SISTEM NAVIGASI OTOMATIS *QUADCOPTER* UNTUK MENGHINDARI *OBSTACLE* DENGAN PENGOLAHAN CITRA MENGGUNAKAN METODE *HOUGH TRANSFORM*

SKRIPSI

Untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Teknik

Disusun oleh:
Okke Rizki Kurniawan
NIM: 145150301111001

PROGRAM STUDI TEKNIK KOMPUTER
JURUSAN TEKNIK INFORMATIKA
FAKULTAS ILMU KOMPUTER
UNIVERSITAS BRAWIJAYA
MALANG
2019

PENGESAHAN

SISTEM NAVIGASI OTOMATIS *QUADCOPTER* UNTUK MENGHINDARI *OBSTACLE* DENGAN PENGOLAHAN CITRA MENGGUNAKAN METODE *HOUGH TRANSFORM*

SKRIPSI

Diajukan untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Teknik

Disusun Oleh :
Okke Rizki Kurniawan
NIM: 1451503011110001

Telah diperiksa dan disetujui oleh:

Dosen Pembimbing I	Dosen Pembimbing II
Gembong Edhi Setyawan, S.T, M.T	Mochammad Hannats Hanafi Ichsan, S.ST.,M.T
NIK: 201208 761201 1 001	NIP: 19881229 201903 1010

Mengetahui
Ketua Jurusan Teknik Informatika

Tri Astoto Kurniawan, S.T., M.T., Ph.D.
NIP. 19710518 200312 1 001

PERNYATAAN ORISINALITAS

Saya menyatakan dengan sebenar-benarnya bahwa sepanjang pengetahuan saya, di dalam naskah skripsi ini tidak terdapat karya ilmiah yang pernah diajukan oleh orang lain untuk memperoleh gelar akademik di suatu perguruan tinggi, dan tidak terdapat karya atau pendapat yang pernah ditulis atau diterbitkan oleh orang lain, kecuali yang secara tertulis disitasi dalam naskah ini dan disebutkan dalam daftar pustaka.

Apabila ternyata didalam naskah skripsi ini dapat dibuktikan terdapat unsur-unsur plagiasi, saya bersedia skripsi ini digugurkan dan gelar akademik yang telah saya peroleh (sarjana) dibatalkan, serta diproses sesuai dengan peraturan perundang-undangan yang berlaku (UU No. 20 Tahun 2003, Pasal 25 ayat 2 dan Pasal 70).

Malang, 29 Juni 2018

Okke Rizki Kurniawan

NIM: 145150301111001

KATA PENGANTAR

Puji syukur kehadirat Allah SWT yang telah melimpahkan rahmat, taufik dan hidayah-Nya sehingga laporan skripsi yang berjudul "Sistem Navigasi Otomatis *Quadcopter* Untuk Menghindari *Obstacle* Dengan Pengolahan Citra Menggunakan Metode *Hough Transform*" ini dapat terselesaikan. Laporan skripsi ini disusun dalam rangka memenuhi persyaratan untuk memperoleh gelar Sarjana Komputer di Fakultas Ilmu Komputer

Penulis menyadari bahwa penyusunan laporan skripsi ini tidak lepas dari bantuan berbagai pihak baik secara langsung maupun tidak langsung. Dalam kesempatan ini penulis ingin menyampaikan ucapan terima kasih atas bantuannya kepada semua pihak, sehingga penulis dapat menyelesaikan laporan ini dengan baik. Ucapan terima kasih tersebut khususnya kepada :

1. Allah yang Maha Esa yang selalu memberikan petunjuk dan hikmah dalam penulisan ini.
2. Orang Tua dan Keluarga atas nasehat, kasih sayang serta dukungan materil dan moril.
3. Bapak Gembong Edhi Setyawan, S.T, M.T dan Bapak Mochammad Hannats Hanafi Ichsan, S.ST.,M.T selaku dosen pembimbing yang telah memberikan dukungan, kritik dan saran yang membangun serta ilmu dalam membimbing penulis hingga pengerjaan skripsi ini selesai.
4. Bapak Wayan Firdaus Mahmudy, S.Si, M.T, Ph.D. selaku Dekan Fakultas Ilmu Komputer Universitas Brawijaya Malang.
5. Bapak Heru Nurwarsito, Ir., M.Kom. selaku Wakil Dekan I Bidang Akademik Fakultas Ilmu Komputer Universitas Brawijaya Malang.
6. Bapak Tri Astoto Kurniawan, S.T., M.T., Ph.D. selaku Ketua Jurusan Teknik Informatika.
7. Bapak Dahnial Syauqy, S.T., M.T., M.Sc. selaku Ketua Program Studi Teknik Komputer Universitas Brawijaya Malang.
8. Seluruh civitas akademika Informatika Universitas Brawijaya dan teman-teman Teknik Komputer Angkatan 2014 yang telah banyak memberi bantuan dan dukungan selama peneliti menempuh studi di Teknik Komputer Universitas Brawijaya dan selama penyelesaian skripsi ini.
9. Sabrina qurroti a'yuna, amelio eric, nahja falahil umam, ananta tumonglo, okky, riyad febrian, hendra, zulfikar ardi serta rekan-rekan lainnya yang turut memberikan motivasi dan hingga pengerjaan skripsi ini selesai.
10. Dan orang-orang yang selalu mendukung serta mendoakan kelancaran proses skripsi ini yang tidak bisa disebutkan satu per satu atas semua doa dan dukungannya.

Dengan segala keterbatasan pengetahuan yang dimiliki, penulis sadar bahwa penulisan laporan skripsi ini masih jauh dari kesempurnaan. Sehingga penulis berharap agar laporan skripsi ini dapat bermanfaat dan merupakan salah satu informasi yang berguna bagi pembaca.

Malang, 29 Juli 2019

Penulis

Okke2269@gmail.com

ABSTRAK

UAV (Unmanned Aerial Vehicle) yang lebih dikenal dengan sebutan drone merupakan kendaraan yang dapat dikendalikan oleh pilot secara jarak jauh. Pada saat ini drone banyak dikembangkan menjadi sistem kendali otomatis untuk mendeteksi suatu objek. Berlandaskan dari permasalahan tersebut *quadcopter* perlu dikembangkan untuk menghindari suatu objek. Pada penelitian ini akan menggunakan pengolahan citra digital yang akan diterapkan pada *quadcopter* untuk menghindari sebuah *obstacle* secara otomatis. Pada penelitian ini pemrosesan citra digital akan menggunakan dua metode yaitu *canny edge detection* dan *hough transform. Quadcopter* yang digunakan dalam penelitian ini berjenis *Parrot Ar.Drone 2.0.* Hasil dari pengujian yang telah dilakukan pada penelitian ini akan mendapatkan ketepatan dari quadcopter dengan menggunakan jarak dari lebar *obstacle* beserta estimasi waktu yang dibutuhkan oleh *quadcopter* yang digunakan untuk menghindari *obstacle*. Pengujian berdasarkan ketepatan (*error*) yang didapatkan dari pengujuan sistem ini adalah 0.07125%. Jarak pendeteksian maksimal pada *quadcopter* yang dapat dideteksi yaitu pada saat lebar *obstacle* telah mencapai 294 *pixel* dan 142 *pixel.* Dan untuk penelitian estimasi waktu akan mendapatkan waktu tempuh *quadcopter* rata-rata 7,464857 detik.

Kata Kunci : *Pengolahan Citra, Hough Transform, Quadcopter*

ABSTRACT

UAV (Unmanned Aerial Vehicle), which is better known as drones is a vehicle that can be controlled remotely by the pilot. At this time the drone developed into an automatic control system for detecting an object. Based on these problems quadcopter need to be developed to avoid an object. This research will use digital image processing to be applied in order to avoid an obstacle quadcopter automatically. In this study, digital image processing will use two methods: the canny edge detection and hough transform. Quadcopter used in this study manifold Parrot Ar.Drone 2.0. The results of the testing that was done in this study will get the accuracy of quadcopter using a wide range of obstacles along with the estimated time required by quadcopter used to avoid the obstacle. The test is based on the accuracy (error) obtained from pengujuan this system is 0.07125%. Maximum detection range at quadcopter that can be detected, namely when the width of the obstacle has reached 294 pixels and 142 pixels. And for research will get a time estimate travel time quadcopter average 7.464857 seconds.

Keywords: *Image Processing, Hough Transform, Quadcopter*

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

BAB 1 PENDAHULUAN

1.1 Latar belakang

UAV (Unmanned Aerial Vehicle) dikenal dengan banyak nama seperti *remotely piloted vehicle,* dan *remote controlled aircraft* atau lebih dikenal dengan sebutan drone karena merupakan kendaraan yang dapat dikendalikan oleh pilot secara jarak jauh (LaFay, 2015). *Quadcopter* adalah jenis drone yang digerakkan oleh empat buah *propeller* sehingga memungkinkan untuk melakukan gerakan lebih stabil daripada jenis drone lain. Beberapa *quadcopter* juga mendukung pengolahan citra digital yang dapat digunakan sebagai acuan sistem kendali otomatis yang dapat diprogram pada drone sehingga memungkinkan pergerakan tanpa kendali manual, serta memiliki beberapa sensor pendukung seperti sensor navigasi yang berfungsi untuk mengetahui posisi drone, sensor jarak yang berfungsi untuk mengukur jarak drone dengan objek didepan sensor, sensor inersia yang berfungsi mengukur kecepatan serta pergerakan drone ketika dijalankan (SA, 2016).

Seiring perkembangan *quadcopter,* kendali pada *quadcopter* mulai merambah pada kendali otomatis dengan menggunakan bantuan pengolahan citra maupun menggunakan sensor ultrasonic. Pada penelitian yang telah dilakukan oleh (Yosa Rosario, 2013) dengan menggunakan sensor *ultrasonic* masih terbatas untuk bidang yang memilki permukaan yang hampir sepenuhnya datar. Dan untuk proses pengolahan citra tidak terbatas pada permukaan bidang datar. Untuk proses pengolahan citra dibagi menjadi 2 macam, yaitu pengolahan citra berdasar segmentasi dan menggunakan metode *learning.* Dengan memanfaatkan fitur tersebut, *quadcopter* dapat menggantikan sistem navigasi manual yang terdapat pada *quadcopter* agar dapat dikendalikan secara otomatis.

(Boudjit & Larbes, 2015) telah melakukan penelitian yang memanfaatkan proses citra digital yang bertujuan agar *quadcopter* dapat mengikuti suatu garis lurus, tetapi tidak dapat mengikuti ketika garis berbelok. Penelitian lain yang telah dilakukan oleh (Brandao, et al., 2015) dapat menjadikan sungai menjadi garis yang akan diikuti oleh *quadcopter* serta *tracking* pada *quadcopter* dengan cara menentukan posisi serta perbatasan dari saluran air, hasil yang didapat dari penelitian tersebut sangat bergantung pada kontras serta bayangan yang terdeteksi pada *frame* yang diambil. Dengan adanya penelitian yang telah disebutkan sebelumnya, penelitian ini akan menggunakan ketinggian dari sebuah *obstacle* serta jarak yang didapat oleh tangkapan *frame* kamera yang akan dikenali sebagai garis, hingga sistem dapat menentukan pergerakan berdasar perhitungan dari tangkapan yang didapat.

Obstacle adalah sebuah halangan atau rintangan yang dapat menghambat sebuah pergerakan. Dalam penelitian pengembangan *quadcopter* sebuah *obstacle* digunakan untuk menguji sebuah *quadcopter* agar dapat melewatinya berdasarkan program yang telah dijalankan secara otomatis. Teknologi yang dapat membatu mengenali *obstacle* salah satunya adalah

menggunakan pengolahan citra digital pada kamera yang terdapat pada *quadcopter.* Pada pemrosesan citra digital dapat diselesaikan dengan beragam metode, antara lain dengan mencocokan pola yang berbentuk garis pada sebuah objek. Metode yang digunakan adalah *Hough transform. Hough Transform* adalah suatu metode untuk mendeteksi garis, lingkaran, atau bentuk lainya. Dasar dari metode *Hough transform* menggunakan transformasi garis, yang mana ditujukan guna mencari garis lurus yang terdapat pada citra biner yang memungkinkan sebagian titik-titik yang didapat merupakan himpunan garis. Titik pada tiap garis direpresentasikan sebagai titik koordinat polar*(rho,theta)* (Zain, 2018).

Parrot AR.Drone merupakan jenis *quadcopter* yang memiliki fitur yang akan digunakan pada penelitian ini, seperti kamera yang telah terpasang sehingga tidak memerlukan kamera tambahan untuk mengambil *frame* yang dibutuhkan. Sebelum citra digital dapat diolah oleh *Hough Transform,* citra digital memerlukan proses *pre-processing. Pre-processing* pada citra digital antara lain konversi RGB menjadi grayscale dan guassian blur agar dapat digunakan pada proses pendeteksian tepi. Setelah tepian diperoleh, hasil pendeteksian tepian dapat dimasukkan untuk diolah pada *Hough Transform.* hasil dari olahan akanmenjadi acuan untuk mengenali obstacle, sehingga dapat digunakan untuk menentukan pergerakan *quadcopter* secara otomatis.

1.2 Rumusan masalah

Pada pemaparan latar belakang tersebut, akan merumuskan suatu masalah yang akan akan dibahas sebagai berikut :

1. Bagaimana perancangan sistem navigasi otomatis *quadcopter* untuk menghindari *obstacle* dengan pengolahan citra memakai metode *hogh transorm* ?
2. Bagaimana cara mengimplementasikan sistem navigasi otomatis *quadcopter* untuk menghindari *obstacle* dengan pengolahan citra menggunakan metode *Hough Transform* ?
3. Bagaimana hasil dari sistem navigasi otomatis *quadcopter* untuk menghindari *obstacle* dengan pengolahan citra menggunakan metode *hough transform* ?

1.3 Tujuan

Berdasar pada Rumusan Masalah, tujuan yang ingin didapatkan oleh peneliti sebagai berikut :

1. Dapat membangun sistem navigasi otomatis *quadcopter* untuk menghindari *obstacle* dengan pengolahan citra menggunakan metode *hough transform.*
2. Dapat mengimplementasikan sistem navigasi otomatis *quadcopter* untuk menghindari *obstacle* dengan pengolahan citra menggunakan metode *hough transform*.

3. Dapat mengetahui hasil dari sistem navigasi otomatis *quadcopter* untuk menghindari *obstacle* dengan pengolahan citra menggunakan metode *hough transform.*

1.4 Manfaat

Pada pnelitian ini akan dapat memberikan maanfaat terhadap penelitian selajutnya sebagai berikut:

1. Memberikan wawasan dan pengeetahuan bagi pembaca atau peneliiti mengenai *quadcopter* menggunakan metode *hough transform* serta aplikasi *Robot Operating System (ROS).*
2. Dapat dijadikan sebuah patokan penelitian *quadcopter* pada penelitian selanjutnya tentang kendai *quadcoopter* secara otomatis.
3. Peningkatan performa *quadcopter* dari segi reliabilitas untuk menghindari penghalang.

1.5 Batasan masalah

Guna mendapatkan penelitian yang sesuai dengan topik maka dibutuhkan batasan masalah agar permasalahan dapat terfokiuskan antara lain:

1. *Quadcoter* yang digunakan adalah *AR Drone parrot 2.0.*
2. *Quadcopter* dioperasikan di ruangann kosong dan meiliki luas ruangan minimal 8 x 8 x 4 m.
3. Hanya menggunakan 1 *obstacle* dengan ukuran yang telah ditentukan serta memiliki bentuk garis vertikal.

1.6 Sistematika pembahasan

Sistematika yang akan diikuti pada penulisan skripsi ini sebagai berikut:

BAB 1 Pendahuluan

Bab nii memaparkan tentang Latar belakang yang memaparkan tentang alasan penelitian ini serta hasiil dari penelitiansebelumnya, Rumusan Masalah yang menguraikan acuan diadakannya penelitian ini, TujuanPenelitian, Manfaat Penelitian, Batasan.Masalah, serta Sistematika penulisan yang berguna untuk perancangan sistem navigasi otomatis *quadcopter* untuk menghindari *obstacle* dengan pengolahan citra menggunakan metode *Hough Transform*.

BAB 2 Landasan kepustakaan

Kajian Pustaka yang akan dipakai sebagai sumber keterangan pendukung penelitian serta sumber-sumber terkait dengan dasar teori yang akan digunakan akan dipaparkan dalam bab ini untuk menunjang proses pelaksanaan penelitian sistem navigasi otomatis *quadcopter* untuk menghindari *obstacle*.

BAB 3 Metode Penelitian

Baa ini menjeaskan tentang metode serta langkaah-langkah kerja yng dilakukan dalm pelaporan penelitiian. Sistematika metodeyangdigunakan antara lain analisis kebutuhan yang menguraikan kebutuhan ketika perancanganistem, perancangan dan implementasi,. pengujian dan analiis yang berisi hasil akhir yang didapat setelah langkah perancangan dan implementasi, dan penutup.

BAB 4 Analisis Kebutuhan

Bab ini akan menjelskan tentangapasajayang dibutuhkan oleh user, baiik dalam kebutuuhan fungsional serta nonfungsional serta kebuthan dari user itu sendiri.

BAB 5 Perancangan dan Implementasi

Padabab nii akan membahas mengenaiperancangan dan pengimplemantasian sistem yang telah dibuat agar *quadcopter* dapat menghindari *obstacle* secara otomatis.

BAB 6 Pengujian dan Analisis

Pengujian dan analisis sering kali disebut dengan tahapan inti dari sebuah penelitian. Karena seluruh hasil akhir yang akan didapatkan akan terlihat saat sistem diterapkan sesuaidenganrancangansistem yag telah dibuatpada bab sebelumnya.

BAB 7 Penutup

Berisikan hasil akhir dari hasil penelitianyang telah dilakukan,. serta berisikan sarann yang tentunya akan berengaruh untuk pengembngan sistem lanjutan.

BAB 2 LANDASAN KEPUSTAKAAN

Pada bab ini berisikan beberapa penjelasan yang terdapat pada penelitian yang dilakukan sebelumnya yang berkaitan dengan penelitian yang diajukan, sedangkan guna membahas berbagai teori yang digunakan dalam penyusunan penelitian ini akan dijelaskan pada Dasar teori.

2.1 Kajian Pustaka

Penelitian yang bertujuan untuk mengimplementasikan sistem navigasi ruangan *quadcopter* dengan menggunakan sensor *ultrasonic* oleh (Yosa Rosario, 2013). Dalam penelitian yang dilakukan oleh Yosa Rosario bertujuan membuat navigasi *quadcopter* didalam ruangan dimana *quadcopter* dapat berpindah posisi didalam ruangan tersebut. Pada penelitian ini *Quadcopter* menggunakan sensor ultrasonic untuk membaca jarak dari *quadcopter* memanfaatkan kontroler PID yang tertanam pada modul ardupilot untuk mengatur dan menggerakkan *quadcopter.*

Skripsi tentang *quadcopter* sebelumnya yang telah dilakukan oleh (Zain, 2018) yang menggunakan garis sebagai jalurnya dengan menggunakan metode *hough transform* untuk mencari garis yang didapat dari *frame* kamera. Pada Hough line transform digunakan titik-titik pada citra biner sebagai bagian dari himpunan kemungkinan garis. Titik pada tiap garis direpresentasikan sebagai titik koordinat polar (rho,theta). Persamaan untuk tiap garis adalah rho = x cos theta + y sin theta. Pada penelitian ini akan mendapatkan hasil melewati *obstacle* dengan menggunakan *quadcopter*. Perbedaan dari penelitian yang telah dilakukan dan sekarang seperti Table 2.1.

Table 2.1 Perbedaan Penelitian

No	(Sistem Navigasi Ruangan Quadcopter Dengan Menggunakan Sensor Ultrasonic) **(Yosa Rosario, 2013)**	(SISTEM AR DRONE PENGIKUT GARIS MENGGUNAKAN ALGORITMA *PROGRESSIVE PROBABILISTIC HOUGH TRANSFORM)* **(Zain, 2018)**	Perbedaan
1	Penelitian yang sebelumnya untuk navigasi otomatis quadcopter menggunakan acuan bidang datar	Pada penelitian sebelumnya bertujuan untuk mengikuti garis yang dijadikan sebagai acuan gerak *quadcopter*.	Pada penelitian ini digunakan untuk menghindari *obstacle.*

No	(Sistem Navigasi Ruangan Quadcopter Dengan Menggunakan Sensor Ultrasonic) **(Yosa Rosario, 2013)**	(SISTEM AR DRONE PENGIKUT GARIS MENGGUNAKAN ALGORITMA *PROGRESSIVE PROBABILISTIC HOUGH TRANSFORM)* **(Zain, 2018)**	Perbedaan
2	Pada penelitian ini menggunakan sensor ultrasonic	Pada penelitian menggunakan kamera bagian bawah.	Pada penelitian ini menggunakan kamera bagian depan
3	Hasil dari penelitian ini bergantung pada bidang yang permukaannya hampir datar	Hasil dari penelitian sebelumnya hanya bergantung pada ketinggian yang dipakai.	Pada penelitian ini ketingian serta bidang datar tidak mempengaruhi hasil pengujian. dan yang akan berpengaruh adalah jarak dan kecepatan.

Pada penelitian ini, *quadcopter* akan menghindari *obstacle* secara otomatis dengan cara mendeteksi *obstacle* menggunakan metode dari *hough transform* yang akan membetuk garis yang diambil dari *frame* yang didapat dari kamera depan *quadcopter*.

2.2 Dasar Teori

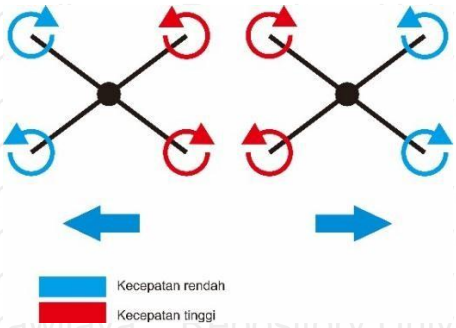
2.2.1 Quadcopter

Quadcopter merupakan jenis drone yang memiliki empat buah *motor* horisontal dan *propeller* dengan dua atau tiga daun baling-baling (Tzivaras, 2016). *motor* - *motor* yang dimiliki *quadcopter* dapat digunakan untuk mengatur dari kecepatanya yang serta dapat digunakan untuk menggubah pergerakan dari *quadcopter.* Kelebihan dari *quadcopter* adalah dapat melakukan tracking objek serta pergerakan ke segala arah memudahkan melalui kamera yang terdapat pada *quadcopter (*Gaol, 2017).

Pada *quadcopter*, setiap baling-baling yang terpasang pada motor memiliki gaya angkat dikarenakan adanya tekanan yang mengarah kebawah. Pada empat buah baling-baling yang tersedia, dibagi dengan dua arah yang

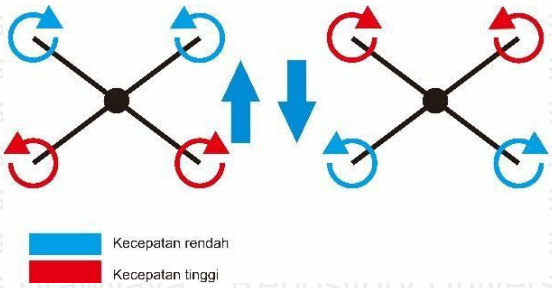
mempunyai putaran seperti arah jarum jan dan dan dua arah melawan jarum jam pada tiap dua baling-baling yang memungkinkan *quadcopter* untuk dapat terbang lebih stabil. Pergerakan *quadcopter* dapat dibagi menjadi 4 gerakan, yaitu Roll atau gerakan kanan dan kiri, *Pitch* atau gerakan kedepan dan belakang, *Yaw* atau gerakan putar kanan dan kiri, serta *Gaz* atau gerakan naik dan turun yang diuraikan sebagai berikut :

1. *Roll* merupakan gerak dari *quadcopter* kearah kanan dan kiri. pada gambar 2.1, jika pada saat *quadcopter* berjalan kearah kanan maka intensitas kecepatan *motor* sebelah kiri akan ditambah sehingga menambah gaya angkat pada tubuh bagian kiri dan *motor* sebelah kanan akan dikurangi sehingga mengurangi gaya angkat yang akan menyebabkan *quadcopter* bergerak kearah kanan, serta akan bergerak kearah kiri apabila sebaliknya. Gerakan ini mempunyai tumpu pda smbu Y.



Gambar 2.1 Gerakan Roll *Quadcopter*

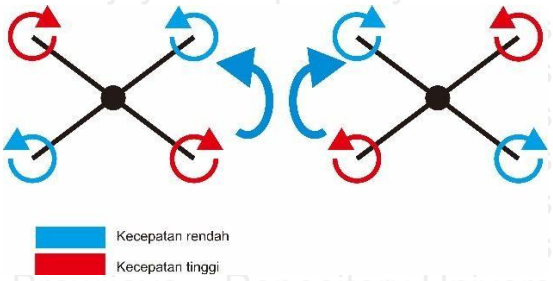
2. *Pitch* merupakan gerak dari *quadcopter* kearah bagian depan dan bagian belakang. Gambar2.2 menjelaskan sama seperti pada gerakan *roll qudcopter* namun jika pada *pitch* jika ingin bergerak maju makan bagian depan motor dari quadcopter akan berkurang intensitas dari gerakanya dan begitu juga bila ingin bergerak ke arah belakang maka intensitas gerak motor bagian depan akan dikurangi. Gerakan ini mempunyai tumpu pada smbu Y



Gambar 2.2 Gerakan pitch *quadcopter*

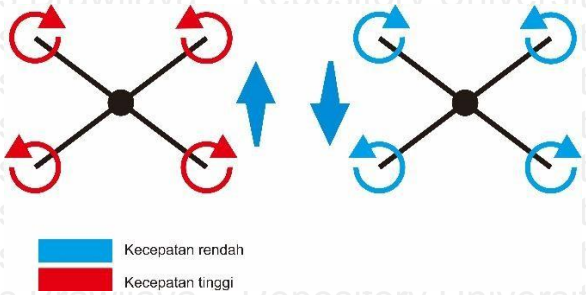
3. Yaw merupakan gerak putar dari *quadcopter* yang berfungsi untuk gerak kiri dan kanan. Jika ingin *quadcopter* bergerak ke arah kiri makan intensitas bagian depan motor sebelah kanan kan dipercepat begitu juga sebaliknya jika ingin bergerak ke arah kanan makan intesitas motor

bagian depan sebalah kiri akan ditambah. gerakan ini bertumpu pada sumbu Z.



Gambar 2.3 Gerakan *yaw quadcopter*

4. Gaz adalah pergerakan naik dan turun dari *quadcopter*. pada gambar 2.4, jika ingin bergerak ke atas makan semua kecepatan dari intensitas motor akan ditambahkan, begitu juga sebaliknya jika ingin bergerak turun maka intensitas dari seluruh motor dikurangi kecepatanya .



Gambar 2.4 Gerakan throttle/gaz *quadcopter*

2.2.2 Pengolahan Citra

Proses pengolahan citra berisi parameter inputan yang berupa informasi citra serta keluaran yang juga berupa informasi citra. Untuk pengolahan citra digital semua data menjadi 2 dimensi (Pitas, 2000). Pada saat ini proses pengolahan citra telah banyak di terapkan di berbagai kepentingan seperti dalam bidang kesehatan atau dalam bidang robotika. Supaya robot bisa menganalisis suatu objek serta lingkungan yang ada disekitarnya maka robot membutuhkan kamera yang dapat digunakan untuk penangkap gambar. Setelah robot menangkap suatu gambar atau objek maka komputer akan memproses gambar tersebut menjadi pemrosesan citra digital .

Meskipun citra kaya dengan informasi, akan tetapi pengolahan citra seringkali mengalami penurunan pada kualitas pada suatu pemrosesan citra digital, missal banyak terdapat derau pada pemrosesan gambar, warna telalu cerah dapat membuat gambar tidak jelas dan dengan adanya penurunan intensitas citra maka gambar akan lebih sulit untuk di proses pada pemprosesan citra digital. Jika pada pemrosesan citra diagital terlalu banyak derau maka akan semakin sedikit informasi yang akan didapat. Untuk mengurangi penurunan intensitas citra dapat menggunakan cara

memanipulasi dengan menggunakan pemrosesan citra lain agar meiliki pemrosesan citra yang lebih baik. Untuk menuju pemrosesan citra yang lebih baik:

1. Untuk meningkatkan kualitas penampakan diperlukan perbaikan atau modifikasi citra.
2. Dalam pemroses citra perlu dikelompokkan, dicocokkan dan diukur.
3. Pada pemrosesan citra tidak bisa hanya mengandalkan satu pemrosesan citra maka harus digabungkan dengan pemrosesan citra yang lainya.

2.2.3 Color Space

2.2.3.1 *RGB*

Sebagian besar aplikasi menggunakan ruang warna RGB sebagai acuan informasi yang digunakan untuk mengolah citra karena tidak membutuhkan tambahan transformasi agar dapat menampilkan informasi warna pada layar monitor (Swedia & Cahyani, 2015).

Warna *RGB* adalah model warna yang berkonsep pada penambahan warna pokok dalam gambar yaitu *Red, Gren* dan *Blue.* Pada saat ruangan yang tidak terdapat cahaya sama sekali ruang tersebut akan menjadi gelap, dengan tidak adanya cahaya maka ruang tersebut akan bernilai *RGB* (0,0,0) dikarenakan tidak ada intensitas cahaya yang dapat diproses oleh indra penglihatan kita. Jika ditambahkan suatu cahaya yang berwarna merah yang diletakan pada ruangan tersebut maka nilai yang terdapat pada *RGB* adalah (255,0,0). Dengan adanya cahaya merah pada ruangan tersebut maka objek yang akan terlihat hanya yang mempunyai warna merah. Begitu juga jika menggunakan cahaya warna hijau serta biru. (Ericks, 2010)

2.2.4 Thresholding

Menurut Pramana (2015) *Thresholding* merupakan teknik segmentasi dalam citra yang berguna untuk mencari nilai perbedaan yang terletak pada objek utama dan background dalam suatu gambar. *Thresholding* mempunyai 2 patokan dalam melakukan metode *thresholding.* Yang akan menjadi patokan dari metode *thresholding* adalah memilih *mean* atau *median*. Pada dasarnya jika objek yang terdeteksi lebih terang dari latar belakang objek maka objek tersebut mempunyai nilai rata-rata yang lebih terang dari latar berlakang suatu objek.

Jika gambar dari suatu objek masih banyak memiliki derau maka *mean* dan *median* akan memproses gambar tersebut secara maksimal dengan menggunakan *threshold.* Pada pendekatan yang lebih dalam, *thresholding* juga dapat memakai histogram dengan cara menemukan cluster dalam gambar dengan penghitungan dari semua piksel dalam gambar yang menggunakan warna atau intensitas sebagai ukuran, serta puncak dan lembah pada histogram (Pramana, 2015).

2.2.5 Edge Detection (Segmentation)

Edge detection merupakan proses yang berfungsi untuk pencarian tepian objek pada suatu *frame* dengan menggunakan penghitungan perbedaan intensitas yang digunakan sebagai acuan batas-batas suatu objek agar dapat digunakan untuk menampakan tepian pada suatu objek didalam pemrosesan sebuah gambar (citra). Proses *edge detection* dimulai dari pengambilan informasi citra digital yang akan diolah, pendeteksian tepi, segementasi dan *thresholding* (witeti, 2004). Hasil dari pemrosesan *canny edge* pada sebuah gambar (citra digital) sangatlah berpengaruh terhadap pemrosesan citra lebih lanjut seperti analisis dan segmentasi citra. Dengan semakin jelasnya hasil dari sebuah tepian objek maka semakin baik pula hasil dari segmentasi dan analisis citra yang telah dilakukan.

Canny edge detection guna mencari garis tepi pada suatu objek pada pemrosesan citra digital. Operator *canny edge* menggunakan *Gaussian Derivative Kernel* yang bertujuan untuk memperhalus hasil dari deteksi tepi. Metode yang paling tepat untuk pendeteksian tepi adalah *canny edge detection* karena dengan menggunakan metode ini akan mendapatakan hasil deteksi tepian yang lebih jelas dan untuk meminimalkan *noise* yang terdapat pada hasil deteksi.

Dalam menggunakan metode tepi garis dari *canny* akan melalui beberapa tahapan, yaitu :

1. Untuk mengurangi derau suatu objek dapat menggunakan metode *averaging* dan *Gaussian Blur. Averaging* digunakan untuk mencari mean dari keseluruhan *pixel*. *Pixel* yang digunakan adalah 3x3

$$K = \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \quad (2.1)$$

Pada *Filter Gaussian,* untuk menghilangkan noise pada tepian objek digunakan sebuah matrix 5x5.

— (2.2)

[ ]

2. Setelah penghilangan *noise* dari *gaussian blur*, dilakukan penghitungan *intensitas gradient* yang digunakan untuk mencari nilai *Gx* dan *Gy* pada tepian objek.

[ ] (2.3)

[ ] (2.4)

√———— (2.5)

3. Pada akhir proses dari metode *canny edge detection* akan dilakukan proses *thresholding* dengan tiga kondisi, yaitu :

a. If *gradient pixel* > dari *high threshold*, maka *pixel* akan dianggap tepi objek.

b. If *gradient pixel* < dari *low threshold*, maka *pixel* akan dianggap 0 atau bukan tepi.

c. Jika *gradient pixel* diantara *high threshold* dan *low threshold* dianggap tepi jika menyatu semua dengan *pixel* yang telah dianggap sebagai tepi.

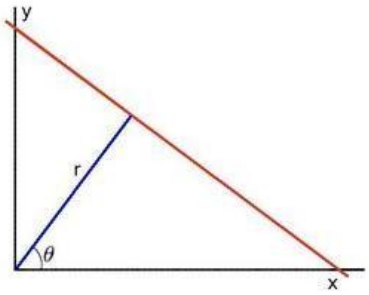
2.2.6 Hough Transform

Hough transform merupakan algoritma pengolahan citra yang dapat digunakan sebagai pendeteksian objek yang memiliki bentuk kurva, seperti lingkaran, lonjong, garis maupun parabola dengan cara penghitungan sudut satu titik tepian kepada titik tepian yang lain. Keuntungan penggunaan *hough transform* adalah memungkinkan untuk mengatur panjang suatu garis yang dapat dideteksi serta batas celah untuk penentuan garis.

Rumus pendeteksian objek pada *hough transform* memiliki perbedaan tergantung dengan jenis objek yang terlihat, seperti pada persamaan 2.6 merupakan rumus untuk pendeteksian objek berupa garis.

(2.6)

Variabel x dan y pada persamaan 2.6 merupakan koordinat piksel dari hasil pendeteksian tepian, sedangkan (*theta*) merupakan sudut yang terbentuk dari sumbu xy dan garis yang menghubungkan titik asal dengan titik terdekat, dan r merupakan jarak asal titik menuju titik terdekat pada garis lurus. Seperti diilustrasikan pada Gambar 2.5.



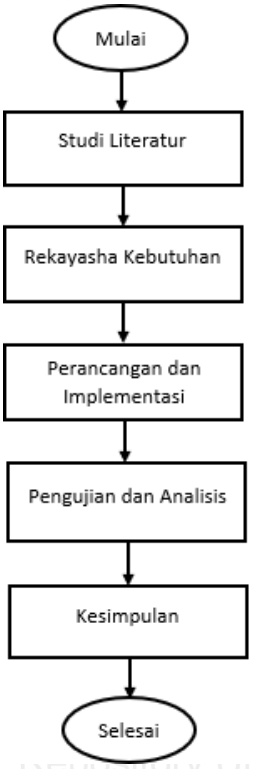
Gambar 2.5 Ilustrasi Hough Transform

Sumber : (OpencvDev, 2011)

BAB 3 METODOLOGI

3.1 Metode Penelitian

Pada bagian nii digunakan untuk menjabarkan mengenai step-step dalam menyusn sbuah penelitian, meliputi Rekayasa Kebutuhan, Perancangan dan Implementasi, Pengujian dan Analisis, Kesimpulan. rekayasa kebutuhan merupakan langkah pertama guna menyusun penelitian ini. di dalam rekayasa kebutuhan akan di jelaskan apa saja bahan atau alat yang dibutuhkan didalam penelitian. Selanjutnya pada perancangan dan implementasi sistem digunakan untuk menjabarkan tentang rancangan dan penerapan sistem yang terdapat pada penelitian ini. Setelah menjabarkan tntang rancangan dan penerapan sistem akan dilakukan uji dan analisi sistem, di fase ini bertujuan guna menguji sistem sekaligus menganalisis hasil dari sistem. Pada fase terakhir akan mendapatkan hasil dari semua tahapan yang disebut dengan kesimpulan.



Gambar 3.1 Alur Metodologi

3.2 Rekayasa Kebutuhan

Tentang rekayasha kebutuhan menjelaskan tentang segala yang menjadi kebutuhan dari pengguna guna membangun system. kebuttuhan pengguna, kebutuhan sistem, serta kebutuhan fungsional dan kebutuhan non-fungsional merupakan rencana yang diperlukan aga sistem biisa berintekasi oleh user. Kemudian dalam tahapan kebutuhan yang dibutuh kan oleh sistem nantinya akan membahas perangkat apa saja yang akan yang dipterlukan guna

membangun sistem ini. Di pembagian sistem nantinya akan bibagi menjadi 2 bagian yaitu kebutuhan lunak dan keras. Di fase ini akan memaparkan kebutuhan dari perangkat lunak dan keras beserta spesifikasinya

Kemudian pada kebutuhan fungsional berguna untuk memonitoring bagaimana sistem dapat berkerja selaras dengan input yang diberikan dan apa dibutuhkan oleh *quadcopter* agar dapat menghindari *obstacle* dengan cara mendeteksi garis yang terbentuk pada *obstacle.*

Selanjutnya Pada kebutuhan non-fungsional apa saja yang akan menjadi batasan masalah, karakteistik pengguna, lingkup operaso dan ketergantungan yang dimilki oleh sistem.

3.3 Perancangan Sistem dan Implementasi

Bab ini memaparkan tentang perancangan sistem yang akan dijelaskan pada pembahasan tentang pengimplementasian rancangan tersebut. Pengelompokan pada tahap perancangan sistem meliputi perancangan pada komunikasi sistem, deteksi *obstacle*, serta pergerakan *quadcopter*.

Pada tahapan ini akan dilakukan penambahan penerapan dari perancangan yang telah dilakukan dengan melalui beberapa proses. Yang pertama akan mengimplementasikan komunikasi terhadap sistem. Pada tahapan digunakan untuk menghubungkan perangkat keras yang digunakan dengan sistem yang telah dibuat. Selanjutnya mengimplementasikan deteksi *obstacle* melalui garis yang terbentuk pada *obstacle*. Pada tahapan ini sistem akan mengambil data menggunakan kamera bagian depan yang terdapat pada *quadcopter*. Pada tahapan sistem dibuat untuk mengatur pergerakan *quadcopter* dapat bekerja secara otomatis. Lalu implementasi pergerakan pada *quadcopter.* Pada tahapan terakhir dilakukan pembuatan sistem yang memungkinkan pendeteksian garis dari *obstacle* sebagai acuan pergerakan otomatis pada *quadcopter.*

3.4 Pengujian dan Analisis

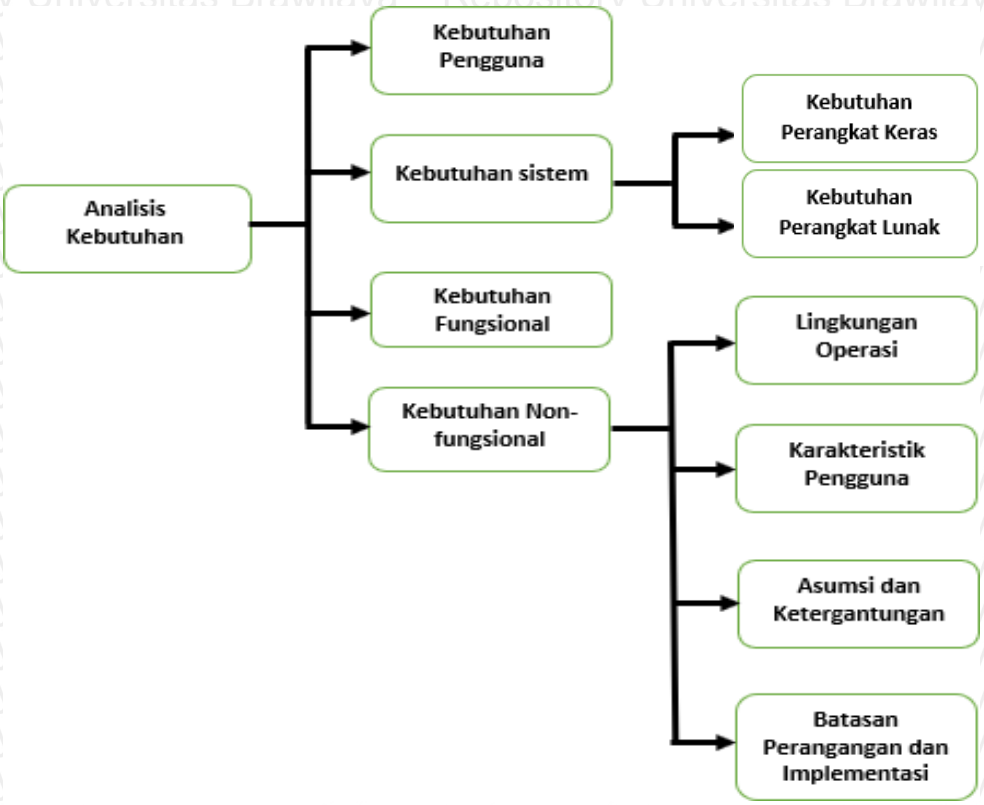
Pengujian dan analisis bertujuan digunakan untuk melihat kinerja sistem secara menyeluruh serta performa dari sistem yang telah dibuat. Sekenario pengujian yang akan dibuat terdiri dari pengujian ketepatan dari gerakan *quadcopter* dan jarak *quadcopter* untuk mendeteksi *obstacle*, dan pengujian waktu sistem untuk menghindari *obstacle*.

3.5 Kesimpulan dan Saran

Kesimpulan dan saran disusun berdasarkan hasil dari tahapan-tahapan yang telah dilalui supaya dapat digunakan sebagai acuan perkembangan yang lebih baik kedepannya.

BAB 4 REKAYASA KEBUTUHAN

Bab ini memaparkan tentang apa saja yang dibutuhkan pada penelitian ini akan dibagi menjadi empat bagian yaitu kebutuhan pengguna yang menjelaskan tentang kebutuhan yang memungkinkan pengguna untuk melihat hasil dari sistem, kebutuhan sistem yang menjabarkan tentang kebutuhan perangkat keras serta perangkat lunak, kebutuhan fungsional dan kebutuhan non-fungsional yang berisi tentang kebutuhan yang harus terpenuhi agar sistem mendapat hasil yang diinginkan, karakteristik pengguna, lingkungan operasi merupakan syarat sistem dapat berjalan optimal, asumsi dan ketergantungan yang menjelaskan tentang batasan berjalannya sistem agar dapat berjalan, serta batasan perancangan dan implementasi yang memaparkan tentang batasan agar sistem berjalan lebih terarah. Analisis kebutuhan sistem dapat dilihat pada gambar 4.1.



Gambar 4.1 Diagram Analisis Kebutuhan

4.1 Kebutuhan Pengguna

Kebutuhan pengguna berisi tentang *user interface* yang dipakai oleh *user* untuk memantau berjalannya sistem serta melihat hasil keluaran sistem seperti pergerakan *drone,* lebar *obstacle*, serta koordingat titik tengah *obstacle* yang ditampilkan pada *window linux.*

4.2 Kebutuhan Sistem

Pada bagian ini, akan menjelaskan kebutuhan perangkat keras serta perangkat lunak yang akan digunakan pada sistem.

4.2.1 Kebutuhan Perangkat Keras

Kebutuhan perangkat keras yang dibutuhkan oleh sistem ini adalah:

1. *Parrot AR.Drone 2.0*

Quadcopter telah banyak berkembang dari tahun ke tahun, mulai dari penambahan fitur-fitur yang mendukung kinerja pergerakan maupun kegunaan lain. Parrot AR Drone merupakan *quadcopter* yang memiliki banyak fitur yang dapat memudahkan pengguna untuk mengontrol *quadcopter* dengan tambahan fitur yang mendukung kontrol manual serta kontrol otomatis seperti dukungan terhadap *remote control*, pengolahan citra, penggunaan sensor jarak, dan sebagainya. (SA, 2016).

Spesifikasi teknis pada *Parrot AR.Drone 2.0* adalah ditunjukkan pada

Tabel 4.1 Spesifikasi *Parrot Ar.Drone 2.0*

No	Parameter	Nilai
1.	Dimensi	517x517 milimeter dengan *blade protector*, 451x451 tanpa *blade protector*
2.	Kamera	HD 720 *pixel*(Video),JPEG(*Photo*)
3.	*Processor*	ARM Cortex 1GHz 32 bit, 1 GB RAM
4.	Sistem Operasi	Linux
5.	Video DSP	800 MHz
6.	Aplikasi	AR Freeflight 2.0 (Android)
7.	Konektivitas	*Wi-fi* (Jarak maksimal 50 meter)
8.	Baterai	LiPo (1.100 mAh / 11,1 volt)
9.	Durasi Baterai	12 menit

Berdasarkan spesifikasi dari Tabel 4.1 *Parrot AR Drone* 2.0 dipilih pada penelitian ini karena memiliki kamera bawaan dan mendukung sifat *open source.*

4.2.2 Kebutuhan Perangkat Lunak

Kebutuhan perangkat lunak yang dibutuhkan pada pembuatan sistem ini adalah sebagai berikut.

1. Sistem Operasi *Linux Ubuntu* versi 14.04
   Ubuntu merupakan salah satu distro *linux* yang berbasiskan *Debian* dan bersifat *Open Source*. *Ubuntu* kebanyakan digunakan untuk penggunakan

sebagai perorangan atau pribadi. Pada *ubuntu* versi 14.04 juga memiliki fitur ubuntu server guna memperbarui *OS* tersebut sesuai dengan kebutuhan penggunanya serta telah banyak dipergunakan secara luas. Pada penelitian ini menggunakan *ubuntu* 14.04 dikarenakan dukungan ROS pada versi 14.04 lebih stabil daripada versi lainnya dikarenakan dukungan library yang belum lama dikembangkan.

2. *OpenCV*

   OpenCV (Open Source Computer Vision) adalah sebuah *library* yang berfungsi sebagai pemograman yang digunakan untuk mengolah gambar secara *real-time. OpenCV* juga mempunyai lisensi *BSD* yang berarti dapat digunakan oleh semua kalangan seperti pengguna akademis secara gratis. Dalam *OpenCV* terdapat beberapa bahasa pemograman seperti *C, C++, Python* dan *Java (Android)* dan mendukung sistem operasi seperti *Linux, Windows, Android, iOS* dan *Mac OS.* Dan algoritma yang terdapat pada *OpenCV* memiliki lebih dari 3000 yang dapat mempermudah pengerjaan sistem dari penggunanya.

3. *ROS*

   Robot Operating System (ROS) merupakan sebuah *platform open source framework* yang digunakan oleh user yang digunakan untuk membuat atau mengunakan sebuah robot. Pada saat ini *ROS* sudah sangat berkembang lebih dari *2000 software packages* yang dapat digunakan oleh pengguna. *ROS* merupakan sebuah *middleware* robot yang berfungsi sebagai penghubung antara perangkat lunak dengan perangkat keras. Dengan adanya *ROS* dapat digunakan untuk mempermudah pengembang guna membuat sebuah robot tanpa harus membuat sebuah kode yang dimulai dari awal. Bagian inti dari *ROS* ditulis menggunakan bahasa *C++,* walau inti *ROS* ditulis menggunakan bahasa *C++* tapi tidak menutup kemungkinan dapat digunakan menggunakan aplikasi yang ditulis dengan menggunakan bahasa *Python atau Lisp.*

4. *Python*

   Python merupakan salah satu jenis bahasa pemograman yang menggunakan mode *interpreted* yang diterjahkan atau dimengerti oleh komputer. Dengan menggunakan bahasa *python* pengguna akan mendapatkan salah satu keuntungan dari bahasa ini yaitu dapat melakukan *maintenance* terhadap kode program yang telah dibuat. Selain dapat maintenance kode program bahasa python juga dapat digunakan pada berbagai Operating System yang salah satunya dapat digunakan pada Operating System Linux.

5. *Geany*

   Geany merupakan sebuah *software* aplikasi komputer yang digunakan penggunanya untuk membuat atau mengubah *file text* yang berupa *plain text* atau kode program. *Geany* merupa *software* aplikasi yang sangat sederhana dan sangat ringan yang dikhususkan untuk beberapa jenis bahasa pemrograman dengan fitur, *type scripting*.

4.3 Kebutuhan Fungsional

Kebutuhan fungsional merupakan penjelesan bagaimana sistem yang akan dibuat untuk mendapatkan hasil *output* sesuai dengan apa yang diinginkan. Ada pun beberapa faktor dari kebutuhan sistem fungsional yang harus di penuhi antara lain.

1. Sistem harus bisa melewati *obstacle*.
2. Sistem dapat melewati *obstacle* dengan menggunakan kecepatan dan jarak yang berbeda-beda.
3. Sistem harus bisa mengambil gambar dari *obstacle* dengan menggunakan kamera.
4. Sistem harus bisa mendeteksi garis yang terbentuk dari tepi *obstacle*
5. Sistem dapat menampilkan beberapa informasi dari *quadcopter* seperti kapasitas baterai, kecepatan, lebar *obstacle* dalam bentuk *pixel* dan waktu.

4.4 Kebutuhan Non-Fungsional

Kebutuhan non-fungsional akan menjelaskan tentang apa yang akan menjadi batasan dalam perancangan sistem. Adapun kebutuhan non-fungsional dari sistem ini adalah sebagai berikut.

4.4.1 Karakteristik Pengguna

Karakteristik pengguna ditujukan untuk pengendali *quadcopter*. Dengan adanya sistem seperti ini diharapkan dapat mempermudah dalam melakukan sistem navigasi agar dapat mengindari *obstacle* secara otomatis sehingga meminimalisirkan kerusakan pada *quadcopter*.

4.4.2 Lingkungan Operasi

Untuk menjalankan sistem dibutuhkan persyaratan lingkungan operasi sebagai berikut.

1. *Obstacle* dengan tinggi maksimal 1.5 m dan lebar 50 cm.
2. Untuk menerbangkan *quadcopter* akan diperlukan ruangan setinggi minimal 8 x 8 x 4 m agar penerbangan lebih optimal.

4.4.3 Asumsi dan Ketergantungan

1. Sistem hanya akan berjalan jika komputer telah memiliki *ROS* dan *OpenCV.*
2. Ketika kebutuhan *pixel* telah terpenuhi oleh lebar dari *obstacle,* maka *quadcopter* dapat melewati *obstacle* secara otomatis.
3. Sistem baru akan bisa berjalan jika komputer sudah terhubung dengan *AR.Drone* menggunakan *WI-FI* yang terdapat pada *quadcoper.*
4. Data dari kamera hanya akan diterima oleh komputer yang terkoneksi pada *quadcopter*.

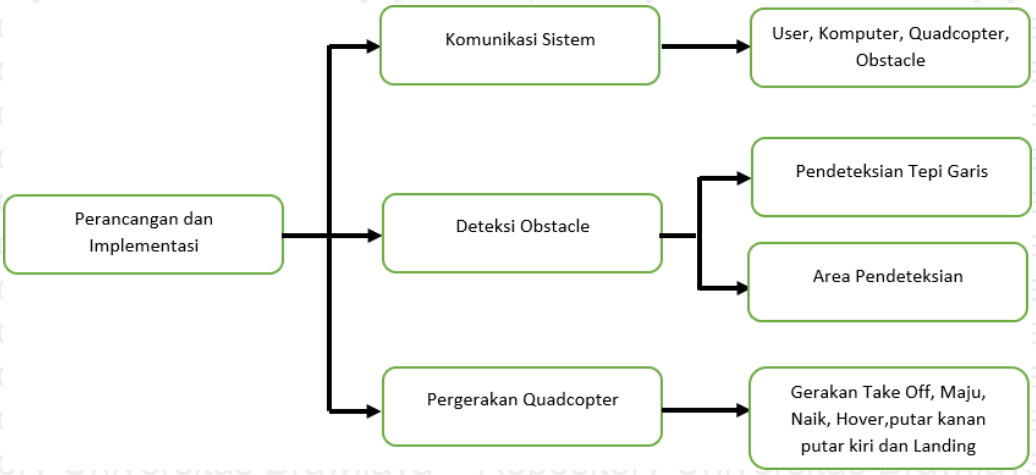
4.4.4 Batasan Perancangan dan Implementasi

Pada bagian ini menjelaskan tentang batasan sistem sehingga membuat penelitian menjadi lebih terarah. Batasan pada penelitian sebagai berikut :

1. Pengujian q*uadcopter* dilakukan di dalam ruangan kosong
2. Maksimal tinggi dari obstacle tersebut adalah 1.5 m dan lebar dari *obstacle* adalah 50 cm.
3. Gerakan pada *quadcopter* yaitu *takeoff, landing, hover,* naik, putar kanan, putar kiri dan maju.
4. *Quadcopter* akan menghindari *obstacle* secara otomatis ketika *obstacle* dapat terdeteksi sesuai dengan *pixel* yang telah ditentukan.
5. Jarak antara *quadcopter* dan *obstacle* tidak lebih dari 6 m.

BAB 5 PERANCANGAN DAN IMPLEMENTASI

Pada bab ini memaparkan tentang perancangan dan implementasi pada penelitian ini yang akan melalui beberapa langkah seperti pada Gambar 5.1. Tahapan yang akan dilalui adalah perancangan komunikasi sistem yang menjelaskan tentang hubungan antar komponen sistem seperti user, komputer, *quadcopter,* dan *obstacle*, kemudian perancangan serta implementasi pendeteksian obstacle yang menjelaskan tentang pendeteksian objek yang dibagi menjadi dua tahapan yaitu deteksi tepi dan pencarian garis serta penghitungan lebar piksel garis. Pada tahap akhir yaitu penentuan pergerakan *quadcopter* seperti *take off* ketika sistem pertama kali dijalankan, maju ketika garis memiliki lebar yang ditentukan, naik setelah garis memiliki lebar melebihi yang ditentukan, *hover* ketika tidak mendeteksi garis, serta putar kanan, putar kiri yang berfungsi untuk menstabilkan arah pergerakan *quadcopter*, dan *landing* ketika sistem berhenti berdasar data yang diperoleh dari tahapan sebelumnya.

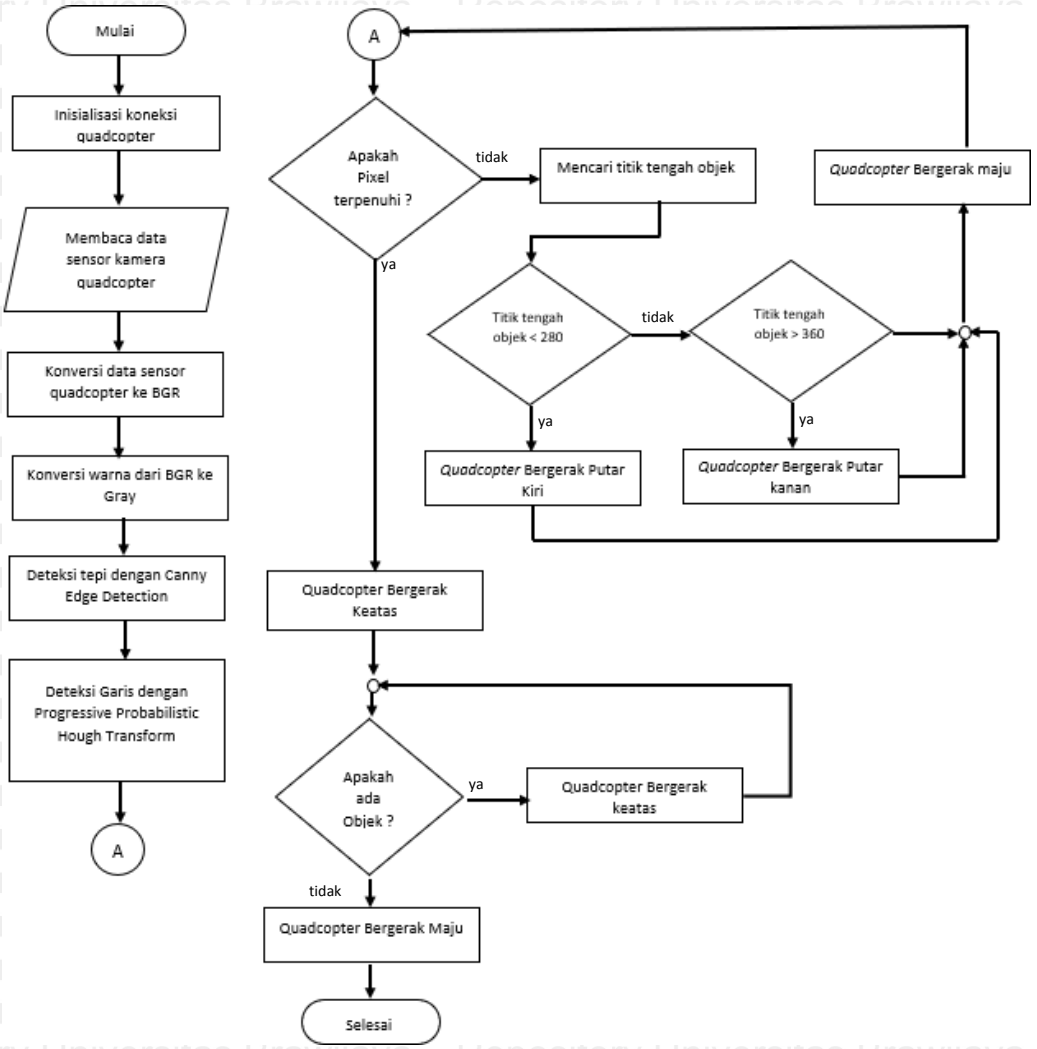


Gambar 5.1 Tahapan Perancangan dan Implementasi

Tahapan perancangan dan implementasi akan membangun sebuah hubungan yang membentuk keseluruhan proses sistem yang akan dipaparkan pada tabel 5.1.

Secara umum, diagram alir sistem akan melakukan proses inisialisasi koneksi antara komputer dan *quadcopter,* kemudian akan dilakukan pembacaan kamera *quadcopter*. Hasil pembacaan kamera akan dimasukkan pada *pre-processing* yaitu *convert RGB to Gray* dan *filtering* menggunakan *gaussianblur*. Hasil dari *pre-processing* akan dimasukkan pada proses pencarian tepian sehingga dapat diolah oleh *Hough Transform* untuk dilakukan pencarian garis. Setelah itu akan dilakukan perhitungan sudut garis dengan ketentuan sudut berada diantara 85° sampai 95°. Setelah mendapatkan nilai sudut maka proses selanjutnya adalah mencari lebar dari *obstacle,* dengan cara menghitung selisih

pada 2 garis yang telah memenuhi sudut. selanjutnya apakah *pixel* sudah memenuhi kebutuhan yang telah ditentukan. Jika belum memenuhi maka *quadcopter* akan mencari koordinat titik tengah *obstacle* untuk menentukan pergerakan *quadcopter* selanjutnya. Jika titik tengah objek kurang dari 280 maka *quadcopter* bergerak putar kiri sedangkan *quadcopter* akan bergerak putar kanan jika titik tengah objek lebih dari 360, kemudian *quadcopter* akan bergerak maju. saat lebar dari *obstacle* terpenuhi maka *quadcopter* akan bergerak ke atas untuk melewati *obstacle.* Jika sistem tidak mendeteksi adanya *obstacle* maka



quadcopter akan bergerak maju secara otomatis untuk melewati *obstacle*.

Gambar 5.2 *Flowchart* sistem pengikut garis *quadcopter*

Pada Tabel 5.1 merupakan penjelasan dari kode yang terdapat pada Gambar 5.2.

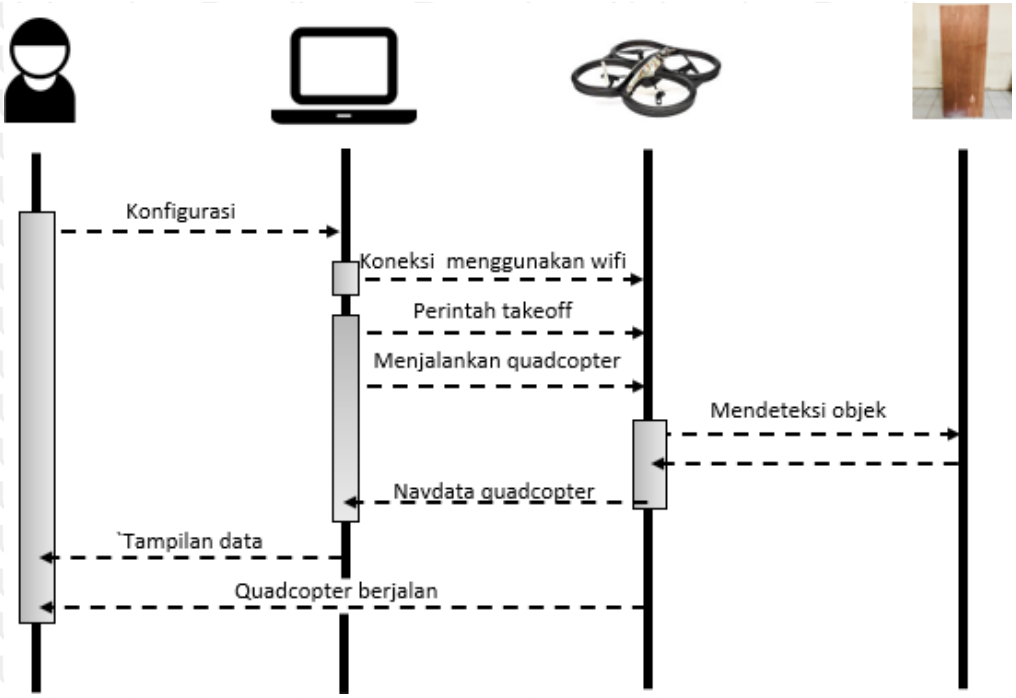
Table 5.1 Hubungan antar proses pada sistem

***Frame* Kamera**	**Gerakan *Quadcopter***
Jika 95 > *angel* > 85 dan koordinat b-a > dari nilai yang telah ditentukan.	Gerak Naik
Jika 95 > *angel* > 85, 360 > koordinat titik tengah objek > 280 dan koordinat b-a < dari nilai yang telah ditentukan. Jika setelah gerak naik, pada *frame* tidak ditemukan sebuah objek.	Gerak Maju
Jika 95 > angel > 85, koordinat b-a < dari nilai yang telah ditentukan dan Koordinat titik tengah objek < 280.	Putar kiri
Jika 95 > angel > 85, koordinat b-a < dari nilai yang telah ditentukan dan koordinat titik tengah objek > 360.	Putar kanan
Jika pada *frame* tidak menemukan objek.	*Hover*

5.1 Komunikasi Sistem

5.1.1 Perancangan Komunikasi Sistem

Sistem ini akan mengolah data sesuai yang dijelaskan pada Gambar 5.3, pertama konfigurasi komputer oleh pengguna yang akan digunakan sebagai *flight control* yang akan dihubungkan pada *quadcopter*. Kemudian komputer dihubungkan dengan *quadcopter* menggunakan koneksi *Wi-Fi* yang telah dipancarkan oleh *quadcopter.* Ketika komputer telah terkoneksi, komputer akan dapat melakukan perintah *takeoff* untuk menjalankan *quadcopter* agar dapat mengambil data objek melalui kamera yang tersedia. Kemudian *quadcopter* akan mengirim data objek kepada komputer sehingga dapat diolah untuk menentukan pergerakan *quadcopter* selanjutnya. Kemudian komputer menampilkan data yang didapat *quadcopter* serta hasil yang telah diproses oleh sistem kepada pengguna.



Gambar 5.3 Alur pertukaran data pada sistem

5.1.2 Implementasi Komunikasi Sistem

Bagian ini menjelaskan tentang komunikasi sistem antara komputer dan *quadcoptere* yang ditunjukan pada Gambar 5.4. komputer melakukan koneksi pada *quadcopter* menggunakan jaringan *Wi-Fi* yang dipancarkan oleh *quadcopter*.

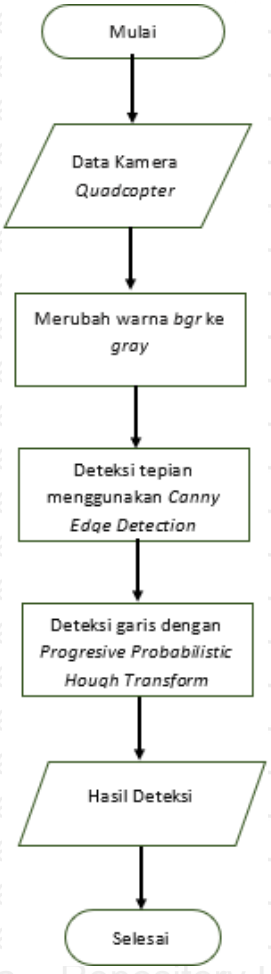
Gambar 5.4 Implementasi komunikasi sistem

5.2 Deteksi Garis

5.2.1 Perancangan Deteksi Obstacle

Data kamera *quadcopter* akan diambil melalui *ROS* yang akan diolah menggunakan *OpenCV* seperti dalam perancangan deteksi *obstacle.* Data tersebut kemudian akan diolah seperti pada Gambar 5.5.

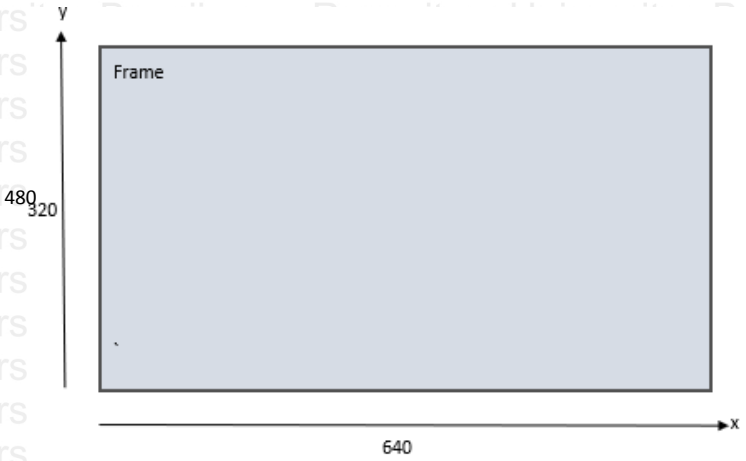
Seperti pada Gambar 5.5 data yang diambil melalui ROS akan diolah dengan beberapa metode pengolahan citra diantaranya perubahan warna *bgr* ke *gray*. Pada proses perubahan warna *bgr* ke *gray* digunakan untuk mempermudah pendeteksian terhadap objek yang memiliki warna dengan cara menghilangkan derau yang terdapat pada suatu objek dengan menggunakan *blur* guna mendeteksi tepian garis. Proses selanjutnya adalah dengan menggunakan *canny edge detection* pada proses ini digunakan untuk meminimalkan adanya derau yang terdapat pada tepi objek yang akan dideteksi. Setelah proses *canny edge* adalah *hough transform.* proses ini bertujuan guna meningkatkan proses sebelumnya. Setelah itu dilakukan pembacaan garis menggunakan garis yang berdasarkan pada koordinat garis berada. Kemudian hasil akan ditampilkan dalam proses pergerakan *quadcopter.*



Gambar 5.5 Alur pendeteksian garis

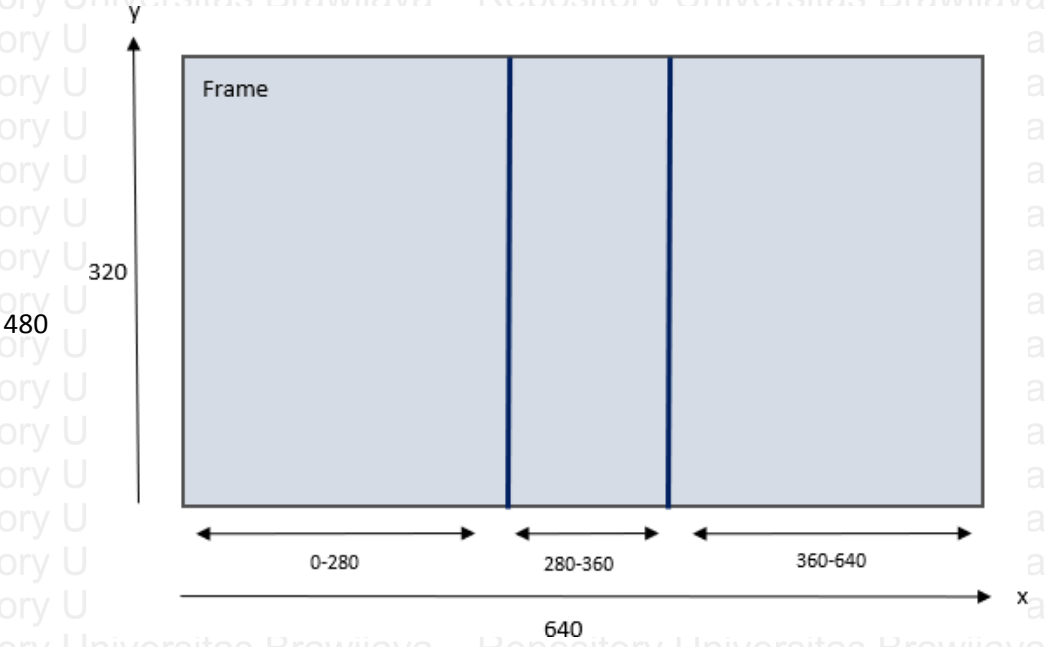
5.2.1.1 Area Pendeteksian

frame yang digunakan untuk pendeteksian objek *obstacle* berukuran 480x640 yang diambil menggunakan kamera depan *quadcopter* yang akan digunakan pada pendeteksian objek. Area pendeteksian dapat dilihat pada Gambar 5.8.



Gambar 5.6 Area pendeteksian

Ukuran yang terdapat pada *frame* nantinya akan digunakan untuk mempermudah proses mendeteksi *obstacle* dan akan di proses lagi menjadi 3 bagian. Bagian kiri adalah *pixel* 0-280, dan pada bagian tengah merupakan *pixel* 280-360 dan bagian kanan merupakan pixel 360-640. Pada proses pembagian *frame* ini hanya akan menggunakan sumbu x dikarenakan *quadcopter* hanya akan mendeteksi *obstacle* yang berbentuk vertikal.



Gambar 5.7 Area pendeteksian dengan pembagian letak

5.2.2 RGB ke Grayscale

Mengubah pixel dari RGB ke *Grayscale* dilakukan agar objek yang berwarna tidak sulit untuk dideteksi. Untuk melakukan konversi dari RGB ke *Grayscale* menggunakan *library* dari OpenCV yaitu fungsi *cvtColor* yang memiliki parameter fungsi *CV_BGR2GRAY* dan masukan *frame* merupakan gambar asli dari data kamera depan *quadcopter* yang masih memiliki nilai RGB dan keluaran berupa *frame* hasil konversi RGB ke *Grayscale*.

Dari gambar tersebut didapatkan nilai *luminosity* dengan hasil perhitungan sebagai berikut:

5.2.3 Implementasi Deteksi Garis

Implementasi pendeteksian garis dilakukan dengan pengambilandata dari kamera *quadcopter* yang akan diolah pada langkah selanjutnya seperti pada *source code* baris ke 2 dilakukan pemanggilan data dari kamera *quadcopter* menggunakan ardrone/image_raw seperti pada Tabel 5.1.

Table 5.2 Implementasi kamera *quadcopter*

No	Kode program
1	`self.bridge = CvBridge()`
2	`self.image_sub = rospy.Subscriber("/ardrone/image_raw", Image, self.callback)`
3	`def callback(self,data):`
4	`try:`
5	`cv_image = self.bridge.imgmsg_to_cv2(data, "bgr8")`
6	`except CvBridgeError as e:`
7	`print(e)`

Setelah selesai mengambil data dari kamera kemudian melakukan beberapa proses seperti berikut ini :

1. Mendeteksi tepian

Langkah ini melakukan *filtering* menggunakan *gaussianblur* yang akan digunakan pada *canny edge detection* agar dilakukan pencarian garis tepi dengan masukan frame yang didapat dari kamera depan *quadcopter*. *Thresholding* pada langkah ini menggunakan *low threshold* sebesar 80 dan *high threshold* sebesar 100 seperti terlihat pada Table 5.3.

Table 5.3 Deteksi *canny edge detection*

No	Kode program
1	`blur_gray = cv2.GaussianBlur(img, (5, 5), 0)`
2	`low_treshold = 80`
3	`high_treshold = 100`
4	`edges = cv2.Canny(blur_gray, low_treshold, high_treshold)`

Pada line 1 dalam kode program 5.2 digunakan untuk menghilangkan noise dengan menggunakan metode *gaussian blur* agar pendeteksian tepian pada objek. Untuk mengubah dari *gray* ke *blur* dapat menggunakan fungsi GaussianBlur. Fungsi GaussianBlur memiliki parameter masukan *image gray,* kernel, sigma. Untuk menggunakan *gaussian filter* dengan kernel berukuran 5x5 pada titik koordinat yang berada pada titik baris 150 hingga 154 dan 201 hingga 205 pada titik kolomnya. Pada proses ini dapat menggunakan persamaan 2.2.

Sebagai contoh, pada gambar awal memiliki nilai matriks seperti pada Tabel 5.3 :

Tabel 5.3 Nilai Matrix Gambar Awal

	201	**202**	**203**	**204**	**205**
150	10	15	20	20	132
151	20	7	23	39	136
152	9	5	15	44	132
153	6	5	37	37	136
154	5	10	15	12	131

Untuk proses *gaussian blur* guna mencari nilai pixel 15 yang berada pada koordinat (203,152) sebagai berikut :

Dari perhitungan *gaussian blur* diatas akan mendapatkan nilai K yang digunakan untuk menggantikan elemen *pixel* sebelumnya seperti pada Tabel 5.4.

Table 5.4 Hasil Perhitungan *Gaussian Blur pixel* (203,152)

	201	**202**	**203**	**204**	**205**
150	10	15	20	20	132
151	20	7	23	39	136
152	9	5	**32**	44	132
153	6	5	37	37	136
154	5	10	15	12	131

Perhitungan terus berulang untuk mencari nilai pada setiap *pixel* gambar seperti pada tabel 5.5.

Table 5.5 Hasil Perhitungan *Gaussian Blur* Seluruh *Pixel*

	201	202	203	204	205
150	5,786164	9,566038	20,73585	31,09434	33,08805
151	7,36478	13,01258	29,06918	27,88679	46,59748
152	7,389937	13,79874	32,58491	49,05031	52,10063
153	5,861635	11,7044	28,22013	43,20755	46,10692
154	3,962264	7,886792	19,57233	30,32075	32,4717

Nilai intensitas *gradient* yang didapat dari *pixel* 32 pada koordinat (203,152) dengan menggunakan persamaan 2.3, 2.4, 2.5 adalah sebagai berikut

[ ]

[ ]

[ ]

[ ]

$\sqrt{\sqrt{\sqrt{\quad}}}$ $\sqrt{\quad}$ $\sqrt{\quad}$

144,0138882191575

hasil dari perhitungan manual metode *sobel* seperti pada tabel 5.6.

Table 5.6 Hasil Perhitungan *Sobel pixel* (203,152)

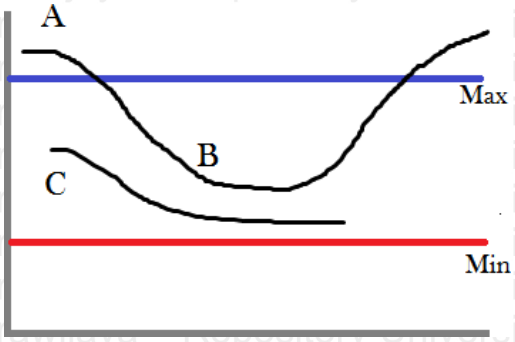
	201	202	203	204	205
150	5,786164	9,566038	20,73585	31,09434	33,08805
151	7,36478	13,01258	29,06918	27,88679	46,59748
152	7,389937	13,79874	144,0139	49,05031	52,10063
153	5,861635	11,7044	28,22013	43,20755	46,10692
154	3,962264	7,886792	19,57233	30,32075	32,4717

Perhitungan akan berulang untuk mencari nilai pada setiap *pixel* gambar pada Tabel 5.7.

Table 5.7 Hasil Perhitungan *Sobel* Seluruh *Pixel*

	201	202	203	204	205
150	59,81639	61,46544	101,1336	411,9927	320,8769
151	36,05551	34,05877	108,074	457,0274	144,0139
152	37,20215	46,17359	144,0139	446,1121	164,0122
153	25,17936	78,23043	108,4159	435,8738	134,3726
154	30,23243	73,5527	121,4578	413,0012	314,9635

Hysteresis merupakan filtering nilai G agar mendapatkan nilai tepi menggunakan dua *treshold* (atas dan bawah).



Gambar 5.8 Hysteresis Thresholding

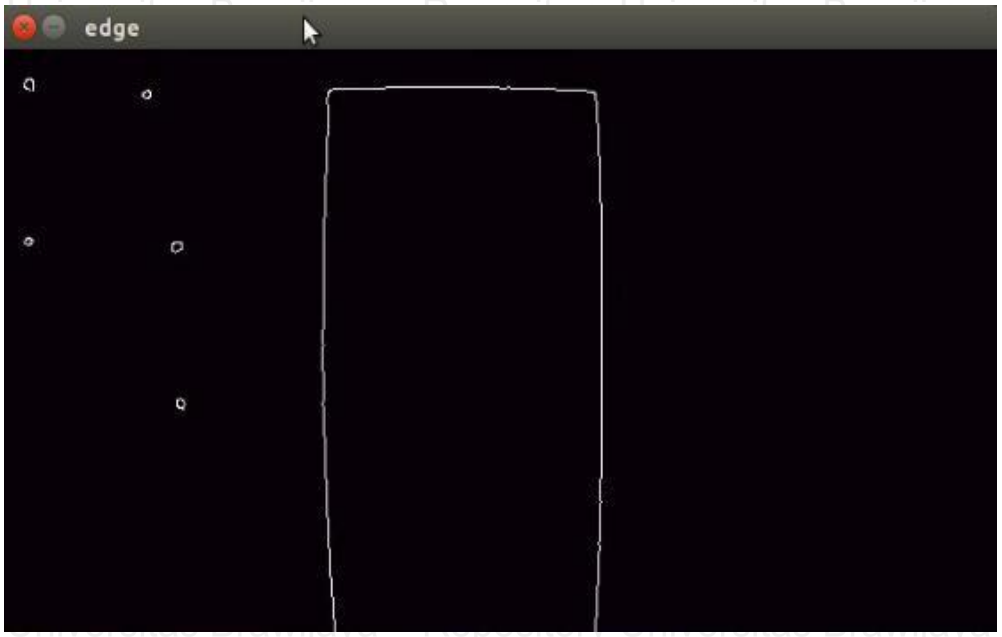
- Jika nilai G > *high_threshold*, maka *pixel* akan dianggap sebagai tepi.
- Jika nilai G < *low_threshold*, maka *pixel* akan dianggap 0 atau bukan tepi.
- Jika nilai G diantara *high threshold* dan *low threshold* dianggap sebagai tepi jika terhubung dengan *pixel* yang telah dianggap sebagai tepi.

Implementasi dari pendeteksian garis tepi dengan menggunakan *canny edge detection* bisa dilihat pada Gambar 5.8.

Hasil *hysteresis* dari *canny edge detection* seperti pada Tabel 5.8.

Table 5.8 Hasil *Hysteresis Pixe*

	201	202	203	204	205
150	0	0	1	1	1
151	0	0	1	1	1
152	0	0	1	1	1
153	0	0	1	1	1
154	0	0	1	1	1



Gambar 5.9 Implementasi Deteksi dengan *Canny Edge detection*

2. Mendeteksi garis dengan *hough transform*

Pada pendeteksian garis menggunakan *progressive probabilistic Hough Transform* akan menggunakan hasil akhir dari perhitungan metode *canny edge detection.* Setelah mendapatkan hasil dari perhitungan *hysteresis* dari Tabel 5.8 maka akan menjadi tolak ukur pada perhitungan *hough transform* seperti pada Gambar 5.10.

Pada perhitungan *hough transform* jika nilai *pixel* bernilai 1 akan dihitung menggunakan persamaan 2.6 seperti :

Persamaan Pixel (203,150)

-197,641

-200,351

-203

Perhitungan *pixel* (203,151)

-197,607

- 200,334

-203

Perhitungan *pixel* (203,152)

-197,572

-200,316

-203

Perhitungan *pixel* (203,153)

-197,537

-200,299

-203

Perhitungan *pixel* (203,154)

-194,502

-200,281

-203

Kurva yang dihasilkan dari perhitungan manualisasi pada metode *hough* transform adalah :

Gambar 5.10 Kurva Transformasi *Hough Transform*

Pada Gambar 5.10 merupakan koordinat (203,150), (203,151), (203,152), (203,153), (203,154) dan akan berpotongan pada titik (180,203). Jika semakin banyak kurva yang berpotongan pada suatu titik yang sama, maka kurva tersebut merupakan transformasi *pixel* dari garis lurus. Banyaknya garis berpotongan

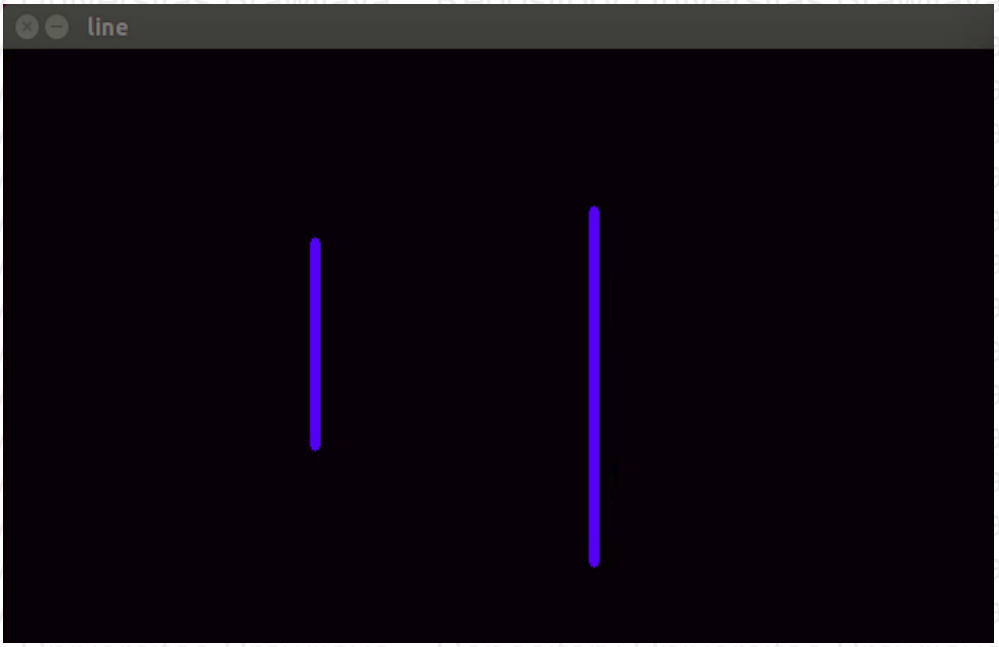
pada satu titik yang sama akan membentuk garis lurus jika banyaknya garis berpotongan lebih dari min_line_length. Dan max_line_gab merupakan celah dari segment garis yang diizinkan untuk membentuk suatu garis.

Masukan yang dipakai pada langkah ini menggunakan hasil keluaran dari proses pendeteksian tepi menggunakan metode *canny edge detection* dengan nilai *threshold* yang dipakai adalah 90, nilai minimal panjang garis yang dapat mendeteksi garis adalah 35 dan maksimal *gap* diantara 2 titik tepian adalah 55. Nilai-nilai ini bisa dilihat pada Tabel 5.10.

Tabel 5.10 Deteksi *progressive probabilistic hough transform*

No	Kode Program
1	`rho = 1`
2	`theta = (np.pi/180)`
3	`treshold = 90`
4	`min_line_length = 35`
5	`max_line_gap = 55`
6	`line_image = np.copy(img)*0`
7	`lines = cv2.HoughLinesP(edges, rho, theta, treshold, np.array([]), min_line_length, max_line_gap)`

Hasil dari pendeteksian garis dari langkah ini dapat dilihat pada Gambar 5.11. Terlihat dua garis yang dapat dipakai sebagai acuan pada pergerakan *quadcopter*.



Gambar 5.11 Implementasi deteksi dengan *hough transform*

3. Mendeteksi garis

pendeteksian garis dilakukan dengan mencari nilai sudut. Untuk (x1,y1) merupakan ujung bawah garis sedangkan untuk (x2,y2) merupakan ujung atas garis. Sementara sudut yang akan digunakan antara 85° sampai 95°. Sudut ini digunakan untuk menggambil garis yang berbentuk vertikal dan akan mengabaikan garis yang membentuk horizontal.

Untuk pembagian sudut seperti pada Tabel 5.11 berikut ini.

Table 5.9 Deteksi sudut

No	Kode program
1	`angle = np.arctan2(y1-y2,x1-x2)*180.0/np.pi`
2	`if (angle<95 and angle>85):`
3	`points.append(((x1 + 0.0, y1 + 0.0), (x2 + 0.0, y2 + 0.0)))`
4	`cv2.line(line_image, (x1, y1), (x2, y2), (255, 0, 0), 5)`

5.3 Pergerakan *Quadcopter*

5.3.1 Perancangan Pergerakan *Quadcopter*

Dalam perancangan gerakan *quadcopter* secara otomatis akan menggunakan tepian dari *obstacle* sebagai masukan dari sistem ini. Informasi dari masukan tersebut akan diolah dengan penghitungan lebar serta koordinat titik tengah garis seperti pada Tabel 5.12 berikut ini.

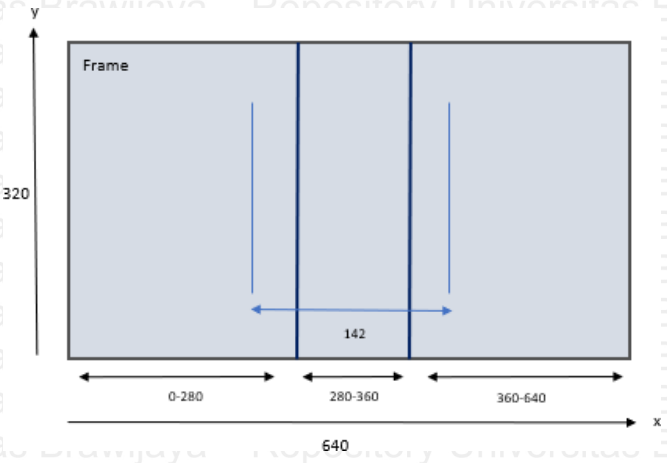
Table 5.10 Menghitung Lebar dan Koordinat Titik Tengah Objek

No	Kode program
1	`lbr = x1-tmp`
2	`tt = lbr/2+tmp`

Pada Tabel 5.11, *variable* x1 berisi nilai koordinat garis kanan yang berasal dari deteksi objek. Dan untuk *variable* tmp berisi nilai koordinat garis kiri yang berasal dari deteksi objek. *Variable* lbr digunakan untuk mencari lebar dari sebuah *obstacle.* Lebar dari sebuah *obstacle* digunakan untuk menentukan naik atau majunya *quadcopter*. *Variable* tt digunakan untuk mencari koordinat titik tengah dari sebuah *obstacle.* Titik tengah digunakan untuk menentukan arah pergerakan kanan atau kiri *quadcopter*.

1. Gerak maju

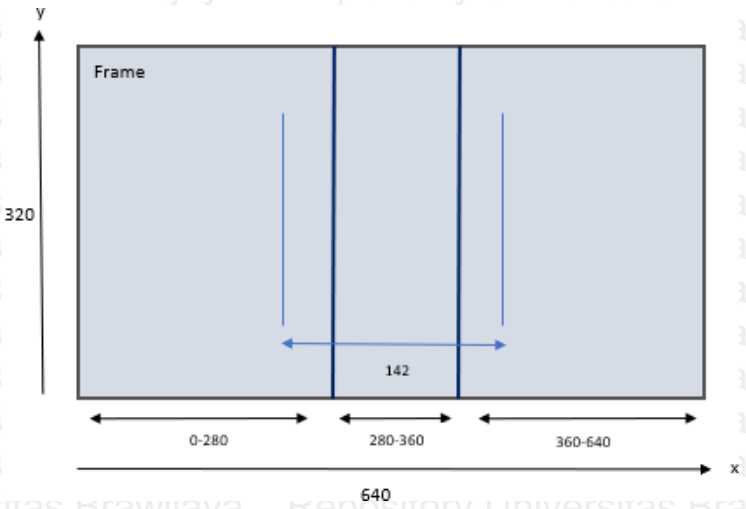
   Q*uadcopter* bergerak maju pada saat titik tengah berada pada *pixel* antara 280-360 dan lebar *pixel* objek kurang dari 142 Gambar 5.12.



Gambar 5.12 Area pendeteksian gerak maju

2. Gerak naik

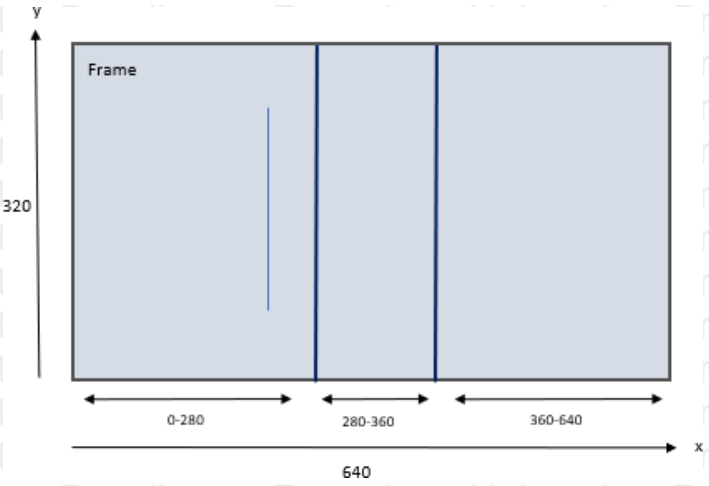
Q*uadcopter* bergerak maju pada saat titik tengah berada pada *pixel* antara 280-360 dan lebar *pixel* objek kurang dari 142 pada Gambar 5.13.



Gambar 5.13 Area pendeteksian gerak naik

3. Hover

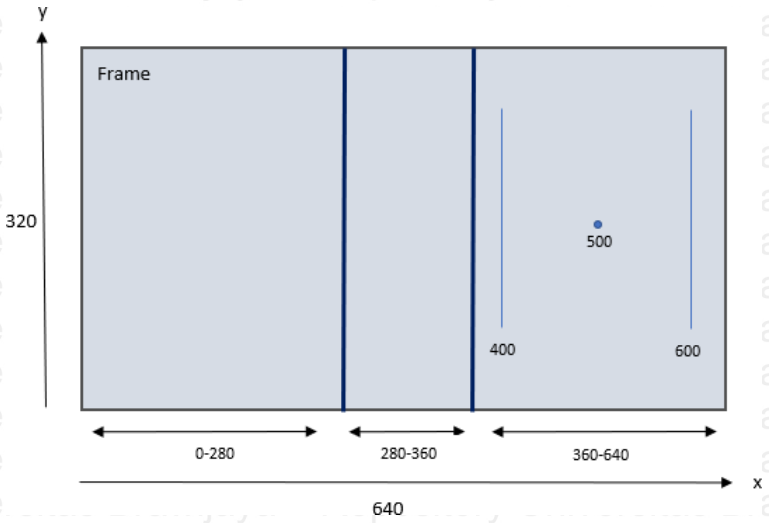
Jika tidak ada garis atau hanya satu garis yang terdeteksi pada *pixel* maka quadcopter akan melakukan pergerakan hover pada Gambar 5.14.



Gambar 5.14 Area pendeteksian gerak hover

4. Putar Kanan

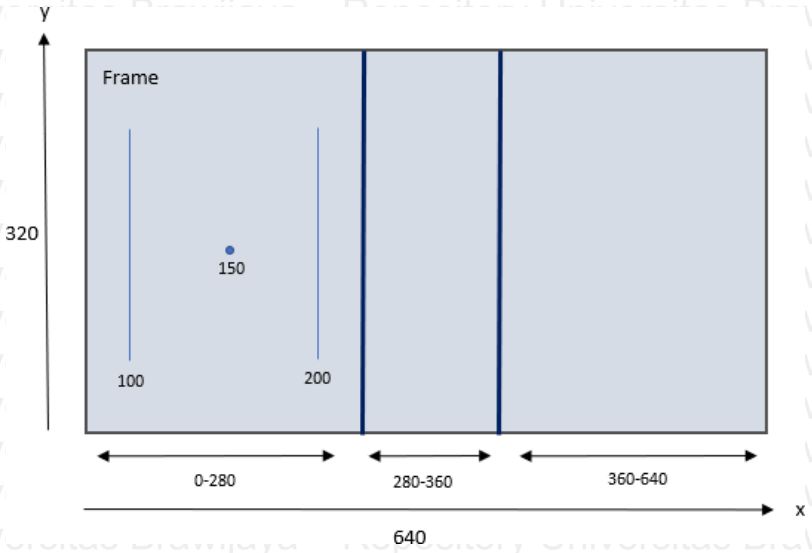
   Q*uadcopter* bergerak putar kanan pada saat titik tengah berada pada *pixel* lebih dari 360 pada Gambar 5.15.



Gambar 5.15 Area pendeteksian gerak putar kanan

5. Putar Kiri

   Q*uadcopter* bergerak putar kanan pada saat titik tengah berada pada *pixel* kurang dari 280 pada Gambar 5.16.



Gambar 5.16 Area pendeteksian gerak putar kiri

BAB 6 PENGUJIAN DAN ANALISIS

Pada bab ini akan membahas pengujian dari sistem yang telah telah dibuat beserta analisis. Dalam pengujian sistem akan menggunakan dua aspek pengujian yaitu ketepatan mendeteksi objek dengan menggunakan kecepatan yang berbeda serta jarak yang berbeda dan pengujian delay sistem.

6.1 Pengujian Ketepatan Pergerakan *quadcopter*

6.1.1 Tujuan pengujian

Pengujian ini adalah untuk mengetahui hasil dari perancangan sistem dapat bekerja sesuai dengan perancangan dan *quadcopter* dapat menghasilkan keluaran yang tepat dalam melakukan penghindaran *obstacle* secara otomatis.

6.1.2 Pelaksanaan Pengujian

Pelaksanaan pengujian dilakukan dengan cara menerbangkan *quadcopter* dengan kecepatan serta jarak yang berbeda. Untuk mengetahui setiap pergerakan dari *quadcopter* dapat dilakukan pemantauan dengan menggunakan terminal *linux* serta menggunakan kamera depan *quadcopter* untuk mengambil *frame* yang telah berisikan objek *obstacle* guna melihat setiap pergerakan *quadcopter*.

6.1.3 Prosedur Pengujian

Pengujian akan dilakukan dengan langkah-langkah sesuai prosedur sebagai berikut

1. Menaruh *obstacle* yang telah di buat sebelumnya pada jarak 6 meter dari *quadcopter*.
2. Persiapan *quadcopter* seperti pemasangan baterai hingga kalibrasi pada *quadcopter* sampai siap digunakan.
3. Memposisikan *quadcopter* di depan *obstacle* yang telah di letakan.
4. Melakukan koneksi antara PC dan *quadcopter* melalui koneksi wifi yang dipancarkan oleh *quadcopter.*
5. Login sebagai root pada *terminal ubuntu* yang akan digunakan sebagai *interface* koneksi ROS dengan drone dengan mengetik “sudo bash”, kemudian masuk pada direktori ROS yang tersimpan pada tum_simulator_ws dengan mengetik “cd tum_simulator_ws” serta mengaktifkan ROS dengan mengetik “source devel/setup.bash”. Kemudian menghubungkan ROS dengan *quadcopter* dengan mengetik .
6. Login sebagai root pada *terminal ubuntu* yang akan digunakan sebagai *interface* program pergerakan otomatis yang telah dibuat dengan mengetik “sudo bash”, kemudian masuk pada direktori ROS yang tersimpan pada

tum_simulator_ws dengan mengetik "cd tum_simulator_ws" serta mengaktifkan ROS dengan mengetik "source devel/setup.bash".

7. Mengedit bb.py untuk mengatur jarak dan kecepatan *quadcopter* dengan nilai yang telah di tentukan. Kemudian mengetik "rosrun mydrone bb.py" untuk melakukan *run* program pada *terminal linux interface* program pergerakan otomatis yang telah dibuat.
8. Mengamati *output* dari kamera *quadcopter* pada terminal untuk mengetahui apakah *quadcopter* dapat bergerak melewati *obstacle* dengan berbagai kecepatan dan jarak yang berbeda.
9. Melakukan pengulangan langkah ke 7 sebanyak lima kali dan mencatat hasil yang telah didapatkan.
10. Melakukan prosedur diatas dengan menggunakan kecepatan 0,03, 0,05, 0,07, 0,09 serta jarak 70 px, 85 px, 115 px, 142 px, 200 px, 294 px, 435 px.

6.1.4 Hasil Pengujian

1. Menggunakan kecepatan 0.03

Pada pengujian ini, jarak yang digunakan berdasarkan pada lebar *pixel obstacle* yang telah didapatkan dari *frame* kamera depan *quadcopter* yang telah diolah menggunakan metode *hough transform.* Hasil yang didapat pada pengujian ini seperti pada tabel 6.1.

Table 6.1 *Ketepatan Gerak* Quadcopter *Menggunakan kecepatan 0.03 m/s*

Jarak (px)	Jarak *quadcopter* menghindari *obstacle* (px)					Rata-rata (px)	Persentase (%)
	Pengujian ke-						
	1	2	3	4	5		
70	80	72	77	98	86	82.6	———
85	95	86	100	89	87	91.4	———
115	117	117	120	132	123	121.8	———
142	169	154	146	144	194	161.4	———
200	275	208	0	218	223	184.8	———
294	421	219	295	372	0	281.4	———

Jarak	Jarak *quadcopter* menghindari *obstacle* (px)					Rata-	Persentase (%)
435	0	0	0	0	0	0	———

2. Menggunakan kecepatan 0.05 m/s

Pada pengujian ini, jarak yang digunakan berdasarkan pada lebar *pixel obstacle* yang telah didapatkan dari *frame* kamera depan *quadcopter* yang telah diolah menggunakan metode *hough transform.* Hasil yang didapat pada pengujian ini seperti pada tabel 6.2.

Table 6.2 *Ketepatan Gerak* Quadcopter *Menggunakan kecepatan 0.05 m/s*

Jarak (px)	Jarak *quadcopter* menghindari *obstacle* (px)					Rata-rata (px)	Persentase (%)
	Pengujian ke-						
	1	2	3	4	5		
70	72	71	114	74	75	81.2	———
85	85	85	92	87	93	88.4	———
115	118	123	116	127	119	120.6	———
142	158	142	143	172	168	156.6	———
200	205	233	201	202	203	208.8	———
294	335	345	301	323	0	260.8	———
435	0	0	0	0	0	0	———

3. Menggunakan kecepatan 0.07 m/s

Pada pengujian ini, jarak yang digunakan berdasarkan pada lebar *pixel obstacle* yang telah didapatkan dari *frame* kamera depan *quadcopter* yang telah diolah menggunakan metode *hough transform.* Hasil yang didapat pada pengujian ini seperti pada tabel 6.3.

Table 6.3 *Ketepatan Gerak* Quadcopter *Menggunakan kecepatan 0.07 m/s*

Jarak (px)	Jarak *quadcopter* menghindari *obstacle* (px)					Rata-rata (px)	Persentase (%)
	Pengujian ke-						
	1	2	3	4	5		
70	71	73	94	76	72	77.2	———
85	86	86	87	89	88	87.2	———
115	130	118	135	143	133	131.8	———
142	158	146	146	159	158	153.4	———
200	226	231	207	220	231	223	———
294	319	316	306	400	300	328.2	———
435	0	0	0	0	0	0	———

4. Menggunakan kecepatan 0.09 m/s

Pada pengujian ini, jarak yang digunakan berdasarkan pada lebar *pixel obstacle* yang telah didapatkan dari *frame* kamera depan *quadcopter* yang telah diolah menggunakan metode *hough transform.* Hasil yang didapat pada pengujian ini seperti pada tabel 6.4.

Table 6.4 *Ketepatan Gerak* Quadcopter *Menggunakan kecepatan 0.09*

Jarak (px)	Jarak *quadcopter* menghindari *obstacle* (px)					Rata-rata (px)	Persentase (%)
	Pengujian ke-						
	1	2	3	4	5		
70	72	71	75	80	77	75	———
85	119	90	86	86	90	94.2	———
115	116	139	118	122	148	128.6	———
142	154	147	153	178	159	158.2	———
200	0	0	0	0	0	0	———
294	0	0	0	0	0	0	———
435	0	0	0	0	0	0	———

6.1.5 Analisis Pengujian

Setelah melakukan pengujian berdasarkan lebar *obstacle* dengan kecepatan yang berbeda-beda, maka akan didapatkan hasil persentase *error* yang digunakan untuk menentukan jarak *optimal* pada *quadcopter* untuk menghindari *obstacle.* Dengan menggunakan rumus persentase eror pada 6.1. Dari rumus 6.1 akan mendapatkan hasil besar nilai eror sistem guna menghitung tingkat akurasi sistem. Setelah mendapatkan hasil dari persentase *error,* untuk mengetahui jarak optimal *quadcopter* guna menghindari *obstacle* dapat menggunakan rumus jarak optimal. dengan rumus 6.2 akan didapatkan jarak optimal dengan lebar *obstacle* 85 pixel dengan kecepatan 0.03 hingga 0.09 berdasarkan dari 6.2.

(6.1)

(6.2)

6.2 Pengujian Estimasi Waktu Gerak Sistem

6.2.1 Tujuan Pengujian

Pengujian ini untuk mendapatkan nilai estimasi waktu yang dibutuhkan *quadcopter* untuk mulai menghindari *obstacle.*

6.2.2 Pelaksanaan Pengujian

Pelaksanaan pengujian dilakukan dengan cara menerbangkan *quadcopter* dengan kecepatan 0.05 serta dengan jarak 6 meter dari *obstacle* untuk mengetahui waktu saat sistem dieksekusi sampai *quadcopter* mulai menghindari *obstacle*. Fungsi import time pada python digunakan untuk mengakses library time yang akan digunakan untuk proses penghitungan waktu. penghitungan waktu dapat dilakukan dengan cara mencari selisih waktu *quadcopter* menghindari *obstacle* dan saat program mulai dijalankan.

6.2.3 Prosedur Pengujian

Pengujian akan dilakukan sesuai prosedur sebagai berikut :

1. Menaruh *obstacle* yang telah di buat sebelumnya pada jarak 6 meter dari *quadcopter*.
2. Persiapan *quadcopter* seperti pemasangan baterai hingga kalibrasi pada *quadcopter* sampai siap digunakan.
3. Memposisikan *quadcopter* di depan *obstacle* yang telah di letakan.
4. Melakukan koneksi antara PC dan *quadcopter* melalui koneksi wifi yang dipancarkan oleh *quadcopter.*
5. Login sebagai root pada *terminal ubuntu* yang akan digunakan sebagai *interface* koneksi ROS dengan drone dengan mengetik “sudo bash”, kemudian masuk pada direktori ROS yang tersimpan pada tum_simulator_ws dengan mengetik “cd tum_simulator_ws” serta mengaktifkan ROS dengan mengetik “source devel/setup.bash”. Kemudian menghubungkan ROS dengan *quadcopter* dengan mengetik .
6. Login sebagai root pada *terminal ubuntu* yang akan digunakan sebagai *interface* program pergerakan otomatis yang telah dibuat dengan mengetik “sudo bash”, kemudian masuk pada direktori ROS yang tersimpan pada tum_simulator_ws dengan mengetik “cd tum_simulator_ws” serta mengaktifkan ROS dengan mengetik “source devel/setup.bash”.
7. Mengedit bb.py untuk mengatur jarak dan kecepatan *quadcopter* dengan nilai yang telah di tentukan. Kemudian mengetik “rosrun mydrone bb.py” untuk melakukan *run* program pada *terminal linux interface* program pergerakan otomatis yang telah dibuat.
8. Mengamati *output* dari kamera *quadcopter* pada terminal untuk mengetahui apakah *quadcopter* dapat bergerak melewati *obstacle* dengan berbagai kecepatan dan jarak yang berbeda.

9. Melakukan pengulangan langkah ke 7 sebanyak lima kali dan mencatat hasil yang telah didapatkan.
10. Melakukan prosedur diatas dengan menggunakan kecepatan 0,05. Serta jarak 70 px, 85 px, 115 px, 142 px, 200 px, 294 px, 435 px.

6.2.4 Hasil Pengujian

Table 6.5 adalah hasil estimasi waktu yang dibutuhkan *quadcopter* pada saat sistem dieksekusi sampai *quadcopter* mulai menghindari *obstacle* dengan 7 jarak yang berbeda dan masing-masing diuji sebanyak 5 kali.

Table 6.5 Estimasi Waktu Gerak Sistem

Jarak (px)	Waktu *quadcopter* menghindari *obstacle* (s)					Rata-rata (s)
	Pengujian ke-					
	1	2	3	4	5	
70	5,193	5,343	4,287	4,492	4,923	4,848
85	6,27	5,078	9,287	6,936	6,462	6,807
115	8,304	7,941	7,646	6,601	14,638	9,026
142	9,179	8,368	10,778	8,798	9,294	9,283
200	8,821	12,257	10,904	8,619	17,125	11,545
294	14,025	9,592	8,382	12,707	9,018	10,745
435	0	0	0	0	0	0

6.2.5 Analisis Pengujian

Hasil pengujian yang didapat, dengan menggunakan perhitungan

(6.3)

Pada Tabel 6.5 dapat dilihat hasil pengujian saat program mulai dijalankan sampai *quadcopter* mulai menghindari *obstacle* pada kecepatan 0.05, bahwa rata-rata waktu yang dibutuhkan oleh *quadcopter* untuk mulai menghindari *obstacle* sebesar 7,464857 detik.

BAB 7 PENUTUP

7.1 Kesimpulan

Diambil dari hasil pengujian pada bab sebelumnya, didapatkan kesimpulan yang sesuai dengan Rumusan Masalah yaitu :

1. Setelah melakukan pengujian ketepatan pergerakan *quadcopter* dengan menggunakan lebar obstacle (px) serta kecepatan (m/s) yang berbeda-beda, maka dihasilkan persentase *error* pada tingkat akurasi pergerakan *quadcopter*. Pada saat *quadcopter* telah mencapai lebar *obstacle* sebesar 85 *pixel* dengan menggunakan kecepatan 0.03, 0.05, 0.07, 0.09 maka dihasilkan *erorr* sebesar 0.07125 %. Sedangkan pada saat *quadcopter* telah mencapai lebar *obstacle* sebesar 435 *pixel* dengan menggunakan kecepatan 0.03, 0.05, 0.07, 0.09 maka dihasilkan *error* sebesar 100%. Sehingga dapat disimpulkan bahwa *quadcopter* dapat menghindari *obstacle* dengan tingkat akurasi *error* paling baik pada saat lebar *obstacle* telah mencapai 85 pixel.
2. Pada pengujian ketepatan pergerakan *quadcopter* dapat disimpulkan bahwa pada saat *quadcopter* bergerak memakai kecepatan 0.03, 0.05, 0.07 dengan satuan meter per detik didapatkan jarak maksimum *quadcopter* dapat menghindari *obstacle* pada saat lebar *obstacle* telah mencapai 294 *pixel*, sedangkan pada saat *quadcopter* bergerak dengan kecepatan 0.09 m/s jarak maksimum *quadcopter* dapat menghindari *obstacle* pada saat lebar *obstacle* telah mencapai 142 *pixel*.
3. Pada pengujian estimasi waktu gerak sistem pada *quadcopter* dengan kecepatan 0.05 satuan meter per detik bahwa *quadcopter* dapat menghindari *obstacle* dengan berbagai jarak *obstacle* dengan waktu rata-rata 7,464857s.

7.2 Saran

Beberapa saran untuk penelitian selanjutnya agar dapat dikembagkan lebih lanjut supaya dapat meningkatkan kinerja sistem untuk kedepannya diantaranya:

1. Pengembangan otomasi navigasi *quadcopter* agar dapat menghindari lebih dari 1 buah *obstacle* serta berbagai bentuk *obstacle.*
2. Penambahan gerakan yang mampu dilakukan *quadcopter* pada navigasi otomatis.
3. Diiharapkan *quadcopter* dapat menghindari *obstacle* tidak hanya dengan naik.

DAFTAR REFERENSI

Boudjit, K. & Larbes, C., 2015. Detection and Implementation Autonomous Target Tracking with a Quadrotor AR.Drone. *International Conference on Informatics in Control, Automation and Robotics (ICINCO),* Volume 02, pp. 223-230.

Brandao, A. S., Martins, F. N. & Soneguetti, H. B., 2015. A Vision-based Line Following Strategy for an Autonomous UAV. *International Conference on Informatics in Control, Automation and Robotics (ICINCO),* Volume 02, pp. 314-319.

Hadi, S. W., Setyawan, G. E. & Maulana, R., 2017. Sistem Kendali Navigasi Ar.Drone Quadcopter Dengan Prinsip Natural. *Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer (J-PTIIK),* Volume 02, pp. 380-386.

Hellmund, A.-M., Wirges, S., Tas, O. S. & Bandera, C., 2016. Robot Operating System: A Modular Software Framework. *Research Center for Information Technology,* pp. 1564-1570.

http://ejurnal.stimata.ac.id/index.php/TI/article/viewFile/110/150

Kusuma, W. A. R. &. I. E., 2012. Perancangan Kontrol Fuzzy-PID pada Pengendalian Auto Take-Off Quadcopter UAV. *JURNAL TEKNIK POMITS,* pp. 1-6.

LaFay, M., 2015. *Drones For Dummies.* Hoboken: John Wiley & sons, Inc..

OpencvDev, 2011. *Hough Line Transform.* [Online] Available at: http://docs.opencv.org/2.4/doc/tutorials/imgproc/imgtrans/hough_lines/hough_lines.html
[Diakses 15 September 2018].

Pallas, F. A., Setyawan, G. E. & Prasetio, B. H., 2017. Sistem Kendali Navigasi Quadcopter Menggunakan Suara Melalui Smartphone dan Arduino dengan Metode Text Processing. *Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer (J-PTIIK),* Volume 02, pp. 732-738.

Pitas, I., 2000. *Digital Image Processing Algorithms and Applications.* canada: John Wiley & Sons, Inc..

Pramana, C. J., 2015. Implementasi Metode Thresholding dan Metode Regionprops Mendeteksi Marka Jalan Secara Live Video. *Psi Udinus,* pp. 1-9.

Purvis, M., 2014. *ROS. org.* [Online]Availableat:http://wiki.ros.org/indigo
[Diakses 05 September 2018].

ROS, 2016. *ROS.* [Online] Available at:http://www.ros.org/
[Diakses 05 September 2018].

Rosario, Y., 2013. Sistem Navigasi Ruangan Quadcopter Dengan Menggunakan Sensor Ultrasonik *ITS.*

SA, P., 2016. *Parrot Official Website.* [Online] Available at: https://www.parrot.com/us/drones/parrot-ardrone-20-elite-edition#parrot-ardrone-20-elite-edition [Diakses 06 September 2018].

Swedia, E. R. & Cahyani, M., 2015. *Algoritma Transformasi Ruang Warna.* Depok: Indie Publishing.

Team, O., 2017. *OpenCV.* [Online] Available at: http://opencv.org/ [Diakses 13 September 2018].

Tzivaras, V., 2016. *Building a Quadcopter with Arduino.* birmingham: Packt Publishing Ltd.

Willard, W., 2006. *HTML A Beginner's Guide.* New York: Mcgraw-Hill Osborne Media.

Winarno, E., 2011. Aplikasi Deteksi Tepi pada Realtime Video Menggunakan Algoritma Canny Detection. *Teknologi Informasi DINAMIK,* pp. 44-49.

Zain, A. B., 2018. Sistem AR Drone Pengikut Garis Menggunakan Algoritma Progressive Probabilistic Hough Transform. *Filkom.*