + J\dot{\omega}^B \\ J\dot{\omega}^B &= -(\omega^B \times J\omega^B) + \tau^B \end{aligned} \tag{2.41}$$

dengan $J = diag\ [J_{xx}\ J_{yy}\ J_{zz}]$ merupakan matriks diagonal yang berisi momen inersia tiap sumbu, $\omega = [p\ \ q\ \ r]^T = [\dot{\varphi}\ \dot{\theta}\ \dot{\psi}]^T$ merupakan kecepatan sudut *Quadrotor*, $\tau = [U_2 l\ U_3 l\ U_4 d]^T$ torsi yang bekerja pada quadcopter, l jarak motor

terhadap pusat massa, dan *d* konstanta drag quadcopter. Untuk mempermudah proses penghitungan maka diselesaikan dulu perkalian *cross product* didalam kurung.

$$-(\omega^B \times J\omega^B) = J\omega^B \times \omega^B$$
$$= \begin{bmatrix} J_{xx} & 0 & 0 \\ 0 & J_{yy} & 0 \\ 0 & 0 & J_{zz} \end{bmatrix} \begin{bmatrix} p \\ q \\ r \end{bmatrix} \times \begin{bmatrix} p \\ q \\ r \end{bmatrix} = \begin{bmatrix} (J_{yy} - J_{zz})qr \\ (J_{zz} - J_{xx})pr \\ (J_{xx} - J_{yy})pq \end{bmatrix} \quad (2.41)$$

Dengan menyelesaikan persamaan (2.41) maka diperoleh persamaan gerak rotasi quadcopter seperti pada persamaan (2.43).

$$J\dot{\omega}^B = -(\omega^B \times J\omega^B) + \tau^B$$
$$\begin{bmatrix} J_{xx}\dot{p} \\ J_{yy}\dot{q} \\ J_{zz}\dot{r} \end{bmatrix} = \begin{bmatrix} (J_{yy} - J_{zz})qr \\ (J_{zz} - J_{xx})pr \\ (J_{xx} - J_{yy})pq \end{bmatrix} + \begin{bmatrix} U_2 l \\ U_3 l \\ U_4 d \end{bmatrix} \quad (2.42)$$

$$\begin{cases} \dot{p} = \dfrac{J_{yy} - J_{zz}}{J_{xx}} qr + \dfrac{U_2 l}{J_{xx}} \\ \dot{q} = \dfrac{J_{zz} - J_{xx}}{J_{yy}} pr + \dfrac{U_3 l}{J_{yy}} \\ \dot{r} = \dfrac{J_{xx} - J_{yy}}{J_{zz}} pq + \dfrac{U_4 d}{J_{zz}} \end{cases} \quad (2.43)$$

Keseluruhan model dinamika quadcopter dinyatakan sebagai berikut:

$$\begin{cases} \ddot{X} = (sin\,\psi\, sin\,\varphi + cos\,\psi\, sin\,\theta\, cos\,\varphi)\dfrac{U_1}{m} \\ \ddot{Y} = (-cos\,\psi\, sin\,\varphi + sin\,\psi\, sin\,\theta\, cos\,\varphi)\dfrac{U_1}{m} \\ \ddot{Z} = -g + (cos\,\theta\, cos\,\varphi)\dfrac{U_1}{m} \\ \dot{p} = \dfrac{J_{yy} - J_{zz}}{J_{xx}} qr + \dfrac{U_2 l}{J_{xx}} \\ \dot{q} = \dfrac{J_{zz} - J_{xx}}{J_{yy}} pr + \dfrac{U_3 l}{J_{yy}} \\ \dot{r} = \dfrac{J_{xx} - J_{yy}}{J_{zz}} pq + \dfrac{U_4 d}{J_{zz}} \end{cases} \quad (2.44)$$

2.2.3 Machine Learning

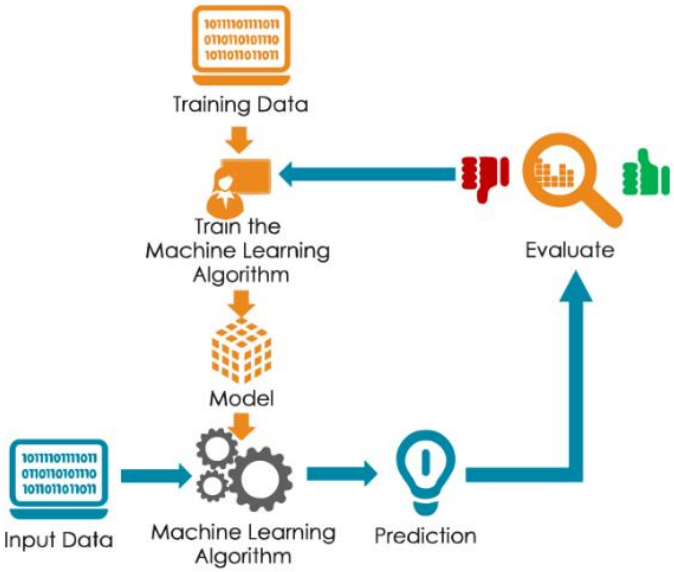
Machine learning adalah aplikasi dari disiplin ilmu kecerdasan buatan (*Artificial Intelligence*) yang menggunakan teknik statistika untuk menghasilkan suatu model otomatis dari sekumpulan data, dengan tujuan memberikan komputer kemampuan untuk "belajar"[7].

Pembelajaran mesin atau *machine learning* memungkinkan komputer mempelajari sejumlah data (*learn from data*) sehingga dapat menghasilkan suatu model untuk melakukan proses *input-output* tanpa menggunakan kode program yang dibuat secara eksplisit. Proses belajar tersebut menggunakan algoritma khusus yang disebut *machine learning algorithms.* Terdapat banyak algoritma machine learning dengan efesiensi dan spesifikasi kasus yang berbeda-beda.

2.2.3.1 Konsep Dasar dan Cara kerja *Machine Learning*

Secara fundamental cara kerja *machine learning* adalah belajar seperti manusia dengan menggunakan contoh-contoh dan setelah itu barulah dapat menjawab suatu pertanyaan terkait [7]. Proses belajar ini menggunakan data yang disebut data *training*. Berbeda dengan program statis, *machine learning* diciptakan untuk membentuk program yang dapat belajar sendiri.

Dari data tersebut, komputer akan melakukan proses belajar (*training*) untuk menghasilkan suatu model. Proses belajar ini menggunakan algoritma *machine learning* sebagai penerapan teknik statistika (seperti pada Gambar 2.21). Model inilah yang menghasilkan informasi, kemudian dapat dijadikan pengetahuan untuk memecahkan suatu permasalahan sebagai proses *input-output.* Model yang dihasilkan dapat melakukan klasifikasi atau pun prediksi kedepannya. Untuk memastikan efesiensi model yang terbentuk, data akan dibagi menjadi data pembelajaran (data *training*) dan data pengujian (data *testing*). Pembagian data yang digunakan bervariasi bergantung algoritma yang digunakan. Pada umumnya data *training* lebih banyak dari data *testing*, misalnya dengan rasio 3:1 [7]. Data *testing* digunakan untuk menghitung seberapa efesien model yang dihasilkan untuk melakukan klasifikasi atau prediksi kedepannya yang disebut *test score*. Semakin banyak data yang digunakan, *test score* yang dihasilkan semakin baik. Nilai *test score* berada dalam rentang 0-1.



Gambar 2.21 Cara Kerja *Machine Learning* [7]

2.2.3.2 Algoritma *Machine Learning* KNN *(K-Nearest Neighbour)*

KNN atau *K-Nearest Neighbour* adalah sebuah metode yang menggunakan algoritma *supervised learning* dimana melakukan klasifikasi terhadap objek berdasarkan data *training* yang memiliki jarak terdekat dengan objek atau data *testing*. Metode ini sangatlah sederhana karena tidak membutuhkan model klasifikasi khusus untuk mencocokkan jarak antara data *training* dan data *testing*, prinsip kerjanya berdasarkan memori.

Data *training* diproyeksikan kedalam ruang-ruang vektor, dimana setiap ruang vektor merepresentasikan fitur klasifikasi setiap data yang dideskripsikan dengan atribut numerik (*ground truth*). Jika sebuah data yang tidak diketahui labelnya diberikan data (*testing*), maka KNN akan mencari *k* buah nilai data *training* yang jaraknya paling dekat dengan data *input* tersebut pada ruang vektor. Jarak antara data *training* dan data *testing* dapat dihitung berdasarkan *Euclidean Distance* dengan rumus matematika sebagai berikut:

$$D(x,y) = \sqrt{\sum_{k=1}^{d}(a_k - b_k)^2} \tag{2.45}$$

dimana,

D : *Euclidean distance*

a, b : Data *training* dan data *testing*

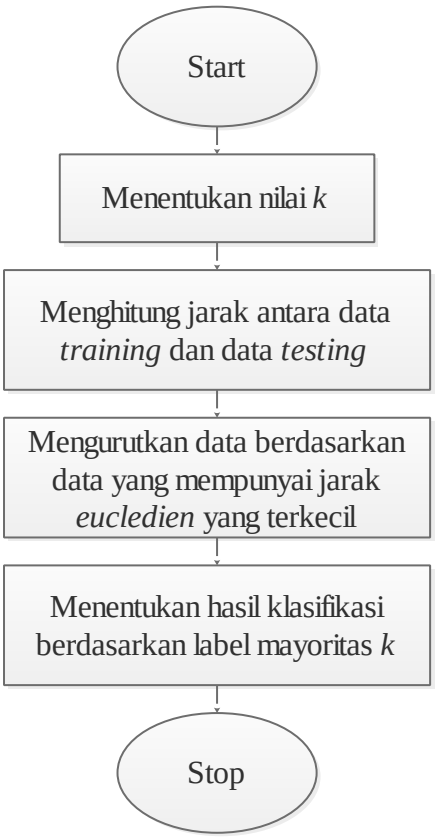
k : Jumlah tetangga

d : Maksimal jumlah tetangga yang diberikan

Pada tahap *learning*, algoritma ini hanya melakukan penyimpanan vektor-vektor fitur dapa pada tahap klasifikasi, kemudian pada tahap *testing* jarak dari vektor yang baru terhadap seluruh vektor data *training* dihitung dan sejumlah *k* yang paling dekat diambil. Nilai k data *training* terdekat akan melakukan *voting* untuk menentukan label mayoritas.

Label data *testing* akan ditentukan berdasarkan label mayoritas dan jika terdapat lebih dari satu label mayoritas maka dipilih secara acak antara label mayoritas terebut. Menentukan nilai *k* terbaik pada algoritma ini bergantung pada data yang ada. Secara umum, nilai *k* yang tinggi akan mengurangi *noise* pada hasil klasifikasi namun juga dapat membuat batasan setiap klasifikasi menjadi lebih kabur. Menentuan nilai *k* yang bagus dapat dipilih dengan optimasi parameter menggunakan *cross-validation*.

Ketepatan algoritma KNN sangat dipengaruhi oleh ada atau tidaknya fitur yang tidak relevan atau bobot fitur tidak setara dengan relevansi hasil klasifikasi. Penelitian terhadap algoritma ini sebagian besar mengenai pemilihan dan pemberian bobot nilai terhadap fitur sehingga menghasilkan klasifikasi yang baik. Kelebihan dari metode ini adalah kokoh terhadap data *training* yang memiliki banyak *noise* dan sangat efektif pada data yang besar. Sedangkan kekurangan dari metode klasifikasi ini adalah diperlukan nilai parameter *k* (jumlah tetangga terdekat), dan biaya komputasi yang cukup tinggi karena memerlukan perhitungan jarak dari setiap *query instance* pada *data training*. Gambar 2.22 merupakan tahapan proses klasifikasi KNN secara garis besar.

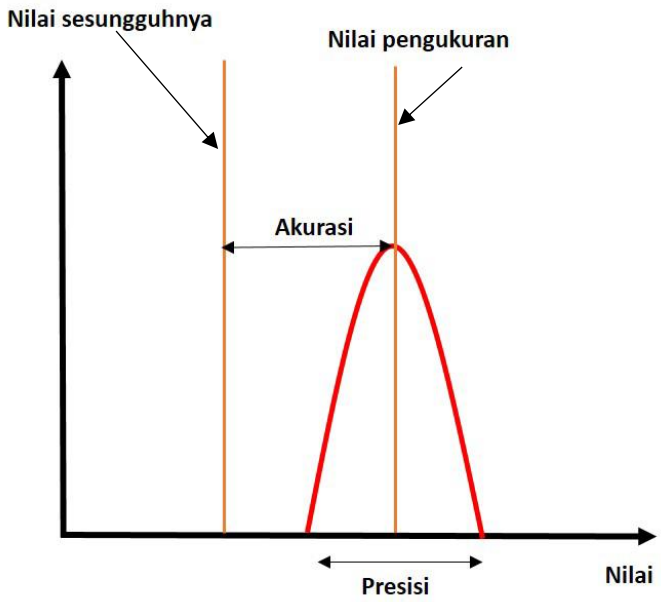


Gambar 2.22 Flowchart KNN

2.2.3.3 Pengujian Akurasi

Teknik pengujian dilakukan terhadap hasil klasifikasi KNN yang dilakukan. Untuk mengetahui seberapa besar hasil klasifikasi dibutuhkan metode perhitungan hasil yaitu akurasi [8]. Akurasi dekat kaitannya dengan presisi. Akurasi didefinisikan sebagai tingkat kedekatan antara nilai prediksi dengan nilai sebenarnya, sedangkan presisi menunjukkan kedekatan perbedaan nilai pada saat dilakukan pengukuran. Ilustrasi perbedaan antara presisi dan akurasi ditunjukkan pada Gambar 2.23. Perhitungan akurasi dinyatakan dalam persamaan:

$$Akurasi = \frac{Banyak\ Data\ yang\ Diklasifikasikan\ Benar}{Jumlah\ data\ yang\ Diklasifikasikan} \times 100\% \quad (2.46)$$



Gambar 2.23 Perbedaan akurasi dan Presisi [8]

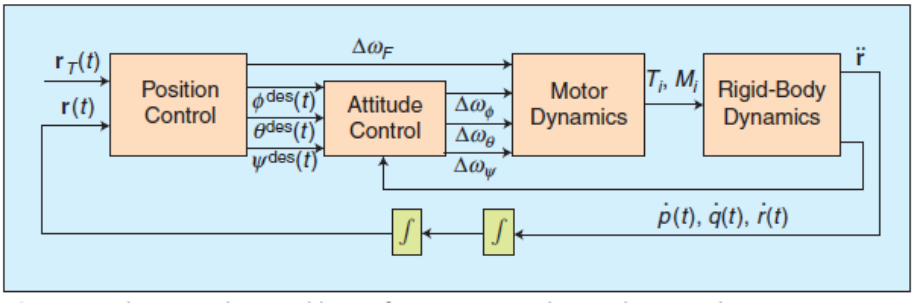
2.2.4 Kontrol Quadcopter

Skema kontrol quadcopter dibagi menjadi 2, yaitu bagian *outer loop* kontroler posisi dan *inner loop* kontroler *attitude* yang dapat dilihat pada Gambar 2.24. Kontrol posisi memiliki masukan posisi target $r_T = [X_T\ Y_T\ Z_T]$ dan posisi quadcopter $r = [X\ Y\ Z]$, sehingga nilai percepatan yang diinginkan agar r mencapai target ($\ddot{r}^{des}$) dapat diketahui dengan:

$$(\ddot{r}_{i,T} - \ddot{r}_i^{des}) + k_{d,i}(\dot{r}_{i,T} - \dot{r}_i) + k_{p,i}(r_{i,T} - r_i) + k_{i,i}\int (r_{i,T} - r_i) = 0 \quad (2.47)$$

Linearisasi dari gerak translasi quadcopter persamaan (2.37) menghasilkan hubungan antara ($\ddot{r}^{des}$) dan sudut *roll* dan *pitch*, yaitu:

$$\begin{aligned} \ddot{r}_1^{des} &= g(\theta^{des} cos\psi_T + \phi^{des} sin\psi_T) \\ \ddot{r}_2^{des} &= g(\theta^{des} sin\psi_T - \phi^{des} cos\psi_T) \\ \ddot{r}_3^{des} &= \frac{U_1}{m} - g \end{aligned} \quad (2.48)$$



Gambar 2.24 Blok Diagram Kontrol Quadcopter [9]

Nilai sudut *roll*, *pitch*, *yaw* yang harus dicapai (φ^{des}, θ^{des}, ψ^{des}) dan besar gaya *thrust* U_1 atau $\Delta\omega_F$ dari nilai percepatan posisi yang diinginkan dapat diketahui dari *invers* dari persamaan (2.54), yaitu:

$$\begin{aligned} \varphi^{des} &= \frac{1}{g}\left(\ddot{r}_1^{des} sin\psi_T - \ddot{r}_2^{des} cos\psi_T\right), \\ \theta^{des} &= \frac{1}{g}\left(\ddot{r}_1^{des} cos\psi_T + \ddot{r}_2^{des} sin\psi_T\right), \\ \Delta\omega_F &= mg + \ddot{r}_3^{des} \end{aligned} \tag{2.49}$$

Berikut kontroler *proportional-derivative* digunakan untuk mengatur *attitude* quadcopter:

$$\begin{aligned} \Delta\omega_\varphi &= k_{p,}(\varphi^{des} - \varphi) + k_{d,\varphi}(p^{des} - p), \\ \Delta\omega_\theta &= k_{p,\theta}(\theta^{des} - \theta) + k_{d,\theta}(q^{des} - q), \\ \Delta\omega_\psi &= k_{p,\psi}(\psi^{des} - \psi) + k_{d,\psi}(r^{des} - r) \end{aligned} \tag{2.50}$$

dimana,

φ : Sudut *roll* quadcopter

θ : Sudut *pitch* quadcopter

ψ : Sudut *yaw* quadcopter

$\Delta\omega_\varphi$: *Bandwith* untuk gerak *roll*

$\Delta\omega_\theta$: *Bandwith* untuk gerak *roll*

$\Delta\omega_\psi$: *Bandwith* untuk gerak *roll*

Nilai dari *bandwith* masing-masing gerak *attitude* dan *thrust* digunakan untuk masukan pada dinamika motor, sehingga dapat menggerakkan dinamika quadcopter sesuai posisi target yang diinginkan.

BAB 3
METODE PENELITIAN

Pada bab ini membahas tentang tahapan konseptual sistem yang akan dibuat. Quadcopter akan digunakan sebagai *plant* dalam sistem adalah Quanser Qdrone. Pada penelitian ini, sistem navigasi quadcopter dalam menghindari halangan ditentukan dengan menggunakan *machine learning* untuk menghasilkan arah hindar yang efisien.

3.1 Plant Quadcopter

Tipe quadcopter yang akan digunakan pada penelitian ini adalah Quanser Qdrone pada Gambar 3.1. Quanser Qdrone merupakan salah satu jenis quadcopter yang dirancang untuk penelitian atau riset yang dilakukan di luar ruangan. Qdrone memiliki kerangka serat karbon yang tahan lama dan ringan sehingga mudah bermanuver dan mampu melakukan aplikasi yang memiliki resiko kerusakan yang tinggi. Ukuran Quanser Qdrone tergolong sedang yaitu $40\ x\ 40\ x\ 15$ cm dan dilengkapi dengan pelindung *propeller*.

Gambar 3.1 Quadcopter Quanser Qdrone

Pemodelan kinematika dan dinamika quadcopter telah dijelaskan pada bab sebelumnya. Model keseluruhan sistem quadcopter [6] yaitu:

$$\begin{cases} \ddot{X} = (sin\,\psi\,sin\,\varphi + cos\,\psi\,sin\,\theta\,cos\,\varphi)\dfrac{U_1}{m} \\ \ddot{Y} = (-cos\,\psi\,sin\,\varphi + sin\,\psi\,sin\,\theta\,cos\,\varphi)\dfrac{U_1}{m} \\ \ddot{Z} = -g + (cos\,\theta\,cos\,\varphi)\dfrac{U_1}{m} \\ \dot{p} = \dfrac{J_{yy} - J_{zz}}{J_{xx}}qr + \dfrac{U_2 l}{J_{xx}} \\ \dot{q} = \dfrac{J_{zz} - J_{xx}}{J_{yy}}pr + \dfrac{U_3 l}{J_{yy}} \\ \dot{r} = \dfrac{J_{xx} - J_{yy}}{J_{zz}}pq + \dfrac{U_4 d}{J_{zz}} \end{cases} \quad (3.1)$$

dengan X, Y, Z adalah posisi quadcopter sedangkan p, q, r adalah kecepatan *roll*, *pitch* dan *yaw*. Parameter digunakan quadcopter seperti Tabel 3.1, sesuai dengan tipe Quanser Qdrone.

Tabel 3.1 Parameter Quanser Qdrone [13]

Parameter	**Simbol**	**Nilai**
Massa (kg)	m	1
Gravitasi (kg/ m^2)	g	9,81
Momen inersia pada sumbu X (kg.m^2)	J_{xx}	0,03
Momen inersia pada sumbu X (kg.m^2)	J_{yy}	0,03
Momen inersia pada sumbu X (kg.m^2)	J_{zz}	0,04
Jarak rotor dari pusat massa (m)	l	0,2
Gaya Drag (N)	d	$3{,}13 \times 10^{-5}$
Gaya trust (N)	b	$7{,}5 \times 10^{-7}$
Bandwith actuator (*rad*/s)	ω	15
Konstanta gaya dorong (N)	K	120

3.2 Kontrol Quadcopter

Kontroler *proportional-derivative* yang digunakan pada penelitian ini berfungsi untuk mengatur gerak quadcopter dari titik awal sampai titik target yang diinginkan pada lingkungan 3D. Dalam kontrol quadcopter, terdapat *inner loop* sebagai kontrol *attitude* dan *outer loop* untuk kontrol posisi [9] seperti yang dijelaskan pada Gambar 3.2.

Kontrol posisi berfungsi untuk menghasilkan nilai sudut *roll, pitch, yaw* yang harus dicapai ($\varphi^{des},\ \theta^{des}, \psi^{des}$) dan u_{alt}. Pada saaat kondisi *hover* ($u_{alt} = mg\ , \theta = \psi = \varphi = 0$), liniearisasi pada persamaan percepatan quadcopter ($\ddot{X}, \ddot{Y}, \ddot{Z}$) yang menghasilkan:

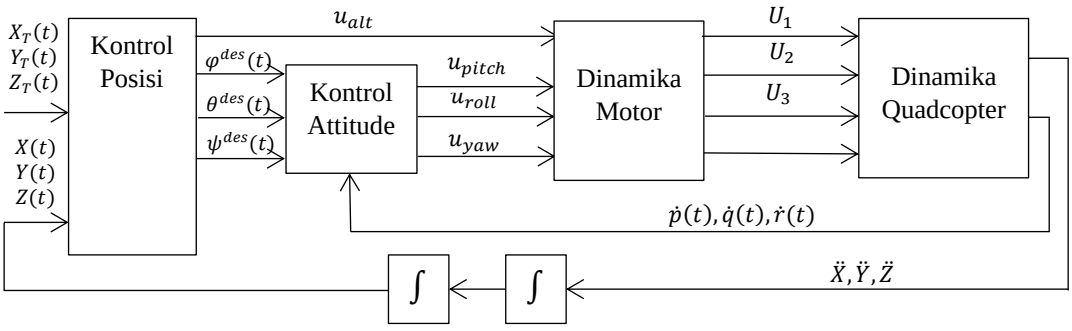
$$\begin{aligned} \ddot{X}^{des} &= g(\theta^{des}\, cos\psi_0 + \varphi^{des} sin\psi_0) \\ \ddot{Y}^{des} &= g(\theta^{des}\, sin\, \psi_0 - \varphi^{des} \cos\psi_0) \\ \ddot{Z}^{des} &= \frac{u_{alt}}{m} - g \end{aligned} \tag{3.3}$$

Sedangkan $\ddot{X}^{des}$, $\ddot{Y}^{des}$ dan $\ddot{Z}^{des}$ diketahui dari persamaan kontroler *proportional-derivative* berikut:

$$\begin{aligned} \ddot{X}^{des} &= \ddot{X}_T + k_{d,X}\left(\dot{X}_T - \dot{X}\right) + k_{p,X}(X_T - X) \\ \ddot{Y}^{des} &= \ddot{Y}_T + k_{d,Y}\left(\dot{Y}_T - \dot{Y}\right) + k_{p,Y}(Y_T - Y) \\ \ddot{Z}^{des} &= \ddot{Z}_T + k_{d,Z}\left(\dot{Z}_T - \dot{Z}\right) + k_{p,Z}(Z_T - Z) \end{aligned} \tag{3.5}$$

Sehingga nilai u_{alt} yaitu:

$$u_{alt} = mg + m\ddot{Z}^{des} \tag{3.6}$$



Gambar 3.2 Blok Diagram Kontrol Quadcopter

Dari eliminasi persamaan (3.3), sudut *roll*, *pitch* dan *yaw* yang diinginkan adalah:

$$\begin{aligned}\varphi^{des} &= \frac{1}{g}\left(\ddot{X}^{des} sin\psi_T - \ddot{Y}^{des} cos\psi_T\right),\\ \theta^{des} &= \frac{1}{g}\left(\ddot{X}^{des} cos\psi_T + \ddot{Y}^{des} sin\psi_T\right),\\ \psi^{des} &= \psi_T\end{aligned} \tag{3.7}$$

Berikut kontroler *proportional-derivative* pada kontrol *attitude:*

$$\begin{aligned}u_{roll} &= k_{p,\phi}(\varphi^{des} - \phi) + k_{d,\phi}(p^{des} - p)\\ u_{pitch} &= k_{p,\theta}(\theta^{des} - \theta) + k_{d,\phi}(q^{des} - q)\\ u_{yaw} &= k_{p,\psi}(\psi^{des} - \psi) + k_{d,\phi}(r^{des} - r)\end{aligned} \tag{3.8}$$

Nilai $u_{alt}, u_{roll}, u_{pitch}$ dan u_{yaw} adalah sinyal kontrol yang digunakan pada dinamika motor dan menghasilkan besar gaya F_1, F_2, F_3, F_4 untuk nilai vektor gaya pada persamaan (2.30) untuk dinamika quadcopter. Berikut adalah persamaan pemodelan motor pada *propeller:*

$$F_i = K\frac{\omega}{s+\omega}u_i \tag{3.2}$$

dimana,

F_i : Gaya pada setiap motor

ω : lebar *bandwidth* motor

K : Konstanta gaya dorong

u_i : Sinyal kontrol untuk setiap motor

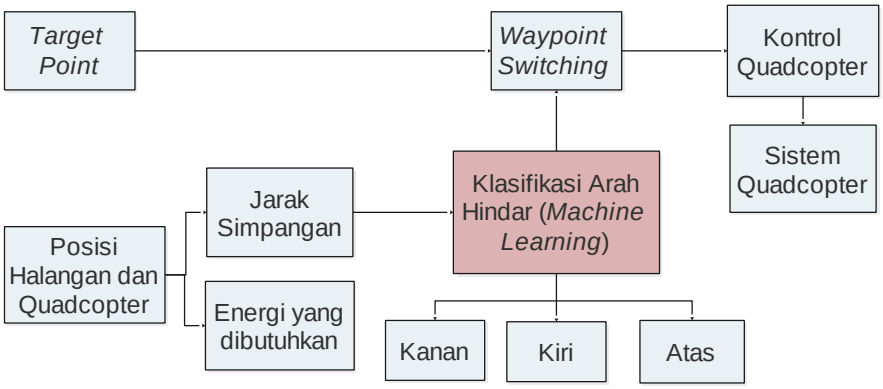
Dari hasil tuning berdasarkan eksperimen, parameter kontrol *propotional-derivative* yang digunakan pada penelitian ini adalah:

$$k_{p,X} = 5, k_{p,Y} = 5\ , k_{p,Z} = 20$$

$$k_{d,X} = 5, k_{d,Y} = 5\ , k_{d,Z} = 10$$

$$k_{p,\phi} = 3000\ , k_{p,\theta} = 3000, k_{p,\psi} = 3000$$

$$k_{d,\phi} = 300\ , k_{d,\theta} = 300, k_{d,\psi} = 300$$



Gambar 3.3 Skema Sistem

3.3 Sistem Navigasi Penghindaran Halangan dengan Machine Learning

Sistem navigasi penghindaran halangan menggunakan *machine learning* pada penelitian ini dapat dilihat pada Gambar 3.3. Titik target yang akan dicapai quadcopter adalah (X_T, Y_T, Z_T). *Waypoint switching* akan merubah *target point* menjadi titik hindar apabila ada halangan yang terdeteksi. Sedangkan arah hindar akan ditentukan secara otomatis dengan menggunakan *machine learning*.

Pada klasifikasi *machine learning*, fitur yang digunakan adalah jarak masing-masing simpangan dan energi yang dibutuhkan. Hasil keputusan dari *machine learning* adalah arah hindar efisien untuk quadcopter menghindari halangan, yang akan dikirim pada *waypoint switching*. *Waypoint switching* akan mengubah *setpoint* posisi quadcopter yang semula menuju titik target menjadi titik hindar.

3.3.1 Deteksi Halangan

Deteksi halangan digunakan untuk memeriksa adanya halangan saat quadcopter bergerak menuju titik target. Halangan akan terdeteksi pada saat jarak 0.5 m antara quadcopter dan halangan. Apabila halangan terdeteksi, maka algoritma *machine learning* akan menentukan titik hindar yang efisien.

Proses tersebut akan terus berjalan sampai quadcopoter mencapai titik target yang diinginkan. Pada penelitian ini, halangan yang diberikan berupa halangan statis dan dinamis.

3.3.2 Perancangan Klasifikasi menggunakan Machine Learning

Pada metode klasifikasi menggunakan *machine learning*, terdapat dua proses yaitu *training data* dan *testing data*. Pada saat *training data*, akan dilakukan percobaan dengan berbagai kondisi quadcopter terhadap halangan dan membagi menjadi 3 arah hindar yaitu kanan, kiri dan atas. *Training data* tersebut akan disimpan dalam *database*. Pada fase *testing*, akan muncul data baru dan akan diklasifikasikan sesuai dengan *training data.*

Pada *training* data, fitur klasifikasi pada penelitian ini memiliki 3 variabel (μ_l, μ_r, μ_u) yang masing-masing memiliki data jarak simpangan $(\delta_l, \delta_r, \delta_u)$ dan energi yang dibutuhkan $(\Delta E_{R,l}, \Delta E_{R,r}, \Delta E_{R,u})$. Dari fitur tersebut, akan diklasifikasikan menjadi arah hindar kanan, kiri dan atas. Persamaan untuk menghitung variabel fitur terdapat pada persamaan (3.9). Nilai bobot W_s dan W_e diberikan untuk menentukan dominan antara jarak dengan energi. Sedangkan $\Delta E_{R,i}$ adalah nilai energi yang dibutuhkan pergerakan quadcopter dari titik koordinat quadcopter (X, Y, Z) sampai koordinat titik hindar $(X_K,\ Y_K,\ Z_K)$. Begitu juga dengan jarak simpangan δ_i.

$$\mu_i = \frac{(W_s \delta_i + W_e\ \Delta E_{R,i})}{2} \tag{3.9}$$

dimana,

μ_i : Fitur i

W_s : Bobot jarak simpangan

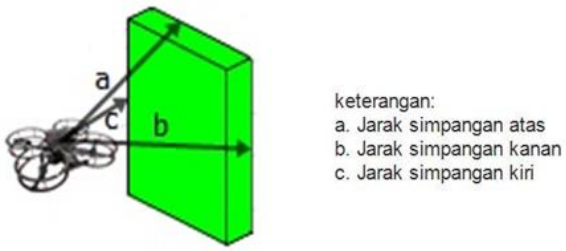
W_e : Bobot energi

δ_i : Jarak simpangan i

$\Delta E_{R,i}$: Total energi dari posisi quadcopter ke titik i

3.3.2.1 Jarak simpangan

Jarak simpangan antara halangan dan quadcopter akan diukur pada jarak 0.5m saat quadcopter mendeteksi halangan. Jarak simpangan yang diukur adalah jarak simpangan kanan, kiri dan atas. Jarak simpangan diukur dari koordinat quadcopter (X, Y, Z) sampai koordinat titik hindar $(X_K,\ Y_K,\ Z_K)$ yang dijelaskan pada Gambar 3.4.



Gambar 3.4 Ilustrasi Jarak Simpangan

Masing- masing jarak simpangan akan dihitung menggunakan persamaan (3.10) dan diilustrasikan pada Gambar 3.5 dan 3.6. Hasil pengukuran tersebut menjadi nilai pada variabel simpangan ($\delta_{l,}\ \delta_{r,}\ \delta_{u}$).

$$\delta_i = \tan\theta_i \,.\, d_i$$
$$\theta_i = tan^{-1}\left(\frac{h_i + t}{d_i}\right) \tag{3.10}$$

dimana,

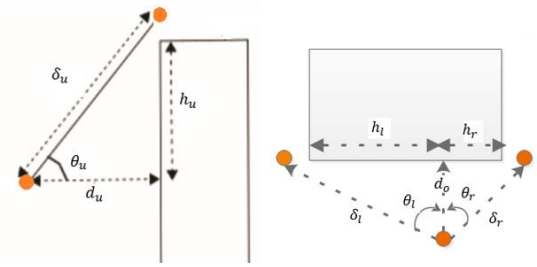
δ_i : Jarak simpangan i

θ_i : Sudut simpangan i

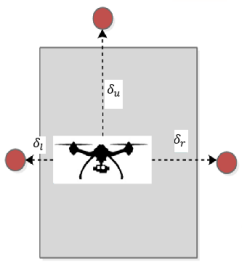
h_i : Tinggi halangan i

t : Jarak aman

d_o : Jarak halangan terhadap quadcopter



Gambar 3.5 Pengukuran Jarak Simpangan



Gambar 3.6 Jarak Simpangan antara Quadcopter dan Titik Hindar

3.3.2.2 Energi [9]

Energi kinetik terjadi apabila quadcopter bergerak tanpa perubahan *altitude*. Sedangkan kecepatan untuk menghitung total energi kinetik menggunakan persamaan (3.11).

$$\Delta E_k = \frac{1}{2} m V^2 \tag{3.11}$$

dimana,

m = Massa quadcopter (kg)

V = Kecepatan quadcopter (m/s)

Diasumsikan bahwa tidak ada energi yang terbuang saat quadcopter mengurangi *altitude*, perbedaan energi potensial dihitung dengan (3.12).

$$\Delta E_p = m.g.\Delta h \tag{3.12}$$

dimana,

m = Massa quadcopter

g = Gravitasi

Δh = Perbedaan ketinggian koordinat quadcopter dan titik hindar

Nilai total energi yang dibutuhkan $\Delta E_{R,i}$ adalah hasil penjumlahan dari energi potensial dan energi kinetik dari pergerakan quadcopter dari titik koordinat quadcopter (X_R, Y_R, Z_R) sampai koordinat titik hindar (X_K, Y_K, Z_K). Nilai total energi yang dibutuhkan $\Delta E_{R,i}$ menggunakan:

$$\Delta E_{i,j} = \Delta E_p + \Delta E_k \tag{3.13}$$

dimana,

ΔE_p = Total energi Potensial

ΔE_k = Total Energi Kinetik

Terdapat perbedaan nilai total energi yang dibutuhkan antara menghindar ke kiri/kanan dan ke atas. Pada saat quadcopter menghindar ke kanan/kiri, ΔE_p adalah nol karena tidak ada perubahan ketinggian. Sedangkan saat quadcopter

menghindar ke atas terdapat perubahan ketinggian yang menyebabkan terjadinya energi potensial ΔE_p.

3.3.3 Perancangan Metode Klasifikasi KNN

Klasifikasi pada penelitian ini dilakukan dengan menggunakan salah satu metode *machine learning* KNN (*K-Nearest-Neighbour*). Dalam penggunaan KNN, *data training* dihasilkan dari perhitungan masing-masing jarak simpangan yang dihasilkan dari persamaan (3.10). Keputusan hasil klasifikasi arah hindar ditentukan dari nilai minimal dari persamaan (3.9) pada masing masing simpangan. Contoh data *training* dan data keputusan dapat dilihat pada Tabel 3.2 dan 3.3. Nilai *k* digunakan untuk menentukan jumlah *neighbour* yang terdekat dengan data yang baru. Untuk menghitung jarak terdekat dengan *neighbour*, menggunakan:

$$j = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2} \tag{3.14}$$

Tabel 3.2 Contoh Data *Training*

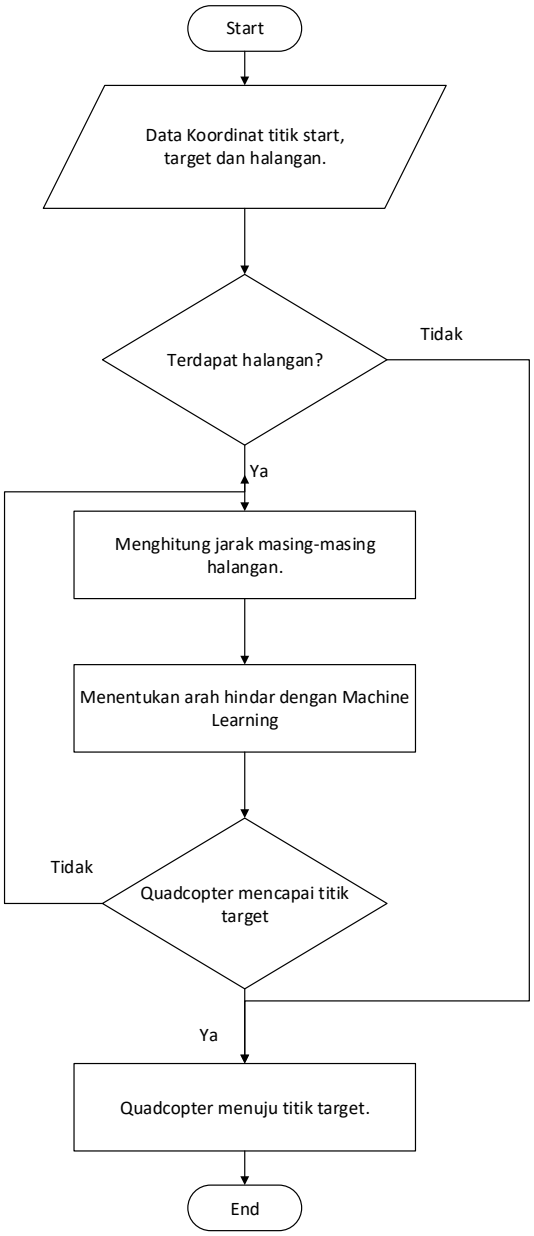
Data ke i	δ_u (m)	δ_r (m)	δ_l (m)	Kelas
1	5.361	3.372	2.468	Kiri
2	4.906	2.314	4.897	Kanan
3	1.786	6.876	6.768	Atas

Tabel 3.3 Contoh Data Keputusan Klasifikasi

Data ke i	μ_u	μ_r	μ_l	Kelas
1	100.31	23.78	9.25	Kiri
2	95.14	6.32	22.88	Kanan
3	30.45	42.33	41.12	Atas

3.3.4 Algoritma Sistem Penghindaran Halangan dengan Machine learning pada Halangan Statis

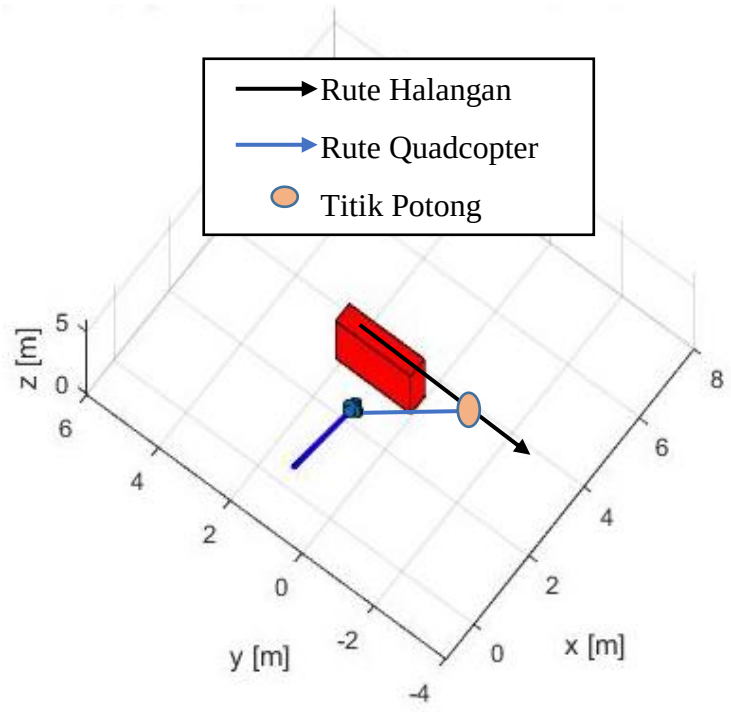
Algoritma sistem Penghindaran halangan pada penelitian ini digunakan untuk menjelaskan urutan cara kerja sistem yang diajukan. Arah hindar dalam menghindari halangan statis ditentukan oleh hasil klasifikasi *machine learning*.



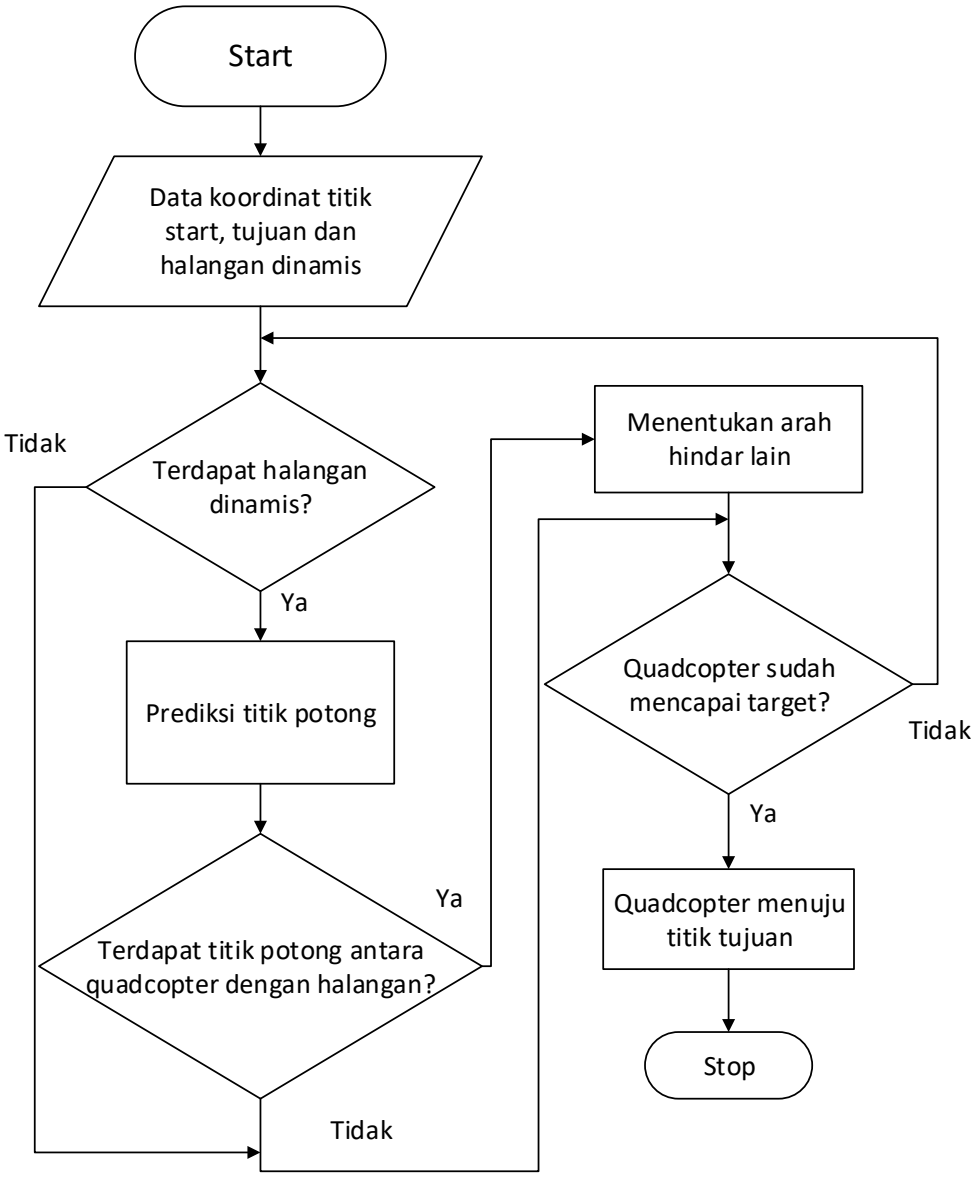
Gambar 3.7 Flowchart Sistem PenghindaranHalangan Statis dengan *Machine Learning*

3.3.5 Algoritma Sistem Penghindaran Halangan dengan Machine learning pada Halangan Dinamis

Halangan dinamis adalah halangan yang dapat bergerak pada arah dan kecepatan tertentu. Pada sistem yang dirancang sebelumnya, apabila terdapat halangan maka arah hindar yang efisien akan ditentukan oleh KNN. Namun, pada halangan dinamis diperlukan suatu mekanisme yaitu saat terdeteksi adanya halangan, maka akan diprediksi apakah arah hindar quadcopter dan halangan bertemu pada titik potong (seperti pada Gambar 3.8) pada waktu yang sama. Apabila bertemu, quadcopter harus mengindari halangan dinamis tersebut dengan arah hindar yang lain. Proses perancangan menghindari halangan pada halangan dinamis dapat dilihat pada Gambar 3.9.



Gambar 3.8 Ilustrasi Titik Potong Halangan dan Quadcopter



Gambar 3.9 *Flowchart* Sistem Penghindaran Halangan Statis dengan *Machine Learning*

BAB 4
HASIL DAN PEMBAHASAN

Pada Bab ini, akan dibahas tentang hasil algoritma yang dirancang dengan beberapa pengujian. Karena algoritma yang dirancang menggunakan *mahine learning* KNN (*K-Nearest-Neighbour*) untuk proses klasifikasi arah hindar, maka akan dibahas terlebih dahulu fitur-fitur yang digunakan untuk *data training* dan akan diuji tingkat akurasinya. Selanjutnya, algoritma akan diuji dengan quadcopter yang harus mencapai titik target dengan beberapa kasus halangan statis dan dinamis.

4.1 Klasifikasi KNN (K-Nearest-Neighbour)

Pada proses Klasifikasi KNN (*K-Nearest-Neighbour*), diperlukan data *training* yang digunakan untuk pembanding pada tahap klasifikasi dengan data *testing*. Data *training* pada penelitian ini dibuat berdasarkan percobaan untuk menghasilkan fitur-fitur dan kelas seperti yang dijelaskan pada bab sebelumnya. Dalam penentuan kelas (arah hindar kiri, kanan atau atas), menggunakan persamaan (3.9) dengan nilai bobot Jarak Simpangan (W_s) dan Energi (W_e). Klasikasi KNN membutuhkan nilai k sebagai penentu banyak data sebagai nilai terdekat dengan data *testing*. Nilai k yang digunakan dalam klasikasi KNN ini adalah bilangan ganjil mulai dari 3 sampai 9.

4.1.1 Data Fitur Klasifikasi

Fitur-fitur klasifikasi KNN yang digunakan pada penelitian ini adalah jarak simpangan antara posisi quadcopter dan halangan seperti yang dijelaskan pada Bab 3. Data masing-masing simpangan δ (simpangan kiri, kanan dan atas) didapatkan berdasarkan percobaan dengan posisi quadcopter dan halangan yang berbeda-beda sehingga menghasilkan data simpangan δ yang berbeda sebanyak 52 data. Data fitur-fitur yang digunakan dapat dilihat pada Tabel 4.1 (Data lengkap Tabel 4.1 terdapat pada Lampiran).

Tabel 4.1 Data Fitur-Fitur yang digunakan

Data ke-	Simpangan Atas (m)	Simpangan Kiri (m)	Simpangan Kanan (m)
1.	0.6474	0.6225	2.3835
2.	0.6474	0.6815	2.2932
3.	0.6474	0.7472	2.2031
4.	0.6474	0.8182	2.1132
5.	0.6474	0.8931	2.0235
6.	0.6474	0.9710	1.9341
7.	0.6474	1.0512	1.8450
8.	0.6266	1.2088	1.6766
9.	0.6266	1.3004	1.5815
10.	0.6266	1.3932	1.4870
:	:	:	:
:	:	:	:
52.	2.9531	1.5799	1.3018

4.1.2 Perbandingan Nilai Variabel Bobot Jarak Simpangan (W_s) dan Energi (W_e) dengan Hasil Klasifikasi.

Pada proses pembuatan data *training*, data fitur pada Tabel 4.1 diklasifikasikan menjadi 3 kelas, yaitu arah hindar kanan, kiri dan atas. Setiap arah hindar memiliki nilai jarak simpangan (δ) dan energi yang dibutuhkan (ΔE). energi yang dibutuhkan (ΔE) pada setiap simpangan terdapat pada Tabel 4.2.

Masing-masing arah hindar juga memiliki nilai μ berdasarkan persamaan (3.9). Maka penentuan kelas yaitu berdasarkan nilai μ yang terkecil pada masing-masing arah hindar. Pada persamaan (3.9), terdapat nilai bobot jarak simpangan dan energi. Nilai bobot akan berpengaruh dalam menentukan kelas arah hindar. Perbedaan tersebut dapat dilihat pada Tabel 4.3 dengan total data *training* 52.

Tabel 4.2 Data Energi yang Dibutuhkan

Data ke-	Energi Simpangan Atas (Joule)	Energi Simpangan Kiri (Joule)	Energi Simpangan Kanan (Joule)
1.	4.2439	0.1937	2.8405
2.	4.2439	0.2322	2.6293
3.	4.2439	0.2791	2.4268
4.	4.2439	0.3347	2.2328
5.	4.2439	0.8931	2.0472
6.	4.2439	0.3988	1.8703
7.	4.2439	0.4714	1.7020
8.	3.9011	0.5525	1.4054
9.	3.9011	0.7305	1.2505
10.	3.9011	0.8455	1.1055
:	:	:	:
52.	32.912	1.2480	1.2480

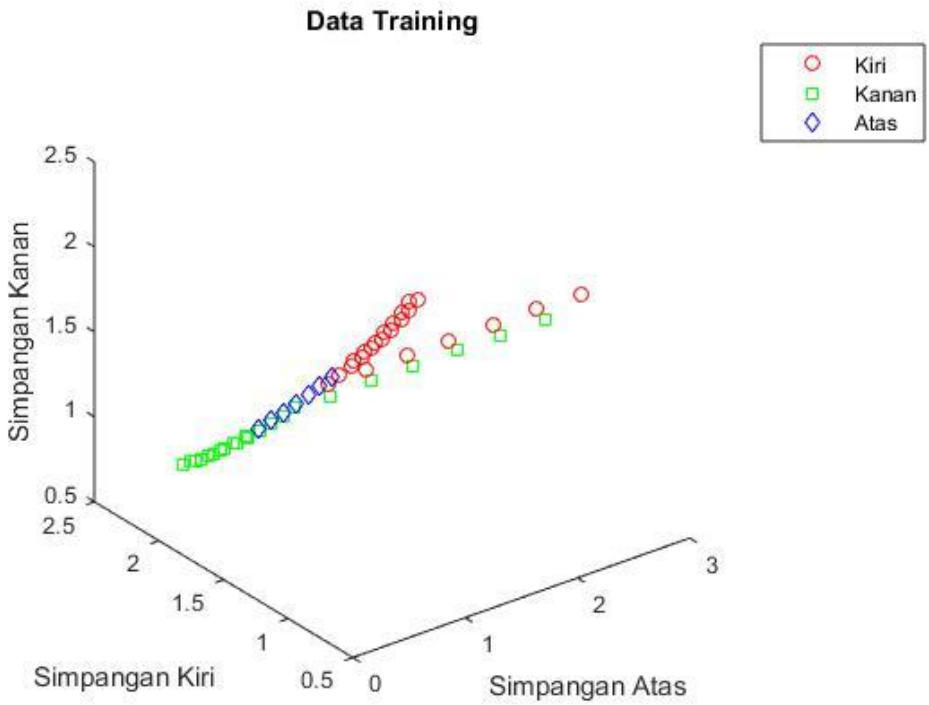
Tabel 4.3 Perbedaan Nilai Bobot

Nilai Bobot	Jumlah Arah Hindar Kanan	Jumlah Arah Hindar Kiri	Jumlah Arah Hindar Atas
Jarak $>$ Energi ($\boldsymbol{W_s = 5, W_e = 1}$)	23	22	7
Jarak $<$ Energi ($\boldsymbol{W_s = 1, W_e = 5}$)	26	26	0
Jarak = Energi ($\boldsymbol{W_s = 1, W_e = 1}$)	26	26	0

Pada Tabel 4.2, dapat dilihat bahwa dengan nilai bobot Jarak $>$ Energi menghasilkan 7 arah hindar ke atas sedangkan yang lain tidak. Dalam menentukan arah hindar dengan efisiensi energi dan jarak simpangan, arah hindar ke atas tetap diperhitungkan karena sebagian memiliki jarak yang terpendek meskipun memerlukan energi yang lebih besar. Maka dari itu, bobot jarak untuk data *training* yaitu $(\boldsymbol{W_s = 5, W_e = 1})$. Data *training* yang digunakan pada penelitian ini dapat dilihat pada Tabel 4.4 dan Gambar 4.1.

Tabel 4.4 Data *Training*

Data ke-	Simpangan Atas (m)	Simpangan Kiri (m)	Simpangan Kanan (m)	Kelas
1.	0.6474	0.6225	2.3835	Kiri
2.	0.6474	0.6815	2.2932	Kiri
3.	0.6474	0.7472	2.2031	Kiri
4.	0.6474	0.8182	2.1132	Kiri
5.	0.6474	0.8931	2.0235	Kiri
6.	0.6474	0.9710	1.9341	Kiri
7.	0.6474	1.0512	1.8450	Kiri
8.	0.6266	1.2088	1.6766	Atas
9.	0.6266	1.3004	1.5815	Atas
10.	0.6266	1.3932	1.4870	Atas
:	:	:	:	:
52.	2.9531	1.5799	1.3018	Kanan



Gambar 4.1 Data *Training*

4.1.3 Pengujian Akurasi Hasil Klasifikasi

Data *training* yang telah dibuat pada Tabel 4.4 selanjutnya diuji akurasinya dengan 10 data *testing*. Data yang diklasifikasikan dengan benar dibagi dengan jumlah data testing. Tabel 4.5 adalah data *testing* yang dilakukan dengan nilai $k = 3$. Pada hasil data *testing* dengan $k = 3$, KNN menghasilkan akurasi 100%.

Tabel 4.5 Data *Testing*

Data ke-	Simpangan Atas	Simpangan Kiri	Simpangan Kiri	Kelas	Status
1	1.9750	0.9521	1.9555	kiri	Benar
2	0.9700	1.0026	1.9005	kiri	Benar
3	0.5688	0.8626	2.0601	kiri	Benar
4	1.2159	0.9780	1.9257	kiri	Benar
5	1.2110	2.6120	3.0120	atas	Benar
6	0.6633	1.3920	1.5273	atas	Benar
7	2.1091	0.8738	2.0457	kiri	Benar
8	2.8972	1.5449	1.3790	kanan	Benar
9	2.0331	1.5495	1.3316	kanan	Benar
10	1.1053	2.2227	0.7326	kanan	Benar

Tabel 4.6 menujukkan hasil akurasi dengan menggunakan nilai k yang berbeda. Dengan nilai k yang semakin banyak, akurasi yang dihasilkan menurun. Hal ini membuktikan bahwa nilai k juga mempengaruhi penentuan klasifikasi dalam KNN. Maka dari itu, klasifikasi yang digunakan pada penelitian ini adalah $k = 3$.

Tabel 4.6 Pengujian Akurasi dengan *k* Berbeda

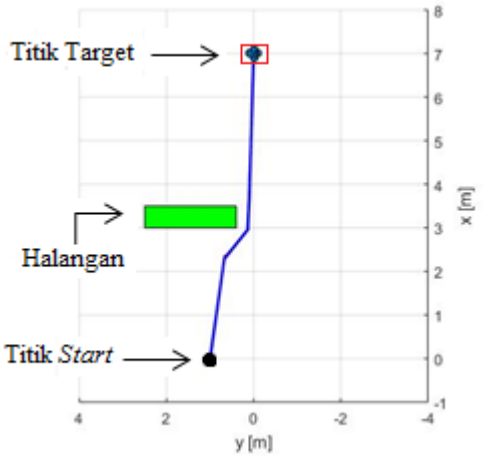
Data Simpangan	k	Akurasi
Data *Testing* Simpangan Kanan, kiri dan atas	5	100%
	7	90%
	9	90%

4.2 Pengujian Algoritma Penghindaran Halangan menggunakan *Machine Learning* pada Halangan statis.

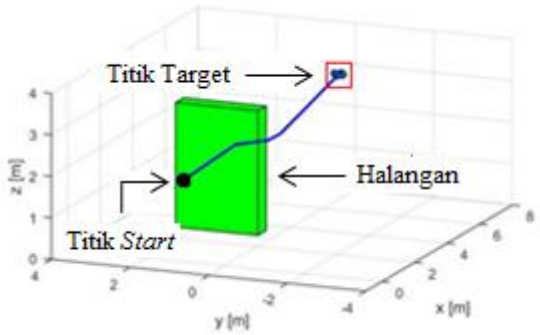
Pengujian performa algoritma yang telah dirancang adalah dengan menguji quadcopter yang harus mencapai titik target dengan beberapa kasus lingkungan 3D yang terdapat halangan statis. Dalam Penghindaran halangan, quadcopter akan menghindar sesuai dengan hasil klasifikasi KNN.

4.2.1 Kasus 1 Halangan Statis

Pada kasus pertama yang dapat dilihat pada Gambar 4.2, titik *start* quadcopter pada koordinat $(0,1,2)$. Sedangkan titik target $(7,0,3)$ dengan halangan 1 halangan statis $(2m \times 0.5m \times 3m)$. Rute awal quadcopter saat tidak ada halangan adalah garis lurus antara titik *start* dan target.

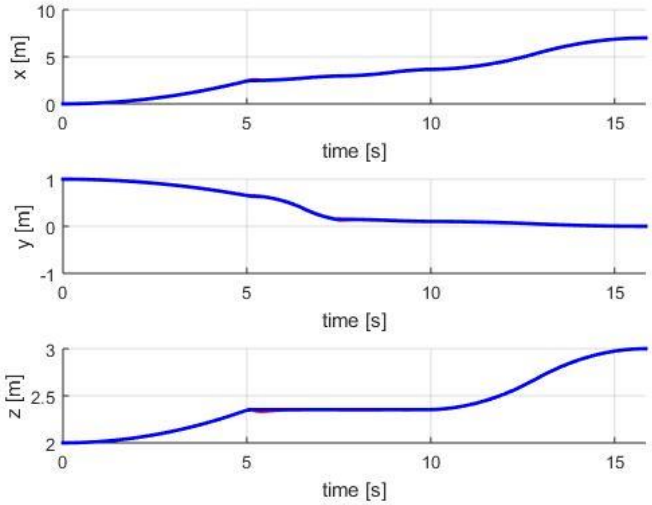


(a)



(b)

Gambar 4.2 Kasus 1 (a) Tampak Atas (b) Tampak Samping



Gambar 4.3 Posisi Quadcopter Terhadap Waktu pada Kasus 1

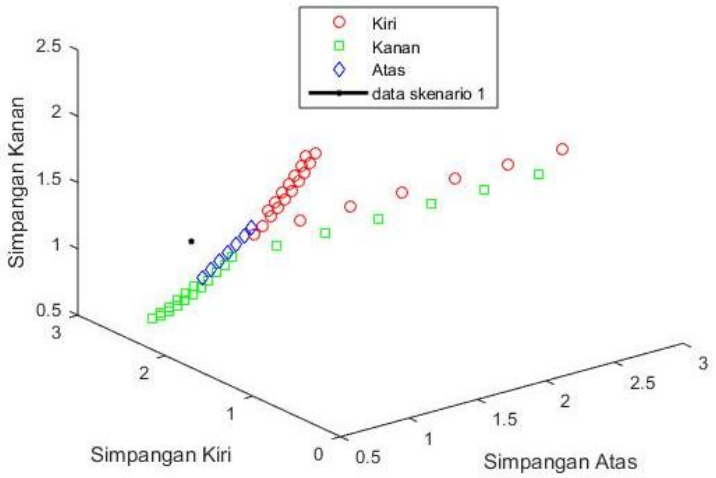
Pertama, quadcopter terbang dari titik *start* menuju titik target. Namun, terdapat halangan terdeteksi saat posisi quadcopter berada pada koordinat $(2.1, 0.4, 2.3)$ atau saat waktu $5s$ yang dapat dilihat pada Gambar 4.3. Posisi quadcopter dan ukuran halangan diketahui sehingga menghasilkan jarak simpangan (simpangan atas $\boldsymbol{\delta_u}$, kiri $\boldsymbol{\delta_l}$ dan kanan $\boldsymbol{\delta_r}$). Masing-masing jarak simpangan adalah $\delta_u = 1.2089$, $\delta_l = 2.8186$, $\delta_r = 0.9163$.

Data tersebut selanjutnya diproses oleh klasifikasi KNN agar menghasilkan arah hindar yang efisien. Posisi data jarak simpangan kasus 1 terhadap data *training* dapat dilihat pada Gambar 4.4. Dengan nilai $k = 3$, data yang terdekat dari data *training* dapat dilihat pada Tabel 4.7. Dari hasil klasifikasi KNN, arah hindar yang efisien adalah ke kanan karena 3 *neighbour* terdekat dengan 52 data *training* (Tabel 4.4) menyatakan 100% kanan dengan energi yang dibutuhkan adalah $0.42\ Joule$.

Arah hindar yang dihasilkan oleh KNN selanjutnya diproses pada *waypoint switching* yang menentukan koordinat arah hindar kanan. *Waypoint switching* berfungsi untuk mengubah koordinat titik target menjadi koordinat arah hindar saat halangan terdeteksi. Setelah menuju koordinat arah hindar, *waypoint switching* mengubah pada titik target semula.

Tabel 4.7 Data *Training* yang Terdekat Kasus 1

Simpangan Atas	Simpangan Kiri	Simpangan Kiri	Kelas	j
0.7110	2.3801	0.6246	Kanan	0.7247
0.7110	2.2890	0.6844	Kanan	0.7629
0.6266	2.3540	0.6412	Kanan	0.7941
0.7110	2.1980	0.7511	Kanan	0.8126
0.6266	2.2564	0.7079	Kanan	0.8358
0.7110	2.1073	0.8230	Kanan	0.8732
0.6266	2.1590	0.7818	Kanan	0.8900
0.7110	2.0168	0.8988	Kanan	0.9439
0.6266	2.0618	0.8609	Kanan	0.9564
0.7110	1.9265	0.9777	Kanan	1.0234

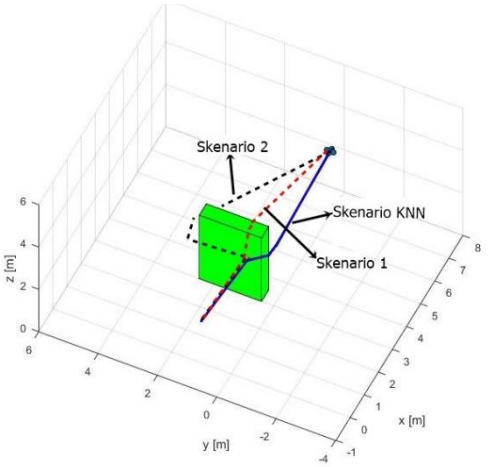


Gambar 4.4 Posisi Data Kasus 1 dengan Data *Training*

Berikut adalah perbandingan jalur yang dihasilkan KNN dan jalur skenario lain, Tabel 4.8 menyatakan perbedaan total jarak dan energi sedangkan keterangan jalur terdapat pada Gambar 4.5.

Tabel 4.8 Perbandingan Jarak dan Energi pada Jalur Kasus 1

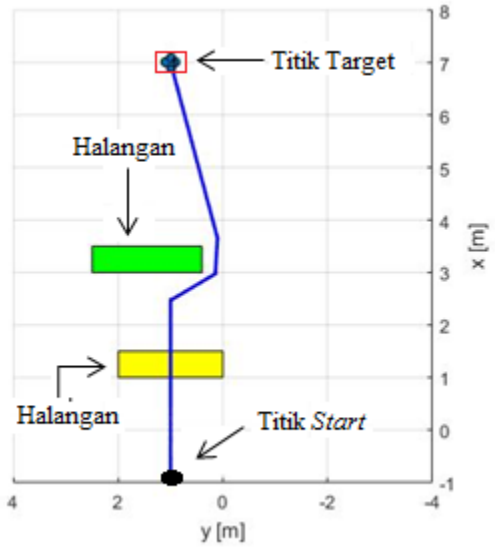
	Total Jarak (m)	Energi ($Joule$)
Skenario KNN	7.2956	19.8312
Skenario 1	9.7432	25.6208
Skenario 2	9.8669	25.8119



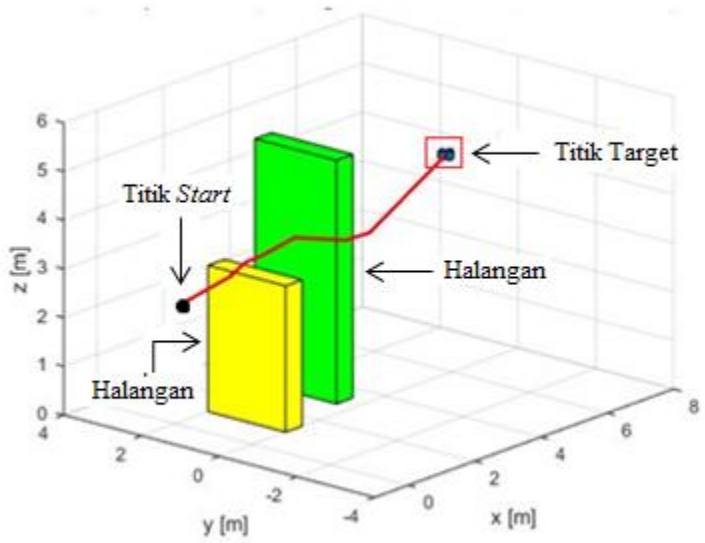
Gambar 4.5 Perbandingan Jalur pada Kasus 1

4.2.2 Kasus 2 Halangan Statis

Pada kasus kedua, terdapat 2 halangan statis yang terdapat diantara titik *start* $(-1,1,2.9)$ dan titik target $(7,1,4)$. Halangan pertama (kuning) berukuran $(2m \times 0.5m \times 3m)$ dan halangan kedua (hijau) dengan ukuran $(2m \times 0.5m \times 5m)$ seperti pada Gambar 4.6.



(a)



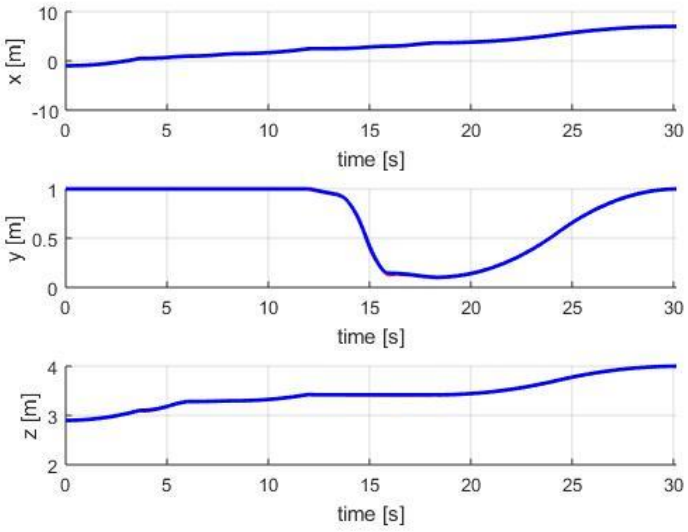
(b)

Gambar 4.6 Kasus 2 (a) Tampak Atas (b) Tampak Samping

Pada destinasi quadcopter dari titik *start* menuju titik target, dapat diamati pada Gambar 4.7 bahwa quadcopter mendeteksi halangan pertama (kuning) pada waktu $4s$ yaitu pada koordinat $(0{,}1{,}3.1)$. Pada waktu $14s$, quadcopter pada koordinat $(7{,}0.8{,}3.5)$ mendeteksi halangan kedua (hijau).

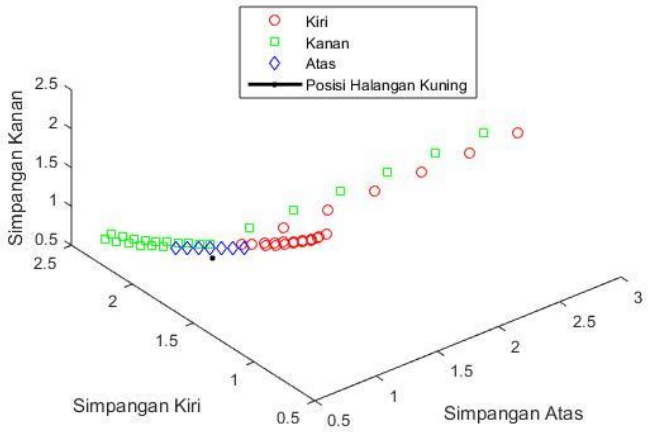
Saat halangan pertama (kuning) terdeteksi, jarak masing-masing simpangan (simpangan atas $\boldsymbol{\delta_u}$, kiri $\boldsymbol{\delta_l}$ dan kanan $\boldsymbol{\delta_r}$) antara quadcopter dan halangan dapat dihitung dengan informasi koordinat posisi quadcopter dan ukuran halangan, yaitu $\delta_u = 1.2089$, $\delta_l = 2.8186$, $\delta_r = 0.9163$. Begitu juga pada halangan kedua (hijau), mempunyai jarak simpangan $\delta_u = 1.9467$, $\delta_l = 1.9204$, $\delta_r = 0.9826$ antara quadcopter dan halangan.

Pada proses klasifikasi KNN, data masing-masing jarak simpangan tersebut dibandingkan dengan 52 data *training* (Tabel 4.4) sehingga menghasilkan arah hindar yang efisien. Karena nilai $k = 3$ pada KNN, maka 3 dari 52 data yang mempunyai jarak terkecil antara data simpangan pada kasus 2 dengan data *training* yang akan menentukan arah hindar yang efisien.

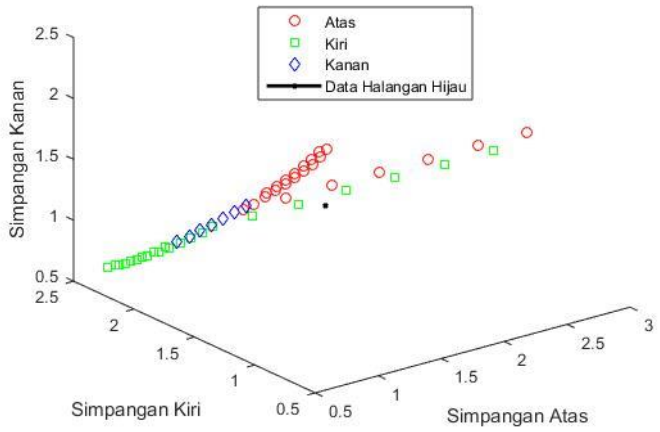


Gambar 4.7 Posisi Quadcopter terhadap Waktu pada Kasus 2

Posisi data jarak simpangan terhadap data *training* dapat dilihat pada Gambar 4.8. Pada Tabel 4.9 dapat dilihat bahwa pada halangan pertama (kuning) arah hindar yang efisien adalah ke atas karena 3 data yang terpilih menunjukkan 100% ke atas dengan energi yang dibutuhkan 2.57 $Joule$. Sedangkan pada halangan kedua (hijau), arah hindar yang efisien yaitu ke kanan dengan hasil KNN 100% ke atas dan memerlukan energi 0.48 $Joule$.



(a)



(b)

Gambar 4.8 Posisi Data Kasus 2 dengan Data *Training* (a) Halangan Pertama/kuning (b) Halangan Kedua/Hijau

Tabel 4.9 Data *Training* yang Terdekat Kasus 2

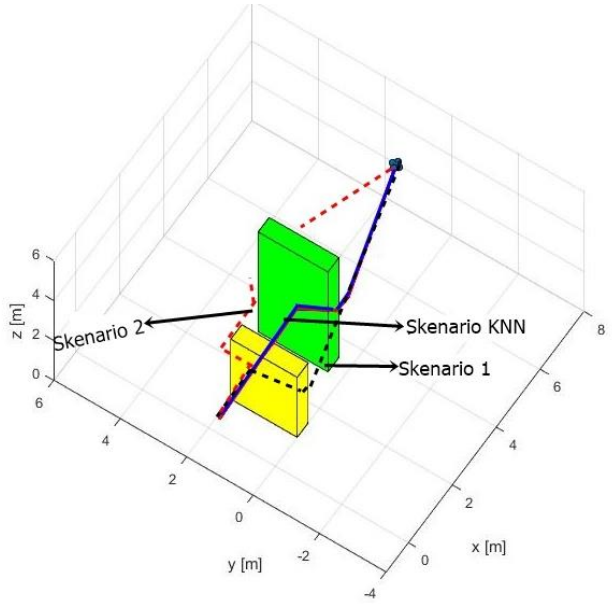
Halangan	Simpangan Atas	Simpangan Kiri	Simpangan Kiri	Kelas	j
	0.6266	1.4870	1.3932	Atas	0.1110
	0.6266	1.3932	1.4870	Atas	0.1110
	0.6266	1.5815	1.3004	Atas	0.2181
	0.6266	1.3004	1.5815	Atas	0.2181
Halangan	0.7111	1.5692	1.3122	Kanan	0.2447
Pertama	0.7111	1.3122	1.5692	Kiri	0.2447
(kuning)	0.6266	1.6766	1.2088	Atas	0.3431
	0.6266	1.2088	1.6766	Atas	0.3431
	0.7111	1.6578	1.2265	kanan	0.3465
	0.7111	1.2265	1.6578	Kiri	0.3465
	1.7821	1.5799	1.3018	kanan	0.4948
	2.1689	1.5799	1.3018	kanan	0.5169
	1.4028	1.5799	1.3018	kanan	0.7166
	2.5598	1.5799	1.3018	kanan	0.7705
Halangan	1.7821	1.3018	1.5799	Kiri	0.8755
Kedua	2.1689	1.3018	1.5799	Kiri	0.8881
(hijau)	1.4028	1.3018	1.5799	Kiri	1.0174
	1.0390	1.5799	1.3018	kanan	1.0206
	2.5598	1.3018	1.5799	Kiri	1.0560
	2.9531	1.5799	1.3018	kanan	1.1093

Tujuan dari kasus ini adalah untuk membuktikan algoritma yang telah dirancang dapat menghindari halangan lebih dari 1. Pada Gambar 4.6 terlihat bahwa quadcopter dapat menghindari halangan lebih dari 1 dengan arah hindar yang memiliki efisiensi energi dan berhasil mencapai titik target.

Pada pengujian pada lingkungan yang terdapat lebih dari 1 halangan, jalur yang dihasilkan KNN juga dibandingkan dengan skenario lain. Tabel 4.10 menyatakan perbedaan jarak dan energi pada setiap jalur (Gambar 4.9) pada kasus 2. Terlihat bahwa Jalur yang dihasilkan KNN adalah yang paling minimal.

Tabel 4.10 Perbandingan Jarak dan Energi pada Jalur Kasus 2

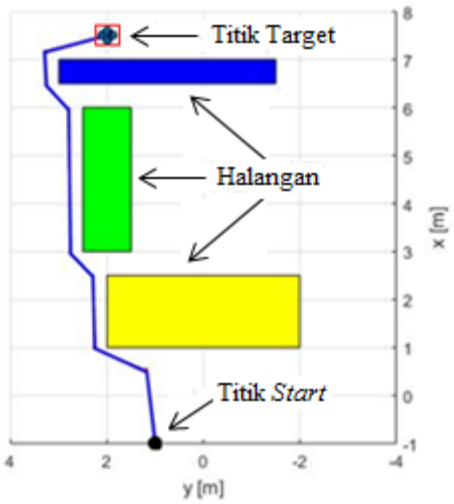
	Total Jarak (m)	Energi ($Joule$)
Skenario KNN	8.7003	20.0553
Skenario 1	8.5113	31.1089
Skenario 2	10.801	22.7423



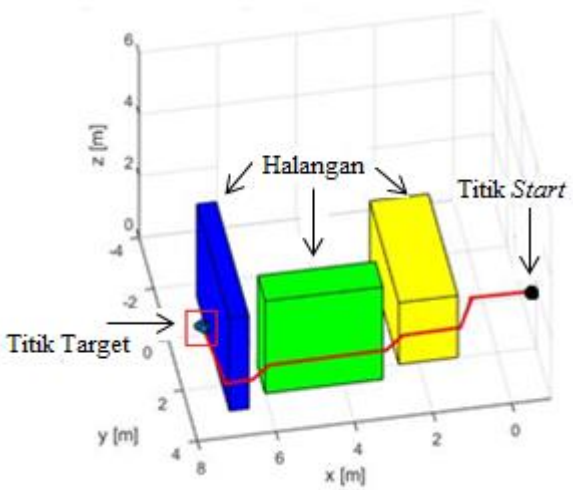
Gambar 4.9 Perbandingan Jalur pada Kasus 2

4.2.3 Kasus 3 Halangan Statis

Pengujian halangan statis kasus 3 berfungsi untuk melihat performa algoritma yang diajukan jika terdapat halangan statis dengan ukuran yang bervariasi. Dapat dilihat pada Gambar 4.10, terdapat 3 halangan diantara titik *start* $(-1,1,1)$ dan titik target $(7.5,2,2.2)$.

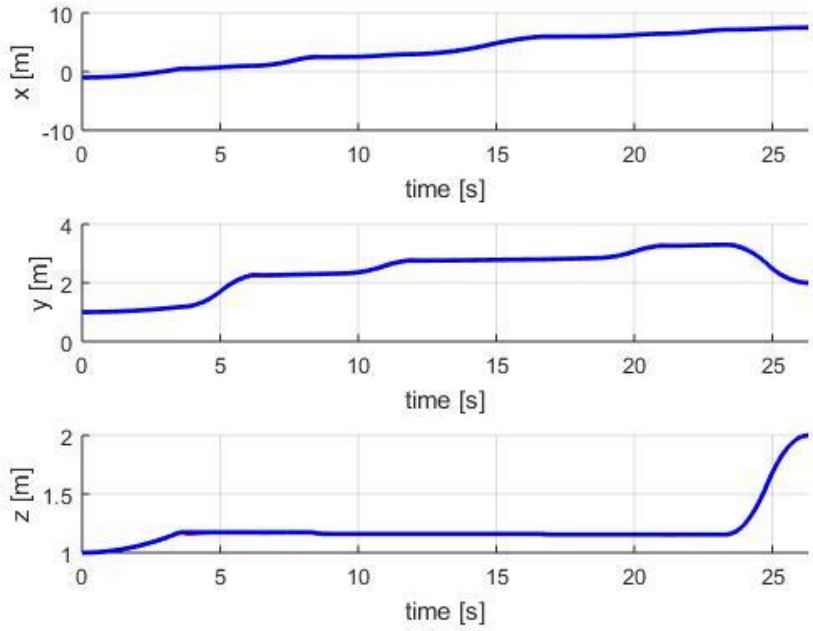


(a)



(b)

Gambar 4.10 Kasus 3 (a) Tampak Atas (b) Tampak Samping



Gambar 4.11 Posisi Quadcopter terhadap Waktu

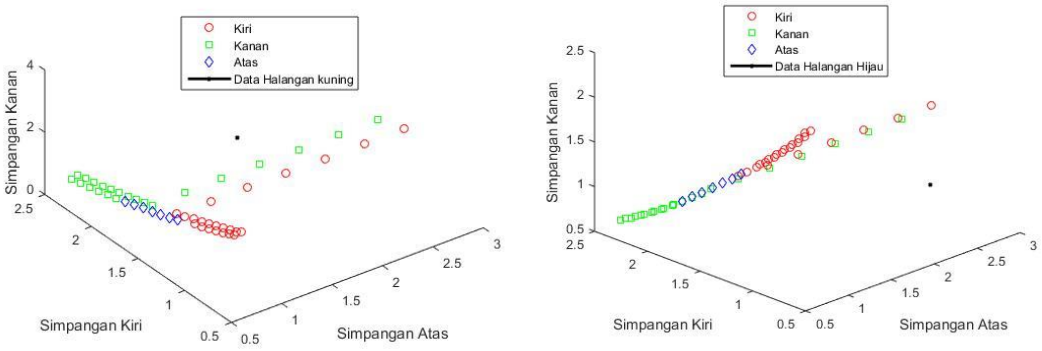
Pada Grafik posisi quadcopter terhadap waktu Gambar 4.11, quadcopter memerlukan waktu $27s$ untuk terbang dari titik *start* menuju titik target. Saat waktu $3s$, quadcopter pada koordinat $(0.7, 1.6, 1.2)$ mendeteksi halangan pertama (kuning). Selanjutnya, quadcopter juga mendeteksi halangan kedua (hijau) pada $8.5s$ saat koordinat $(2.8, 3, 1.15)$ dan saat $17s$ pada koordinat $(6, 2.8, 1.15)$ mendeteksi halangan ketiga (Biru).

Saat quadcopter mendeteksi halangan, secara otomatis sistem akan menghitung data jarak masing-masing simpangan (simpangan atas $\boldsymbol{\delta_u}$, kiri $\boldsymbol{\delta_l}$ dan kanan $\boldsymbol{\delta_r}$) antara posisi quadcopter dan halangan. Maka pada halangan pertama (kuning) memiliki $\delta_u = 1.2379$, $\delta_l = 1.2368$, $\delta_r = 3.5141$, halangan kedua (hijau) $\delta_u = 2.1973$, $\delta_l = 0.6937$, $\delta_r = 1.2259$ dan halangan ketiga (biru) $\delta_u = 2.2031$, $\delta_l = 0.6798$, $\delta_r = 4.6672$.

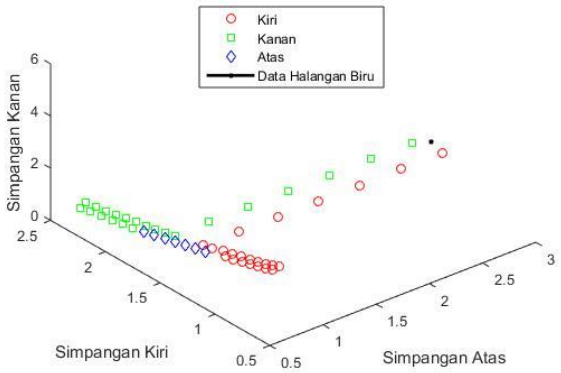
Data jarak simpangan 3 halangan tersebut masing-masing akan dibandingkan dengan data *training* pada KNN. Sama dengan pengujian kasus sebelumnya, data *training* yang memiliki jarak terdekat dengan data simpangan pada kasus 3 akan menghasilkan keputusan arah hindar yang efisien. Pada Gambar 4.12 terdapat posisi data jarak simpangan yang diuji pada kasus 3

terhadap data *training*. Perhitungan 3 *neighbour* terdekat ($k = 3$) pada data *training* yang dapat menghasilkan keputusan arah hindar yang efisien dapat dilihat pada Tabel 4.11.

Keputusan arah hindar efisien yang dihasilkan KNN menyatakan bahwa arah hindar ke kiri adalah arah hindar yang efisien untuk semua halangan pada kasus 3. Pada halangan pertama (kuning), energi yang dibutuhkan untuk menghindar sebesar 0.76 $Joule$. Sedangkan halangan kedua (hijau) membutuhkan 0.25 $Joule$ dan ketiga (biru) 0.23 $Joule$.



(a) (b)



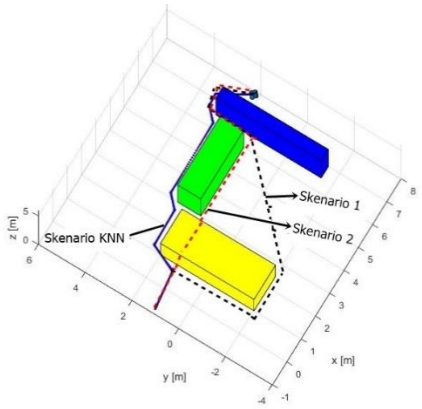
(c)

Gambar 4.12 Posisi Data Kasus 3 dengan Data *Training* (a) Halangan Pertama/kuning (b) Halangan Kedua/Hijau (c) Halangan Ketiga/Biru

Tabel 4.11 Data *Training* yang Terdekat Kasus 3

Halangan	Simpangan Atas	Simpangan Kiri	Simpangan Kiri	Kelas	j
Halangan Pertama (kuning)	0.7149	0.6225	2.3835	Kiri	1.3889
	0.6474	0.6225	2.3835	Kiri	1.4156
	0.7149	0.6815	2.2932	Kiri	1.4395
	0.6474	0.6815	2.2932	Kiri	1.4654
	0.7149	0.7472	2.2031	Kiri	1.4939
	0.6474	0.7472	2.2031	Kiri	1.5188
	0.7149	0.8182	2.1132	Kiri	1.5527
	0.6474	0.8182	2.1132	Kiri	1.5768
	0.7149	0.8931	2.0235	Kiri	1.6166
	0.6474	0.8931	2.0235	Kiri	1.6396
Halangan Kedua (hijau)	2.1689	1.3018	1.5799	Kiri	0.7042
	2.5598	1.3018	1.5799	Kiri	0.7915
	1.7821	1.3018	1.5799	Kiri	0.8170
	2.1689	1.5799	1.3018	Kanan	0.8898
	2.5598	1.5799	1.3018	Kanan	0.9604
	1.7821	1.5799	1.3018	Kanan	0.9815
	2.9531	1.3018	1.5799	Kiri	1.0326
	1.4028	1.3018	1.5799	Kiri	1.0612
	2.9531	1.5799	1.3018	Kanan	1.1671
	1.4028	1.5799	1.3018	Kanan	1.1926
Halangan Ketiga (hijau)	0.7149	0.6225	2.3835	Kiri	2.7264
	0.6474	0.6225	2.3835	Kiri	2.7638
	0.7149	0.6815	2.2932	Kiri	2.8018
	0.6474	0.6815	2.2932	Kiri	2.8383
	0.7149	0.7472	2.2031	Kiri	2.8794
	0.6474	0.7472	2.2031	Kiri	2.9148
	0.7149	0.8182	2.1132	Kiri	2.9591
	0.6474	0.8182	2.1132	Kiri	2.9937
	0.7149	0.8931	2.0235	Kiri	3.0412
	0.6474	0.8931	2.0235	Kiri	3.0748

Pada kasus 3, sistem yang dirancang dapat mencapai titik target tanpa menabrak halangan meskipun ukuran halangan yang bervariasi. Sistem yang dirancang tetap menyesuaikan dengan ukuran halangan yang bervariasi dengan mengatur *waypoint switching* setelah mencapai titik hindar, *waypoint* selanjutnya adalah koordinat akhir dari ketebalan halangan lalu kembali ke *waypoint* titik target.



Gambar 4.13 Perbandingan Jalur pada Kasus 3

Pengujian kasus 3 ini, jalur yang dihasilkan KNN dibandingkan dengan jalur yang lain. Terlihat bahwa Jalur yang dihasilkan KNN adalah yang paling minimal dapat dilihat pada Tabel 4.12 yang menyatakan perbedaan jarak dan energi pada setiap skenario (Gambar 4.13) pada kasus 3.

Tabel 4.12 Perbandingan Jarak dan Energi pada Jalur Kasus 3

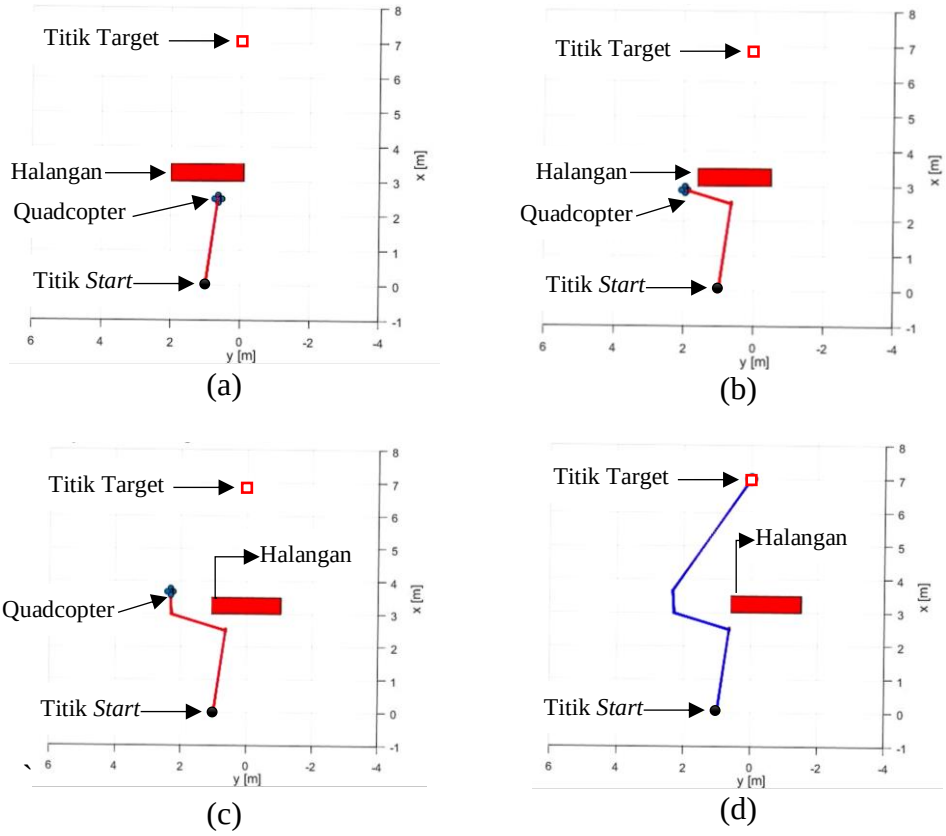
	Total Jarak (m)	Energi ($Joule$)
Skenario KNN	10.588	19.1248
Skenario 1	15.480	38.9753
Skenario 2	11.001	21.6762

4.3 Pengujian Algoritma Penghindaran Halangan menggunakan *Machine Learning* pada Halangan Dinamis

Selain pengujian pada halangan statis pada kasus 1-3, sistem yang dirancang pada penelitian ini juga diuji di lingkungan 3D dengan halangan dinamis atau bergerak. Saat halangan terdeteksi, quadcopter menghindar sesuai dengan keputusan yang dihasilkan KNN.

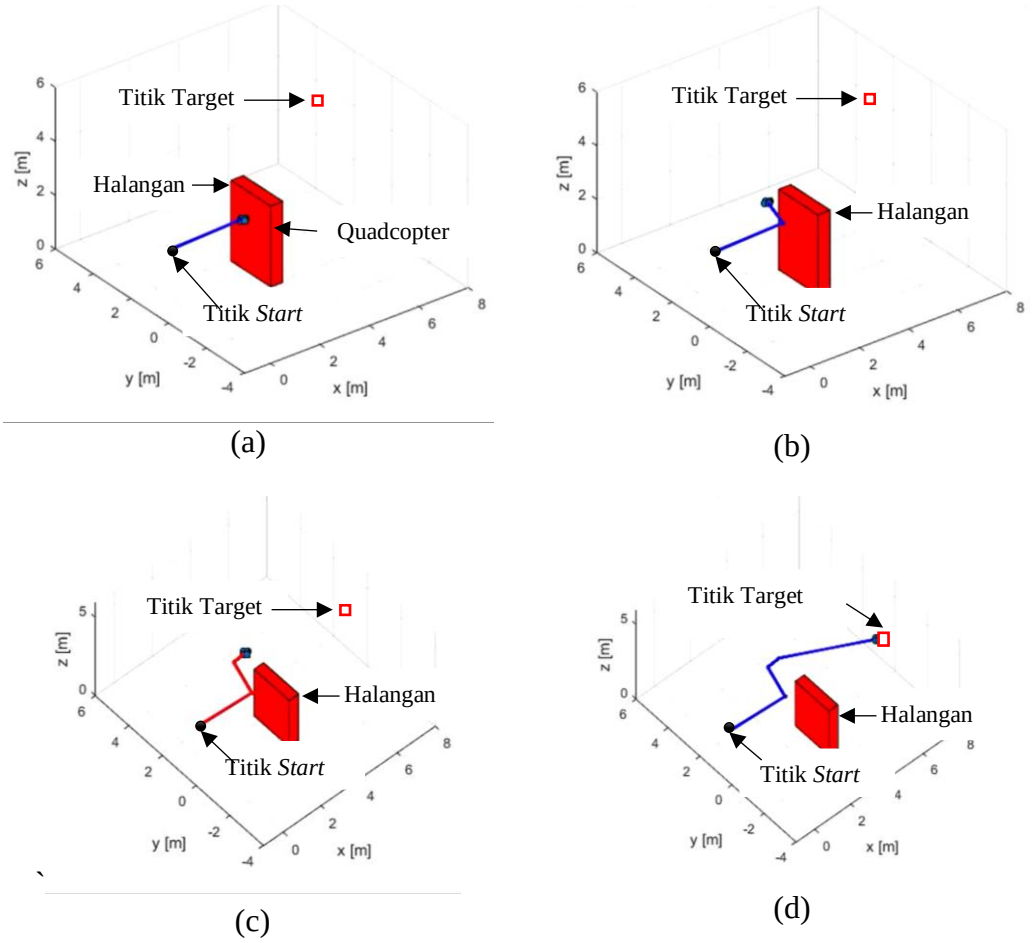
4.3.1 Kasus 4 Halangan Dinamis

Pada kasus 4, lingkungan sama dengan kasus 1 yaitu titik *start* dengan koordinat (0,1,2) dan titik target (7,0,3). Namun, halangan dengan ukuran ($2m \times 0.5m \times 3m$) pada kasus 1 bersifat statis sedangkan pada kasus 4 bersifat dinamis. Halangan tersebut mempunyai kecepatan $0.005\ m/s$ dan bergerak pada arah sumbu y negatif. Dari hasil pengujian ini, quadcopter dapat mencapai titik target tanpa menabrak halangan seperti pada Gambar 4.14 dan 4.15.



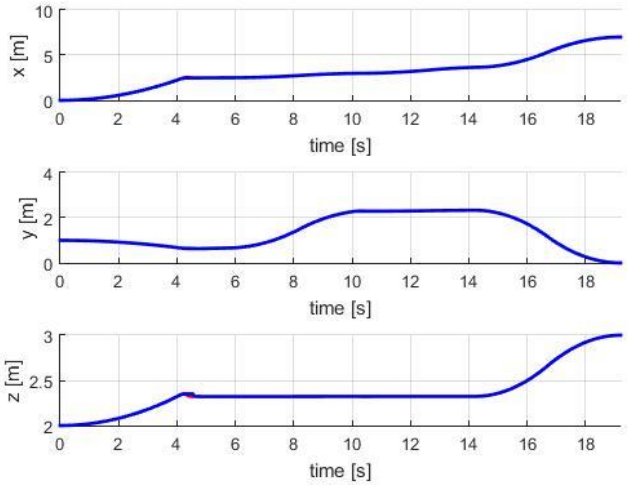
Gambar 4.14 Kasus 4 Tampak Atas (a) 3s (b) 6s (c) 9s (d) 12s

Total waktu tempuh destinasi quadcopter dari titik *start* sampai tutuk target adalah $12s$. Pada Gambar 4.11 dan 4.12, menunjukkan hasil simulasi pada kasus 4 saat $3s$, $6s$, $9s$ dan $12s$. Dapat terlihat bahwa terjadi pegerakan halangan pada sumbu y negatif yang ditandai dengan perubahan posisi halangan tersebut.



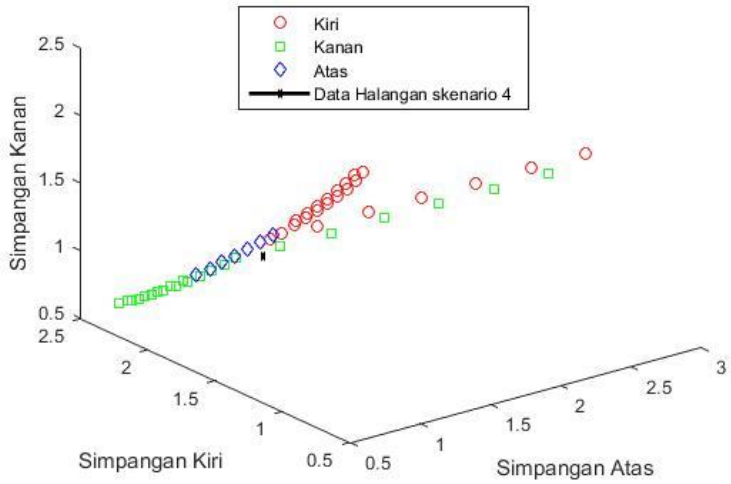
Gambar 4.15 Kasus 4 Tampak Samping (a) 3s (b) 6s (c) 9s (d) 12s

Saat quadcopter terbang dari titik *start* menuju titik target, terdapat halangan terdeteksi saat posisi quadcopter berada pada koordinat $(2.1, 0.4, 2.3)$ atau saat waktu $3s$ yang dapat dilihat pada Gambar 4.16. Pada waktu $3s$ tersebut, sistem menghitung jarak masing-masing simpangan (simpangan atas $\boldsymbol{\delta_u}$, kiri $\boldsymbol{\delta_l}$ dan kanan $\boldsymbol{\delta_r}$) antara quadcopter dengan halangan. Masing-masing jarak simpangannya adalah $\delta_u = 1.1024$, $\delta_l = 1.7788$, $\delta_r = 1.1179$.



Gambar 4.16 Posisi Quadcopter terhadap Waktu Kasus 4

Hasil perhitungan masing-masing jarak simpangan tersebut dibandingkan dengan data *training* sebagai proses klasifikasi arah hindar berdasarkan KNN untuk menghasilkan arah hindar yang efisien. Posisi data simpangan pada kasus 4 terhadap data *training* dapat dilihat pada Gambar 4.17. Data *training* yang terdekat menentukan keputusan arah hindar yang efisien.



Gambar 4.17 Posisi Data Kasus 4 dengan Data *Training*

Tabel 4.13 Data *Training* yang Terdekat Kasus 4

Simpangan Atas	Simpangan Kiri	Simpangan Kiri	Kelas	j
1.0390	1.5799	1.3018	Kanan	1.0390
0.7110	1.7470	1.1420	Kanan	0.7110
0.7110	1.8366	1.0589	Kanan	0.7110
1.4028	1.5799	1.3018	Kanan	1.4028
0.7111	1.6578	1.2265	Kanan	0.7111
0.7110	1.9265	0.9777	Kanan	0.7110
0.6266	1.7723	1.1185	Atas	0.6266
0.7111	1.5692	1.3122	Kanan	0.7111
0.6266	1.8685	1.0301	Kanan	0.6266
0.6266	1.6766	1.2088	Atas	0.6266

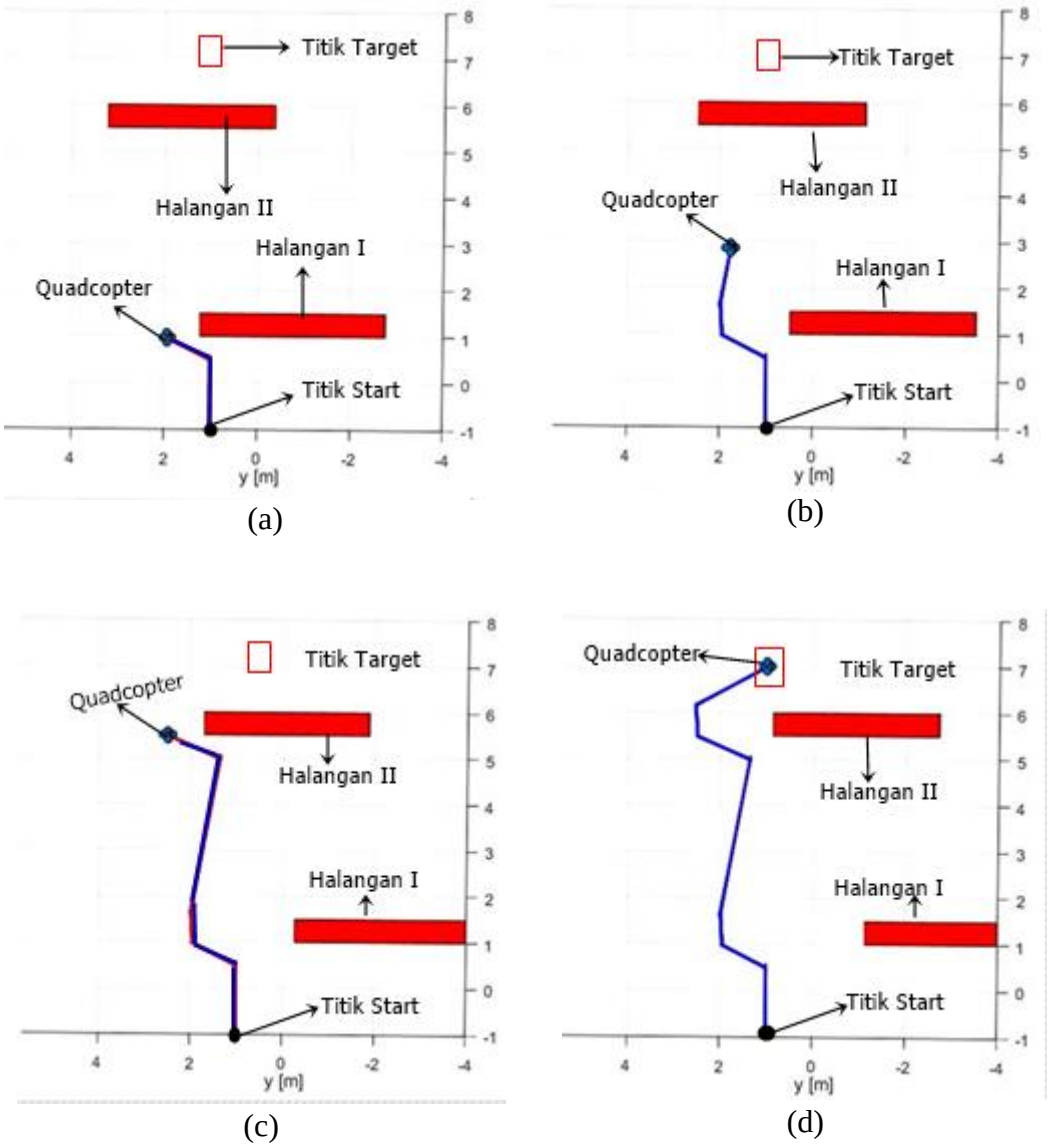
Pada kasus 4, arah hindar efisien hasil keputusan KNN adalah ke kiri. Seperti yang terlihat pada Tabel 4.13, 3 data terdekat menyatakan 100 % kiri. Quadcopter memerlukan energi sebesar $0.62\,Joule$ yang dibutuhkan untuk menghindar ke kiri.

Namun, ternyata halangan bergerak sehingga jarak masing-masing simpangan berubah. Pada saat itu, sistem melakukan prediksi titik potong antara rute halangan dengan arah hindar quadcopter. Pada kasus 4, terdapat titik potong antara rute halangan (ke kanan) dan arah hindar quadcopter (ke kanan). Maka, sistem akan merubah arah hindar quadcopter yang sebelumnya ke kanan menjadi ke kiri untuk menghindari halangan.

Sistem yang dirancang pada penelitian ini dapat berjalan dan tetap dapat menghasilkan arah hindar yang efisien meskipun halangan yang terdeteksi berpindah posisi setiap waktu. Selanjutnya, sistem akan diuji dengan lingkungan yang terdapat halangan dinamis lebih dari 1 pada kasus 5.

4.3.2 Kasus 5 Halangan Dinamis

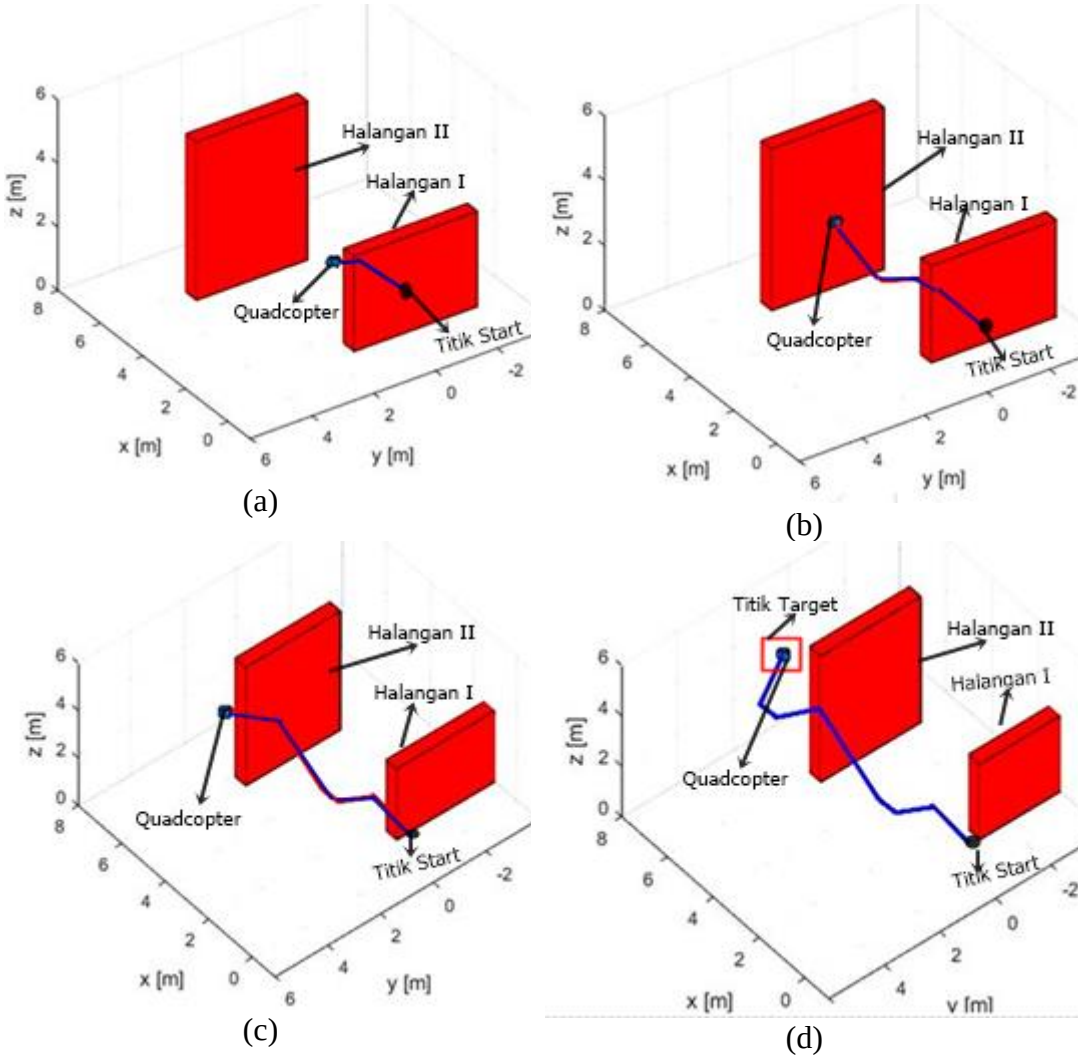
Pengujian pada kasus 5 berfungsi untuk menunjukkan performa sistem yang dirancang dapat menghindari halangan dinamis lebih dari 1. Lingkungan pada kasus ini (Gambar 4.18 dan 4.19) dengan titik *start* $(-1,1,2.9)$ dan titik target $(7,1,4)$. Terdapat 2 halangan, yaitu halangan pertama yang memiliki ukuran $(3.6\ m\ x\ 0.5\ m\ x\ 5\ m)$ dan halangan kedua $(4\ m\ x\ 0.5\ m\ x\ 3m)$. Kedua halangan tersebut bersifat dinamis dengan pergerakan pada sumbu y negatif dengan kecepatan $0.005m/s$.



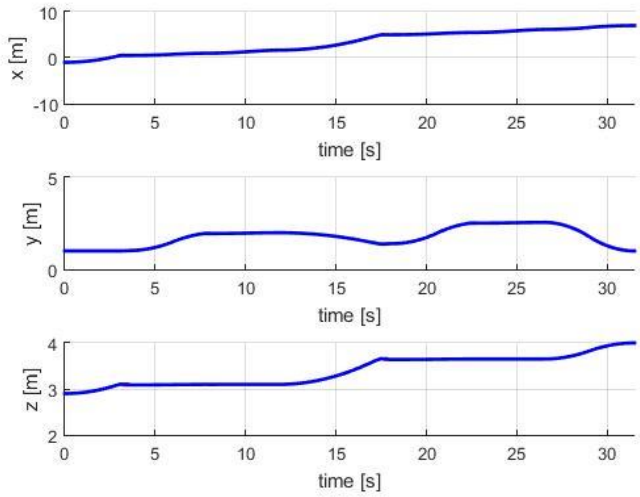
Gambar 4.18 Kasus 5 Tampak Atas (a) 5s (b) 10s (c) 15s (d) 20s

Pada kasus 5, quadcopter memerlukan waktu tempuh $20s$ sampai menuju titik target dengan mengindari 2 halangan statis. Pergerakan kedua halangan dapat terlihat pada Gambar 4.18 dan 4.19 saat $5s, 10s, 15s$ dan $20s$ dimana posisi halangan berubah setiap waktu.

Pada grafik posisi quadcopter terhadap waktu Gambar 4.20 terlihat bahwa saat quadcopter terbang dari titik *start* menuju titik target, terdapat halangan I pada waktu $3s$ yaitu pada koordinat (0,1,3.1). Pada waktu $18s$, quadcopter pada koordinat (5,1,3.8) mendeteksi halangan II.

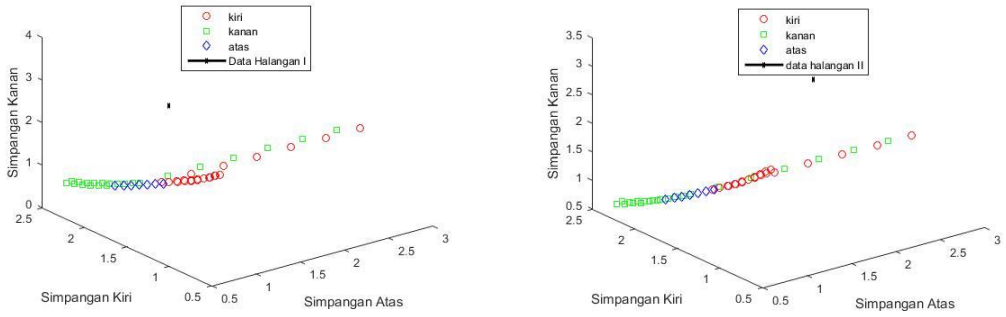


Gambar 4.19 Kasus 5 Tampak Samping (a) 5s (b) 10s (c) 15s (d) 20s



Gambar 4.20 Posisi Quadcopter terhadap Waktu Kasus 5

Sistem secara otomatis akan menghitung jarak masing-masing simpangan (simpangan atas $\boldsymbol{\delta_u}$, kiri $\boldsymbol{\delta_l}$ dan kanan $\boldsymbol{\delta_r}$) antara posisi quadcopter dan halangan saat quadcopter mendeteksi halangan. Pada halangan I memiliki $\delta_u = 0.5440$, $\delta_l = 1.0551$, $\delta_r = 3.7052$ dan halangan II $\delta_u = 1.7352$, $\delta_l = 1.1977$, $\delta_r = 3.1529$. Arah hindar yang efisien dihasilkan oleh KNN dengan perbandingan masing-masing jarak simpangan dengan data *training*. Posisi data simpangan pada kasus 5 terhadap data *training* dapat dilihat pada Gambar 4.21.



(a) (b)

Gambar 4.21 Posisi Data Kasus 5 dengan Data *Training* (a) Halangan Pertama (b) Halangan Kedua

Tabel 4.14 Data *Training* yang Terdekat Kasus 5

Halangan	Simpangan Atas	Simpangan Kiri	Simpangan Kiri	Kelas	j
	0,6474	0,6225	2,3835	kiri	1,394534048
	0,7149	0,6225	2,3835	kiri	1,401156829
	0,6474	0,6815	2,2932	kiri	1,464244693
	0,7149	0,6815	2,2932	kiri	1,470553559
Halangan	0,6474	0,7472	2,2031	kiri	1,536814361
Pertama	0,7149	0,7472	2,2031	kiri	1,542826507
	0,6474	0,8182	2,1132	kiri	1,612847535
	0,7149	0,8182	2,1132	kiri	1,618577283
	0,6474	0,8931	2,0235	kiri	1,692645991
	0,7149	0,8931	2,0235	kiri	1,698106504
	0,7149	0,6225	2,3835	kiri	1,401372003
	0,7149	0,6815	2,2932	kiri	1,43057982
	0,6474	0,6225	2,3835	kiri	1,451254712
	0,7149	0,7472	2,2031	kiri	1,464951323
Halangan	0,6474	0,6815	2,2932	kiri	1,479478074
Kedua	0,7149	0,8182	2,1132	kiri	1,505326685
	0,6474	0,7472	2,2031	kiri	1,51273895
	0,6474	0,8182	2,1132	kiri	1,551871509
	0,7149	0,8931	2,0235	kiri	1,552204114
	1,7821	1,3018	1,5799	kiri	1,577138364

Arah hindar yang dihasilkan KNN pada kasus 5 berbeda dengan kasus 2 meskipun ukuran halangan sama. Posisi quadcopter terhadap halangan pada kasus 5 berbeda dengan skenerio 2 karena terjadi pergerakan kedua halangan sehingga posisi halangan juga berubah.

Pada kasus 5, arah hindar efisien hasil keputusan KNN adalah ke kiri. Seperti yang terlihat pada Tabel 4.14, 3 data terdekat menyatakan 100 % kiri. Quadcopter memerlukan energi sebesar $0.43\,Joule$ yang dibutuhkan untuk

menghindar ke kiri. Sedangkan pada halangan II, 3 data terdekat juga menyatakan 100% ke kiri dengan energi yang dibutuhkan 0.17 $Joule$.

Sistem yang dirancang pada penelitian ini dapat berhasil menghasilkan arah hindar yang efisien saat quadcopter mendeteksi halangan pada kasus 5 yang terdapat halangan dinamis lebih dari 1. Quadcopter dapat mencapai titik target meskipun terdapat halangan dinamis.

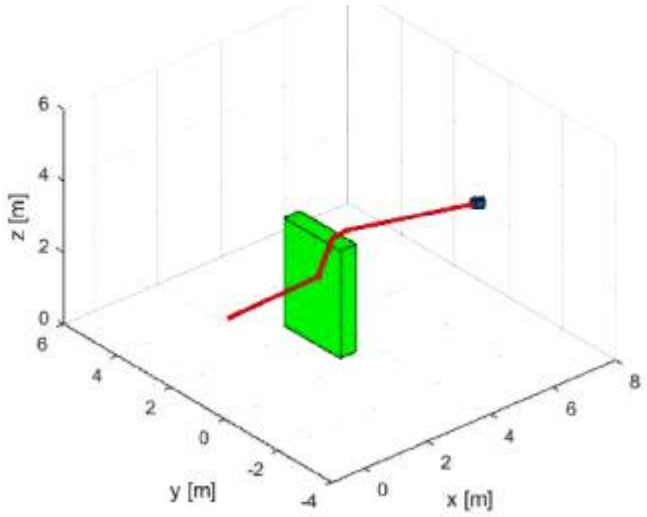
4.4 Evaluasi

Perbandingan dari hasil penelitian sebelumnya dengan hasil desain tesis ini ditunjukkan dengan hasil keputusan yang dihasilkan pada suatu kasus yang sama. Dapat dilihat pada Gambar 4.22 dan 4.23 bahwa terjadi perbedaan keputusan antara metode *hybrid path planning* yang mempertimbangkan tinggi halangan dalam memilih keputusan. Jika tinggi halangan $<$ 150 m, maka quadcopter harus menghindar ke atas sedangkan apabila tinggi halangan $\geq$ 150 m quadcopter menghindar ke samping [2]. Pada desain tesis ini, pertimbangan posisi quadcopter terhadap halangan dan efisiensi masing-masing arah hindar (kanan, kiri dan atas) digunakan untuk pemilihan keputusan. Pada Tabel 4.15 menunjukkan bahwa total jarak dan energi yang dihasilkan desain tesis ini lebih kecil daripada metode *hybrid path planning*. Namun, karena desain pada tesis ini menggunakan *machine learning* KNN pada pemilihan keputusan, waktu komputasi (waktu yang dibutuhkan dalam pengambilan keputusan) juga lebih lama daripada metode *hybrid path planning*.

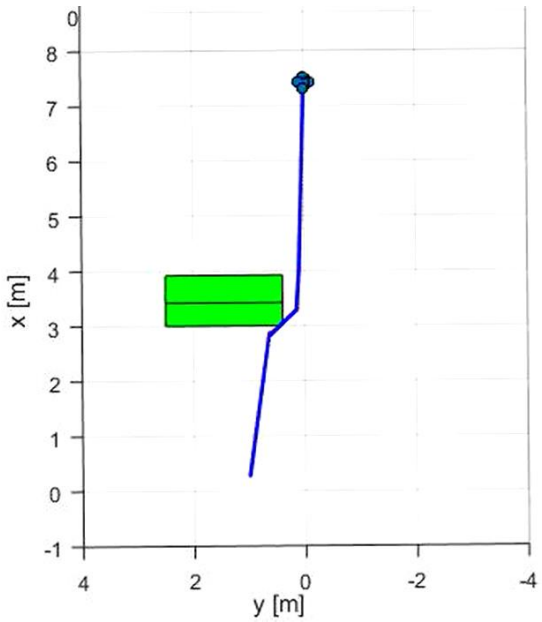
Tabel 4.15 Perbandingan Hasil

	Total Jarak (m)	Energi ($Joule$)	Waktu komputasi (t)
Hasil Desain	7.2956	19.8312	0.006898
Metode *hybrid path planning*	9.7432	25.6208	0.000096

Pemilihan keputusan menggunakan *machine learning* juga bisa digunakan menggunakan RVM (*Relevance Vector Machine*) [4]. Namun waktu komputasi yang dibutuhkan RVM lebih besar yaitu $1.875985\ s$ dibandingkan KNN yang hanya memerlukan $0.006898\ s$.



Gambar 4.22 Hasil Metode *Hybrid Path Planning*



Gambar 4.23 Hasil Desain

BAB 5
PENUTUP

5.1 Kesimpulan

Tesis ini membahas tentang desain sistem navigasi pada quadcopter untuk menghindari halangan pada ruang 3D. Desain sistem navigasi dirancang untuk menghasilkan arah hindar yang efisien, yaitu dengan meminimalkan energi dan jarak untuk menghindar. KNN (*K-Nearest-Neighbour*), salah satu algoritma metode *machine learning* digunakan untuk menghasilkan keputusan arah hindar yang efisien. Quadcopter pada penelitian ini menggunakan kontroler *Proportional-Derivative* untuk mencapai *waypoint* yang diinginkan.

Dalam algoritma KNN, perlu adanya data *training* dan jumlah *neighbour* k yang ditentukan untuk proses klasifikasi. Pada Tesis ini, data *training* ditentukan berdasarkan percobaan dengan posisi quadcopter dan halangan yang berbeda-beda sampai 52 data. Jumlah k yang digunakan adalah 3. Dari hasil pengujian akurasi KNN yang telah dirancang menunjukkan 100% sesuai, yang telah dibuktikan pada proses data *testing*. Simulasi pada penelitian ini menunjukkan bahwa desain sistem yang dirancang dapat membuat quadcopter mencapai titik target tanpa menabrak halangan statis dan dinamis dengan ukuran yang bervariasi. Quadcopter juga berhasil menghindari halangan dengan arah hindar yang efisien. Namun, masih ada kemungkinan quadcopter menabrak halangan dinamis yang mempunyai kecepatan tertentu.

5.2 Saran

Pada penelitian selanjutnya, penulis menyarankan untuk menambahkan algoritma prediksi kecepatan pada halangan dinamis. Dengan begitu, quadcopter mampu menghindari halangan dinamis yang memiliki kecepatan bervariasi.

Halaman ini sengaja dikosongkan

DAFTAR PUSTAKA

[1] L. Zhixiang, C. Laurent, Y. Chi, Z. Youmin and D. Theilliol, "Path following control of unmanned quadrotor," in *International Conference onUnmanned Aircraft Systems (ICUAS)*, Miami, Florida, United States, 2017.

[2] D. Ortiz-Arroyo, "A Hybrid 3D Path Planning Method for UAVs," in *2015 Workshop on Research, Education and Development of Unmanned Aerial Systems (RED-UAS)*, Cancun, Mexico, 2015.

[3] Z. Ahmad, F. Ullah, C. Tran, Lee and L. Sungchang, "Efficient Energy Flight Path Planning Algorithm Using 3-D Visibility Roadmap for Small Unmanned Aerial Vehicle," *Hindawi International Journal of Aerospace Engineering*, p. Article ID 2849745, 2017.

[4] Y. Hongshan, Z. Jiang, W. Yaonan, J. Wenyan, S. Mingui and T. Yandong, "Obstacle Classification and 3D Measurement in Unstructured Environments Based on ToF Cameras," *MDPI*, vol. 14, pp. 10753-10782, 2014.

[5] K. Chee and Z. Zhong, "Control, Navigation and Collision Avoidance for an Unmanned Aerial Vehicle," *Elsevier*, pp. 66-76, 2012.

[6] T. Agustinah, I. Feni and N. Mohammad, "Tracking Control of Quadrotor Using Static Output Feedback with Modified Command-Generator Tracker.," *International Review of Automatic Control (I. RE. A. CO.)*, 9.4 2015.

[7] M. Stamp, Security, Intoduction to Machine Learning with Applications in Information, California: Taylor & Francis Group, 2018.

[8] S. I-Fang, C. Ding-Li, L. Chiang and C. Yu-Chi, "Finding Visible kNN Objects in the Presence of Obstacles within the User's View Field," *International Journal of Geo-Information*, 2019.

[9] M. Nathan, M. Daniel, L. Quentin and K. Vijay, "Experimental Evaluation of Multirobot Aerial Control Algorithms," *IEEE Robotics & Automation Magazine*, pp. 1070-9932, 2010.

[10] V. A. Hasini, A. J. Beeshanga, H. Ying and D. Eryk, "Algorithm for Energy Efficient Inter-UAV Collision Avoidance," in *17th International Symposium on Communications and Information Technologies (ISCIT)*, Cairns, Queensland, Australia, 2017.

[11] O. Katsuhiko, Modern Control Engineering, 5rd Edition, New Jersey: Pretince Hall, 2010.

[12] Jean-Jacques, S. E and L. Weiping, Applied nonlinear control, New Jersey: Prentice Hall, 1991.

[13] “Qdrone,” Quanser Innovate-Educate.

LAMPIRAN

LAMPIRAN A

Data *Training*

No.	Simpangan Atas	Simpangan Kiri	Simpanagan Kanan	Kelas
1	0,6474	0,6225	2,3835	Kiri
2	0,6474	0,6815	2,2932	Kiri
3	0,6474	0,7472	2,2031	Kiri
4	0,6474	0,8182	2,1132	Kiri
5	0,6474	0,8931	2,0235	Kiri
6	0,6474	0,971	1,9341	Kiri
7	0,6474	1,0512	1,845	Kiri
8	0,6266	1,2088	1,6766	Atas
9	0,6266	1,3004	1,5815	Atas
10	0,6266	1,3932	1,487	Atas
11	0,6266	1,487	1,3932	Atas
12	0,6266	1,5815	1,3004	Atas
13	0,6266	1,6766	1,2088	Atas
14	0,6266	1,7723	1,1185	Atas
15	0,6266	1,8685	1,0301	Kanan
16	0,6266	1,965	0,944	Kanan
17	0,6266	2,0618	0,8609	Kanan
18	0,6266	2,159	0,7818	Kanan
19	0,6266	2,2564	0,7079	Kanan
20	0,6266	2,354	0,6412	Kanan
21	0,7149	0,6225	2,3835	Kiri
22	0,7149	0,6815	2,2932	Kiri
23	0,7149	0,7472	2,2031	Kiri
24	0,7149	0,8182	2,1132	Kiri
25	0,7149	0,8931	2,0235	Kiri
26	0,715	0,971	1,9341	Kiri
27	0,715	1,0512	1,845	Kiri
28	0,715	1,1334	1,7562	Kiri
29	0,7111	1,2265	1,6578	Kiri
30	0,7111	1,3122	1,5692	Kiri
31	0,7111	1,5692	1,3122	Kanan
32	0,7111	1,6578	1,2265	Kanan
33	0,711	1,747	1,142	Kanan
34	0,711	1,8366	1,0589	Kanan
35	0,711	1,9265	0,9777	Kanan

36	0,711	2,0168	0,8988	Kanan
37	0,711	2,1073	0,823	Kanan
38	0,711	2,198	0,7511	Kanan
39	0,711	2,289	0,6844	Kanan
40	0,711	2,3801	0,6246	Kanan
41	1,039	1,3018	1,5799	Kiri
42	1,039	1,5799	1,3018	Kanan
43	1,4028	1,3018	1,5799	Kiri
44	1,4028	1,5799	1,3018	Kanan
45	1,7821	1,3018	1,5799	Kiri
46	1,7821	1,5799	1,3018	Kanan
47	2,1689	1,3018	1,5799	Kiri
48	2,1689	1,5799	1,3018	Kanan
49	2,5598	1,3018	1,5799	Kiri
50	2,5598	1,5799	1,3018	Kanan
51	2,9531	1,3018	1,5799	Kiri

LAMPIRAN B PEMROGRAMAN

```
close all
clear all
clc
addpath('utils')
addpath('trajectories')

% You can change trajectory here
% trajectory generator;
%  trajhandle = @step;
% trajhandle = @circle;
trajhandle = @diamond;
%  trajhandle = @tj_from_line;

% controller
controlhandle = @controller;

% real-time
real_time = true;

% *********** YOU SHOULDN'T NEED TO CHANGE ANYTHING BELOW
**********
% number of quadrotors
nquad = 1;

% max time
time_tol = 40;

% parameters for simulation
params = crazyflie();
```

```
DataSimpangan=xlsread('cobak_3','B2:D53');
DataKelas=xlsread('cobak_3','L2:L53');
action =0;
tujuan=[7;0;3];
tujuanakhir=[7;0;3];
tikawal0=[0;1;2];
tikawal=tikawal0;
active=0;
tt=0;
savememory=0;
datamemory=0;
change=0;
cektujuan=0;

%% ************************** FIGURES
*****************************
fprintf('Initializing figures...\n')
h_fig = figure;
h_3d = gca;
axis equal
grid on
view(3);
xlabel('x [m]'); ylabel('y [m]'); zlabel('z [m]')
quadcolors = lines(nquad);

%Halangan Skenario 1 (patch 2)
% start [0;1;2] stop=[7;0;3]

vert(:,:,3) = [3.5 0.4 0;
    3 0.4 0;
    3 0.4 3;
    3.5 0.4 3;
    3.5 2.5 0;
    3 2.5 0;
    3 2.5 3;
    3.5 2.5 3;];
%-------------------------------------------------

%Halangan Skenario 2 (patch 1,2)
% start [-1;1;2.9] stop=[7;1;4]

% vert(:,:,3) = [1.5 0 0;
%     1 0 0;
%     1 0 3;
%     1.5 0 3;
%     1.5 2 0;
%     1 2 0;
%     1 2 3;
%     1.5 2 3;];
%
% vert(:,:,2) = [3.5 0.4 0;
%     3 0.4 0;
%     3 0.4 5;
%     3.5 0.4 5;
%     3.5 2.5 0;
%     3 2.5 0;
```

```
%     3 2.5 5;
%     3.5 2.5 5;];

%-----------------------------------------------------------------
--

%Halangan Skenario 3 (patch 1,2,3)
%start=[[-1;1;1]] stop=[7.5;2;2]
% vert(:,:,1) = [2.5 -2 0;
%     1 -2 0;
%     1 -2 2;
%     2.5 -2 2;
%     2.5 2 0;
%     1 2 0;
%     1 2 2;
%     2.5 2 2;];
% %lebar
%  vert(:,:,2) = [6 1.5 0;
%     3 1.5 0;
%     3 1.5 3;
%     6 1.5 3;
%     6 2.5 0;
%     3 2.5 0;
%     3 2.5 3;
%     6 2.5 3;];
%
% vert(:,:,3) = [7 -1.5 0;
%     6.5 -1.5 0;
%     6.5 -1.5 3;
%     7 -1.5 3;
%     7 3 0;
%     6.5 3 0;
%     6.5 3 3;
%     7 3 3;];
% %-----------------------------------------------------------


% % vert(:,:,1) = [1.5 0 0;
% %     1 0 0;
% %     1 0 3;
% %     1.5 0 3;
% %     1.5 2 0;
% %     1 2 0;
% %     1 2 3;
% %     1.5 2 3;];
%
% vert(:,:,1) = [2.5 -2 0;
%     1 -2 0;
%     1 -2 2;
%     2.5 -2 2;
%     2.5 2 0;
%     1 2 0;
%     1 2 2;
%     2.5 2 2;];
%
%
% % vert(:,:,2) = [3.5 0.4 0;
```

```
% %      3 0.4 0;
% %      3 0.4 5;
% %      3.5 0.4 5;
% %      3.5 2.5 0;
% %      3 2.5 0;
% %      3 2.5 5;
% %      3.5 2.5 5;];
%
% % Halangan Tebal
% % vert(:,:,2) = [6 -2 0;
% %      3 -2 0;
% %      3 -2 3;
% %      6 -2 3;
% %      6 2 0;
% %      3 2 0;
% %      3 2 3;
% %      6 2 3;];
%
% %mujur
%  vert(:,:,2) = [6 1.5 0;
%      3 1.5 0;
%      3 1.5 3;
%      6 1.5 3;
%      6 2.5 0;
%      3 2.5 0;
%      3 2.5 3;
%      6 2.5 3;];
%
% vert(:,:,3) = [7 -1.5 0;
%      6.5 -1.5 0;
%      6.5 -1.5 3;
%      7 -1.5 3;
%      7 3 0;
%      6.5 3 0;
%      6.5 3 3;
%      7 3 3;];

% define the arbitrary polygon(patch) using the vertice
number(index) you defined abov
fac = [1 2 6 5;2 3 7 6;3 4 8 7;4 1 5 8;1 2 3 4;5 6 7 8];
% specify patch (polygons) in patch() function
% just call the patch function multiple times to draw multiple
cubes
%    patch('Faces',fac,'Vertices',vert(:,:,1),'FaceColor','y');  %
draw the green cube
%    patch('Faces',fac,'Vertices',vert(:,:,2),'FaceColor','g');  %
draw the green cube
%    patch('Faces',fac,'Vertices',vert(:,:,3),'FaceColor','b');  %
draw the green cube
axis([-1 8 -4 6 0 6])
hold on
% set(gcf,'Renderer','OpenGL')

%% ********************** INITIAL CONDITIONS
**********************
fprintf('Setting initial conditions...\n')
max_iter  = 2000;      % max iteration
```

```
starttime = 0;          % start of simulation in seconds
tstep     = 0.01;       % this determines the time step at which
the solution is given
cstep     = 0.05;       % image capture time interval
nstep     = cstep/tstep;
time      = starttime; % current time
err = []; % runtime errors
for qn = 1:nquad
    % Get start and stop position
    des_start = trajhandle(0, qn, active, tt, tujuan, tikawal);
    des_stop  = trajhandle(inf, qn, active, tt, tujuan, tikawal);
    stop{qn}  = tujuanakhir;
    x0{qn}    = init_state( des_start.pos, 0 );
    xtraj{qn} = zeros(max_iter*nstep, length(x0{qn}));
    ttraj{qn} = zeros(max_iter*nstep, 1);
end

x         = x0;         % state

pos_tol   = 0.01;
vel_tol   = 0.01;
h = fill3([vert(1:2,1,3);vert(6,1,3);vert(5,1,3)],
[vert(1:2,2,3);vert(6,2,3);vert(5,2,3)],[vert(1:2,3,3);vert(6,3,3)
;vert(5,3,3)],'r');
h1 = fill3([vert(2:3,1,3);vert(7,1,3);vert(6,1,3)],
[vert(2:3,2,3);vert(7,2,3);vert(6,2,3)],[vert(2:3,3,3);vert(7,3,3)
;vert(6,3,3)],'r');
h2 = fill3([vert(3:4,1,3);vert(8,1,3);vert(7,1,3)],
[vert(3:4,2,3);vert(8,2,3);vert(7,2,3)],[vert(3:4,3,3);vert(8,3,3)
;vert(7,3,3)],'r');
h3 = fill3([vert(4,1,3);vert(1,1,3);vert(5,1,3);vert(8,1,3)],
[vert(4,2,3);vert(1,2,3);vert(5,2,3);vert(8,2,3)],[vert(4,3,3);ver
t(1,3,3);vert(5,3,3);vert(8,3,3)],'r');
h4 = fill3([vert(1:2,1,3);vert(3,1,3);vert(4,1,3)],
[vert(1:2,2,3);vert(3,2,3);vert(4,2,3)],[vert(1:2,3,3);vert(3,3,3)
;vert(4,3,3)],'r');
h5 = fill3([vert(5:6,1,3);vert(7,1,3);vert(8,1,3)],
[vert(5:6,2,3);vert(7,2,3);vert(8,2,3)],[vert(5:6,3,3);vert(7,3,3)
;vert(8,3,3)],'r');

% h6 = fill3([vert(1:2,1,2);vert(6,1,2);vert(5,1,2)],
[vert(1:2,2,2);vert(6,2,2);vert(5,2,2)],[vert(1:2,3,2);vert(6,3,2)
;vert(5,3,2)],'r');
% h7 = fill3([vert(2:3,1,2);vert(7,1,2);vert(6,1,2)],
[vert(2:3,2,2);vert(7,2,2);vert(6,2,2)],[vert(2:3,3,2);vert(7,3,2)
;vert(6,3,2)],'r');
% h8 = fill3([vert(3:4,1,2);vert(8,1,2);vert(7,1,2)],
[vert(3:4,2,2);vert(8,2,2);vert(7,2,2)],[vert(3:4,3,2);vert(8,3,2)
;vert(7,3,2)],'r');
% h9 = fill3([vert(4,1,2);vert(1,1,2);vert(5,1,2);vert(8,1,2)],
[vert(4,2,2);vert(1,2,2);vert(5,2,2);vert(8,2,2)],[vert(4,3,2);ver
t(1,3,2);vert(5,3,2);vert(8,3,2)],'r');
% h10 = fill3([vert(1:2,1,2);vert(3,1,2);vert(4,1,2)],
[vert(1:2,2,2);vert(3,2,2);vert(4,2,2)],[vert(1:2,3,2);vert(3,3,2)
;vert(4,3,2)],'r');
% h11 = fill3([vert(5:6,1,2);vert(7,1,2);vert(8,1,2)],
[vert(5:6,2,2);vert(7,2,2);vert(8,2,2)],[vert(5:6,3,2);vert(7,3,2)
```

```
;vert(8,3,2)],'r');
%% ************************ RUN SIMULATION
************************
OUTPUT_TO_VIDEO = 1;
if OUTPUT_TO_VIDEO == 1
    v = VideoWriter('diamond.avi');
    open(v)
end

fprintf('Simulation Running....')
% Main loop
for iter = 1:max_iter
    iter;
    timeint = time:tstep:time+cstep;
      vert(:,2,3)=vert(:,2,3)-0.005;
% %       vert(:,2,2)=vert(:,2,2)+0.005;
     set(h,'Vertices',[vert(1:2,:,3);vert(6,:,3);vert(5,:,3)]);
     set(h1,'Vertices',[vert(2:3,:,3);vert(7,:,3);vert(6,:,3)]);
     set(h2,'Vertices',[vert(3:4,:,3);vert(8,:,3);vert(7,:,3)]);

set(h3,'Vertices',[vert(4,:,3);vert(1,:,3);vert(5,:,3);vert(8,:,3)
]);
     set(h4,'Vertices',[vert(1:2,:,3);vert(3,:,3);vert(4,:,3)]);
     set(h5,'Vertices',[vert(5:6,:,3);vert(7,:,3);vert(8,:,3)]);

%       set(h6,'Vertices',[vert(1:2,:,2);vert(6,:,2);vert(5,:,2)]);
%       set(h7,'Vertices',[vert(2:3,:,2);vert(7,:,2);vert(6,:,2)]);
%       set(h8,'Vertices',[vert(3:4,:,2);vert(8,:,2);vert(7,:,2)]);
%
set(h9,'Vertices',[vert(4,:,2);vert(1,:,2);vert(5,:,2);vert(8,:,2)
]);
%
set(h10,'Vertices',[vert(1:2,:,2);vert(3,:,2);vert(4,:,2)]);
%
set(h11,'Vertices',[vert(5:6,:,2);vert(7,:,2);vert(8,:,2)]);
     drawnow

    tic;
            % Deteksi Halangan
        for i=1:size(vert,3)
            if x{qn}(2)>vert(2,2,i)&&x{qn}(2)<vert(6,2,i)
                if x{qn}(3)>vert(2,3,i)&&x{qn}(2)<vert(3,3,i)
                    jarakku(i)=vert(2,1,i)-x{qn}(1);
                    if jarakku(i)<0
                        jarakku(i)=444;
                    end
                end
            else
                jarakku(i)=444;
            end
            [jarak,rintangan]=min(jarakku);
        end

        if active==1
        if x{qn}(1)>(tujuan(1)-0.05)&&x{qn}(1)<(tujuan(1)+0.05)
            if x{qn}(2)>(tujuan(2)-
0.05)&&x{qn}(2)<(tujuan(2)+0.05)
```

```
                if x{qn}(3)>(tujuan(3)-
0.05)&&x{qn}(3)<(tujuan(3)+0.05)
                    active=2;
                    tt=time;
                    tikawal=[x{qn}(1);x{qn}(2);x{qn}(3)];
                    datamemory=0;
                    savememory=0;
                    if change==1
                        change=0;
                    end
                end
            end
        end
        end

        if active==2
        if x{qn}(1)>(tujuan2(1)-0.05)&&x{qn}(1)<(tujuan2(1)+0.05)
            if x{qn}(2)>(tujuan2(2)-
0.05)&&x{qn}(2)<(tujuan2(2)+0.05)
                if x{qn}(3)>(tujuan2(3)-
0.05)&&x{qn}(3)<(tujuan2(3)+0.05)
                    active=0;
                    tikawal=[x{qn}(1);x{qn}(2);x{qn}(3)];
                    tt=time;
                end
            end
        end
        end

        if jarak>0.5&&jarak<0.59
            tinggi=vert(3,3,rintangan)-x{qn}(3)+0.3;
            kiri=vert(7,2,rintangan)-x{qn}(2)+0.3;
            kanan=x{qn}(2)-vert(3,2,rintangan)+0.3;
            dalam=vert(1,1,rintangan)-vert(2,1,rintangan);
            smpat=sqrt(jarak^2+tinggi^2);
            smpkr=sqrt(jarak^2+kiri^2);
            smpkn=sqrt(jarak^2+kanan^2);
            if change==0
                action=simpangan(smpat, smpkr, smpkn,
DataSimpangan, DataKelas);
            end
            tikawal=[x{qn}(1);x{qn}(2);x{qn}(3)];
            tt=time;
            active=1;
            cek=[smpat smpkr smpkn action]
            [x{qn}(1);x{qn}(2);x{qn}(3)];
            if action==1
                if cektujuan<1

tujuan=[x{qn}(1)+jarak;x{qn}(2)+kiri;x{qn}(3)];
                    tujuan2=tujuan+[dalam+0.2;0;0];
                    cektujuan=1;
                end
                pojok=vert(7,:,rintangan);
                jrkpojok=((pojok(1)-tujuan(1))^2+(pojok(2)-
tujuan(2))^2)^0.5;
            elseif action==2
```

```
                if cektujuan<1
                    tujuan=[x{qn}(1)+jarak;x{qn}(2)-
kanan;x{qn}(3)];
                    tujuan2=tujuan+[dalam+0.2;0;0];
                    cektujuan=1;
                end
                pojok=vert(3,:,rintangan);
                jrkpojok=((pojok(1)-tujuan(1))^2+(pojok(2)-
tujuan(2))^2)^0.5;
            else
                if cektujuan<1

tujuan=[x{qn}(1)+jarak;x{qn}(2);x{qn}(3)+tinggi];
                    tujuan2=tujuan+[dalam;0;0];
                    cektujuan=1;
                end
                pojok=vert(3,:,rintangan);
                jrkpojok=((pojok(1)-tujuan(1))^2+(pojok(3)-
tujuan(3))^2)^0.5
            end
            tujuan;
            savememory=savememory+1;
            if savememory>0
                datamemory(savememory,:)=jrkpojok;
                if savememory>2
                    if abs(datamemory(savememory,:)-
datamemory(savememory-1,:))>0.0001&&change==0
                        if action==1
                        action=2;
                        cektujuan=0;
                        elseif action==2
                        action=1
                        cektujuan=0;
                        end
                        change=1;
                    end
                end
            end
            datamemory
        end

        if active==2
            tujuan=tujuan2;
        elseif active==0
            tujuan=tujuanakhir;
        end

        % Iterate over each quad
    for qn = 1:nquad
        % Initialize quad plot
        if iter == 1
            QP{qn} = QuadPlot(qn, x0{qn}, 0.1, 0.04,
quadcolors(qn,:), max_iter, h_3d);
            desired_state = trajhandle(time, qn, active, tt,
tujuan, tikawal);
            QP{qn}.UpdateQuadPlot(x{qn}, [desired_state.pos;
desired_state.vel], time);
```

```
            h_title = title(sprintf('iteration: %d, time: %4.2f,
kecepatan: %4.2f, yaw: %4.2f', iter, time, desired_state.vel,
x0{qn}(10)));
        end

        % Run simulation
        [tsave, xsave] = ode45(@(t,s) quadEOM(t, s, qn, active,
tt, tujuan, tikawal, controlhandle, trajhandle, params), timeint,
x{qn});
        x{qn}    = xsave(end, :)';

        % Save to traj
        xtraj{qn}((iter-1)*nstep+1:iter*nstep,:) = xsave(1:end-
1,:);
        ttraj{qn}((iter-1)*nstep+1:iter*nstep) = tsave(1:end-1);

        % Update quad plot
        desired_state = trajhandle(time + cstep, qn, active, tt,
tujuan, tikawal);

        QP{qn}.UpdateQuadPlot(x{qn}, [desired_state.pos;
desired_state.vel], time + cstep);
        set(h_title, 'String', sprintf('iteration: %d, time:
%4.2f, kecepatan: %4.2f, Halangan: %4.2f action: %4.2f', iter,
time + cstep, desired_state.vel, jarak, action))
        if OUTPUT_TO_VIDEO == 1
            im = frame2im(getframe(gcf));
            writeVideo(v,im);
        end
    end
    time = time + cstep; % Update simulation time
    t = toc;
    % Check to make sure ode45 is not timing out
%     if(t> cstep*50)
%         err = 'Ode45 Unstable';
%         break;
%     end

    % Pause to make real-time
    if real_time && (t < cstep)
        pause(cstep - t);
    end

    % Check termination criteria
    if terminate_check(x, time, stop, pos_tol, vel_tol, time_tol)
        break
    end
end

if OUTPUT_TO_VIDEO == 1
    close(v);
end

%% ************************ POST PROCESSING
************************
% Truncate xtraj and ttraj
```

```
for qn = 1:nquad
    xtraj{qn} = xtraj{qn}(1:iter*nstep,:);
    ttraj{qn} = ttraj{qn}(1:iter*nstep);
end

% Plot the saved position and velocity of each robot
for qn = 1:nquad
    % Truncate saved variables
    QP{qn}.TruncateHist();
    % Plot position for each quad
    h_pos{qn} = figure('Name', ['Quad ' num2str(qn) ' :
position']);
    plot_state(h_pos{qn}, QP{qn}.state_hist(1:3,:),
QP{qn}.time_hist, 'pos', 'vic');
    plot_state(h_pos{qn}, QP{qn}.state_des_hist(1:3,:),
QP{qn}.time_hist, 'pos', 'des');
    % Plot velocity for each quad
    h_vel{qn} = figure('Name', ['Quad ' num2str(qn) ' :
velocity']);
    plot_state(h_vel{qn}, QP{qn}.state_hist(4:6,:),
QP{qn}.time_hist, 'vel', 'vic');
    plot_state(h_vel{qn}, QP{qn}.state_des_hist(4:6,:),
QP{qn}.time_hist, 'vel', 'des');
end
if(~isempty(err))
    error(err);
end

fprintf('finished.\n')
```

Halaman ini sengaja dikosongkan

RIWAYAT PENULIS

Irfin Sandra Asti lahir di Malang pada tanggal 27 Juni 1995 sebagai anak pertama dari pasangan Akhmad Alwi dan Kholisatul Khotimah. Penulis adalah alumni dari SMAN 5 Malang yang lulus pada tahun 2013. Pendidikan sarjana terapan di tempuh di Program Studi DIV Teknik Elektronika Politeknik Negeri Malang, lulus tahun 2017. Pada tahun yang sama, penulis diterima di Program Magister di Jurusan Teknik Elektro, Institut Teknologi Sepuluh Nopember dengan bidang keahlian Teknik Sistem Pengaturan. Pada saat ini, penulis sedang menyelesaikan studi magisternya pada bulan Januari 2020.

(irfinsandra@gmail.com / 082210312018)