TESIS – TE142599

PERANCANGAN KENDALI MANIPULATOR *REMOTELY OPERATED VEHICLE* UNTUK MENGAMBIL OBJEK DENGAN MENGGUNAKAN KAMERA SEBAGAI *VISUAL SENSOR*

ERWIN ARDIAS SAPUTRA
07111650040001

DOSEN PEMBIMBING
Ronny Mardiyanto, S.T., M.T., Ph.D.

PROGRAM MAGISTER
BIDANG KEAHLIAN TEKNIK ELEKTRONIKA
DEPARTEMEN TEKNIK ELEKTRO
FAKULTAS TEKNOLOGI ELEKTRO
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA
2018

TESIS – TE142599

PERANCANGAN KENDALI MANIPULATOR *REMOTELY OPERATED VEHICLE* UNTUK MENGAMBIL OBJEK DENGAN MENGGUNAKAN KAMERA SEBAGAI *VISUAL SENSOR*

ERWIN ARDIAS SAPUTRA
07111650040001

DOSEN PEMBIMBING
Ronny Mardiyanto, S.T., M.T., Ph.D.

PROGRAM MAGISTER
BIDANG KEAHLIAN TEKNIK ELEKTRONIKA
DEPARTEMEN TEKNIK ELEKTRO
FAKULTAS TEKNOLOGI ELEKTRO
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA
2018

LEMBAR PENGESAHAN

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

oleh:

Muhammad Qomaruzzaman
07111650040009

Tanggal Ujian : 3 Juli 2018
Periode Wisuda : September 2018

Disetujui oleh:

1. Ronny Mardiyanto, S.T., M.T., Ph.D. (Pembimbing)
   NIP: 198101182003121003
2. Ir. Totok Mujiono, M.Ikom, Ph.D. (Penguji)
   NIP: 196504221989031001
3. Dr.Ir. Djoko Purwanto, M.Eng. (Penguji)
   NIP: 196512111990021002
4. Dr. Muhammad Rivai, ST.,MT. (Penguji)
   NIP: 196904261994031003

Dekan Fakultas Teknologi Elektro

Dr. Tri Arief Sardjono, S.T., M.T.
NIP. 197002121995121001

Halaman ini sengaja dikosongkan

PERNYATAAN KEASLIAN TESIS

Dengan ini saya menyatakan bahwa isi keseluruhan Tesis saya dengan judul "**PERANCANGAN KENDALI MANIPULATOR *REMOTELY OPERATED VEHICLE* UNTUK MENGAMBIL OBJEK DENGAN MENGGUNAKAN KAMERA SEBAGAI *VISUAL SENSOR***" 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, Mei 2018

Erwin Ardias Saputra
NRP. 07111650040001

Halaman ini sengaja dikosongkan

PERANCANGAN KENDALI MANIPULATOR *REMOTELY OPERATED VEHICLE* UNTUK MENGAMBIL OBJEK DENGAN MENGGUNAKAN KAMERA SEBAGAI *VISUAL SENSOR*

Nama mahasiswa : Erwin Ardias Saputra
NRP : 07111650040001
Pembimbing : Ronny Mardiyanto, S.T., M.T., Ph.D.

ABSTRAK

Kebocoran pipa minyak dalam laut menyebabkan terjadinya pencemaran dan kerusakan ekosistem laut. Kebocoran dapat terjadi akibat terjadinya korosi dan keretakan pada pipa. Seperti kebocoran pipa yang terjadi di laut Balikpapan. Kebocoran pipa tersebut menyebabkan rusaknya ekosistem laut di daerah tersebut. Sumber kebocoran pipa minyak dalam air mudah untuk dideteksi karena warna air laut dan minyak sangat berbeda. Berdasarkan permasalahan ini perlu upaya dalam melakukan inspeksi dan perawatan pipa dalam laut secara berkala. Penggunaan penyelam untuk melakukan inspeksi dan memindahkan peralatan didasar laut sangat berbahaya dan beresiko. Oleh karena itu, para peneliti menggembangkan *Remotely Operated Vehicle* (ROV) dengan manipulator yang dapat membantu para penyelam dalam melakukan inspeksi dan perawatan pipa dalam laut. ROV adalah robot dalam air yang dikendalikan dari jarak jauh menggunakan kabel diatas permukaan air. Penambahan manipulator pada ROV bertujuan untuk mengambil objek dalam laut yang sulit untuk dilakukan oleh seorang penyelam. Pengambilan objek dengan memanfaatkan kamera sebagai sensor adalah proses mengendalikan ROV dan untuk mendekat ke arah objek kemudian menggerakan manipulator untuk mengambil objek menggunakan informasi visual. Proses ini dilakukan dengan cara mendeteksi bentuk objek berdasarkan warna. kemudian secara otomatis menggerakan ROV mendekat dan mengambil objek. Hasil pengujian pendeteksian objek menghasilkan akurasi sebesar 96.8% pada jarak 60 cm. Jarak maksimum objek yang dapat dideteksi adalah 2.7 m. Kontrol posisi dengan menggunakan metode kontrol PID menghasilkan *settling time* rata-rata 18 detik dan ROV mampu mengangkat beban maksimal 50 gram.

Kata kunci: *Remotely Operated Vehicle, visual sensor, settling time*

Halaman ini sengaja dikosongkan

DESIGN CONTROL OF REMOTELY OPERATED VEHICLE MANIPULATOR TO TAKE AN OBJECT USING CAMERA AS VISUAL SENSOR

Student Name	: Erwin Ardias Saputra
Student Identity Number	: 07111650040001
Supervisor	: Ronny Mardiyanto, S.T., M.T., Ph.D.

ABSTRACT

Oil spill is on of major causes of environmental destruction. Corosions or fracture on the oil pipes are the cause of the oil spill. Oil spill in Balikpapan is an example of pipeline fracture. Oil spill cause damage to marine ecosystems. The source of oil spill is easy to detect because the color of seawater and oil is different. That accident and many other oil spill accident show the importance of oil pipes maintenance. This maintanance includes regular inspections on the pipelines. Inspection of these pipelines is a dangerous job, even for profesionals. Researchers have developed an underwater robot called Remotely Operated vehicle (ROV) that is equiped with manipulators to help divers do the job. ROVs are controlled from surface through tether. Manipulators are tools to grab underwater objects that are not possible to be grabbed by divers. Taking the object by utilizing camera as visual sensor is the process of controlling the ROV to get closer to the object then move the manipulator to retrieve object using visual information. This process is done by detecting object shape based on its color. The ROV is then approaching the object grab it. The experiment results of detection rate of the object placed 60 cm from camera using this method is 96.8%. The maximum distance of detection is 2.7 m. The position control using this method gives 18 s of setting time. The manipulator used in this experiment is able to lift object with the weight of up to 50 g.

Key words: Remotely Operated Vehicle, visual sensor, settling time

Halaman ini sengaja dikosongkan

KATA PENGANTAR

Puji syukur penulis panjatkan kepada Allah Subhanahu Wa Ta'ala atas segala limpahan rahmat dan hidayahnya kepada penulis sehingga penulis dapat menyelesaikan tesis yang berjudul "Perancangan Kendali Manipulator *Remotely Operated Vehicle* Untuk Mengambil Objek Dengan Menggunakan Kamera Sebagai *Visual Sensor*". Tesis ini disusun untuk memenuhi persyaratan untuk memperoleh gelar Magister Teknik (M.T) pada bidang keahlian Teknik Elektronika, Departemen Teknik Elektro, Fakultas Teknologi Elektro, Institut Teknologi Sepuluh Nopember.

Tesis ini tidak dapat tersusun dengan baik tanpa bimbingan, bantuan dan dukungan dari banyak pihak yang diberikan kepada penulis. Oleh karena itu penulis memberikan ucapan terimakasih dan penghargaan yang sebesar-besarnya kepada semua pihak yang membantu dalam menyelesaikan tesis ini dengan segala kerendahan hati penulis menyampaikan terima kasih kepada:

1. Ibunda, Ayahanda, Kakak, dan Adik tercinta atas segala bantuan, dukungan, dan doa hingga tesis ini dapat terselesaikan dengan baik.
2. Bapak Ronny Mardiyanto, S.T., M.T., Ph.D selaku dosen pembimbing yang telah banyak memberikan saran, bantuan, serta bimbingan.
3. Bapak Ir. Totok Mujiono, M.Ikom, Ph.D, Dr. Muhammad Rivai, S.T., M.T dan Ir. Djoko Purwanto, M.Eng., Ph.D selaku dosen penguji dalam ujian tesis yang telah memberikan saran-saran yang sangat bermanfaat bagi penulis.
4. Rekan-rekan S2 dan rekan-rekan Lab A206 yang telah banyak membantu dalam penyelesaian tesis ini.
5. Seluruh pihak yang membantu penulis dalam menyelesaikan tesis ini.

Menyadari bahwa tesis ini masih jauh dari kata sempurna, maka kritik dan saran sangat penulis harapkan untuk perbaikan dimasa datang. Penulis berharap agar tesis ini dapat bermanfaat.

Surabaya, 26 Juli 2018

Penulis

DAFTAR ISI

LEMBAR PENGESAHAN iii
PERNYATAAN KEASLIAN TESIS v
ABSTRAK vii
ABSTRACT ix
KATA PENGANTAR xi
DAFTAR ISI xiii
DAFTAR GAMBAR xvii
DAFTAR TABEL xix
BAB 1 PENDAHULUAN 1
1.1 Latar Belakang 1
1.2 Rumusan Masalah 3
1.3 Tujuan 4
1.4 Batasan Masalah 4
1.5 Kontribusi 5
BAB 2 KAJIAN PUSTAKA 7
2.1 Kajian Penelitian Terkait 7
2.1.1 Metode Pemanfaatan Fitur Objek 7
2.1.2 Metode Penjejakan Objek 8
2.1.3 Manipulator ROV 9
2.1.4 Desain ROV 12
2.2 Teori Dasar 14
2.2.1 ROV 14
2.2.2 Jenis ROV 15
2.2.3 Sensor Visual 16
2.2.4 Proyeksi Kamera 16
2.2.5 Pengolahan Citra 17
2.2.6 Kinematika Robot 21
2.2.7 *Inverse Kinematics* 22

2.2.8 PID (*Propotional Integral Derivative*) 23
2.2.9 Sensor MS5803 25
2.2.10 Sensor *Inertial Measurement Unit* (IMU) 25
2.2.11 Sensor Sharp GP2Y0A21 26
2.2.12 Raspberry Pi 3 26
2.2.13 Arduino Mega 2560 27
2.2.14 Raspberry Pi kamera modul 27
2.2.15 Motor *Bilge Pump* 28
BAB 3 METODOLOGI PENELITIAN 29
3.1 Pengolahan Gambar 31
3.2 Pemilihan Objek 33
3.3 Kontrol ROV dan Manipulator 37
3.4 Perancangan Manipulator Arm 42
3.4.1 Kebutuhan Manipulator 42
3.4.2 Spesifikasi Manipulator 43
3.4.3 Manipulator Struktur 44
3.4.4 Peralatan *End effector* 44
3.4.5 Bahan Manipulator 45
3.5 Perancangan Sistem dan Konstruksi ROV 45
3.5.1 Rangka Utama 46
3.5.2 Tabung Utama 46
BAB 4 HASIL DAN PEMBAHASAN 49
4.1 Keseluruhan Sistem ROV 49
4.2 Pengujian Pendeteksian Objek 50
4.2.1 Pengujian Pendeteksian Kontur Objek 51
4.2.2 Pengujian Akurasi Pendeteksian Objek Dengan Variabel Kecepatan 52
4.2.3 Pengujian Akurasi Pendeteksian Objek Dengan Variabel Jarak 55
4.2.4 Pengujian Akurasi Posisi Penjejakan Objek 56
4.3 Pengujian Pengukuran Jarak 58
4.4 Pengujian Respon ROV Saat Pendektesian Objek 60
4.4.1 Pengujian Pada Aquarium Percobaan di Laboratorium 60
4.4.2 Pengujian Pada Kolam Renang 61

4.5 Evaluasi Kebutuhan Manipulator 63
4.6 Pengujian Waktu Mencapai Titik Koordinat 64
4.7 Pengujian Akurasi Sudut Servo 65
4.8 Pengujian Kemampuan Manipulator Terhadap Beban 66
4.9 Pengujian Kestabilan *Pitch* Terhadap Beban 67
4.10 Pengujian Kestabilan *Depth* Terhadap Beban 68
4.11 Pengaruh Efek Gerakan Manipulator Pada Kestabilan *Pitch* 69
4.12 Pengaruh Efek Gerakan Manipulator Pada Kestabilan *Depth* 70
4.13 Pengujian Parameter Kontrol PID 71
4.14 Pengujian Tingkat Keberhasilan Sistem Secara Keseluruhan 72
BAB 5 KESIMPULAN 75
5.1 Kesimpulan 75
DAFTAR PUSTAKA 77
LAMPIRAN 81
RIWAYAT HIDUP PENULIS 115

Halaman ini sengaja dikosongkan

DAFTAR GAMBAR

Gambar 1.1 Penggunaan ROV dalam perbaikan pipa 1
Gambar 1.2 Penggunaan ROV dalam membersihkan pipa..... 2
Gambar 2.1 Manipulator hidrolik 6 DoF 11
Gambar 2.2 Manipulator elektrik dengan kamera..... 11
Gambar 2.3 Mini ROV dengan manipulator..... 12
Gambar 2.4 Desain ROV *medium wrok class*..... 13
Gambar 2.5 Desain *micro* ROV..... 13
Gambar 2.6 Desain mini ROV untuk survei dan inspeksi 13
Gambar 2.7 *Remotely Operated vehicle* (ROV) [15]..... 14
Gambar 2.8 Model proyeksi kamera..... 17
Gambar 2.9 a). Gambar sebelum difilter b). Gambar hasil filter 18
Gambar 2.10 Gambar Threshold..... 21
Gambar 2.11 *Inverse kinematics*..... 22
Gambar 2.12 Diagram blok kontrol PID..... 25
Gambar 2.13 Sensor tekanan MS5803..... 26
Gambar 2.14 Sensor IMU MPU6050 26
Gambar 2.15 Sensor Sharp GP2Y0A21..... 27
Gambar 2.16 Raspberry Pi 3 27
Gambar 2.17 Arduino Mega 2560 28
Gambar 2.18 Raspberry Pi modul kamera..... 28
Gambar 2.19 Motor bilge pump..... 28
Gambar 3.1 Keseluruhan sistem ROV..... 29
Gambar 3.2 Diagram blok sistem secara keseluruhan 30
Gambar 3.3 Diagram blok pengolahan gambar 31
Gambar 3.4 Diagram blok pemilihan objek..... 33
Gambar 3.5 Diagram blok kontrol ROV dan manipulator..... 37
Gambar 3.6 Estimasi posisi objek berdasarkan sudut pandang kamera ROV 38
Gambar 3.7 Ilustrasi estimasi posisi objek..... 39
Gambar 3.8 Estimasi jarak objek 39
Gambar 3.9 Perencanaan posisi awal manipulator pada ROV 44
Gambar 3.10 Desain gripper [28]..... 45
Gambar 3.11 Desain dan ukuran manipulator..... 45
Gambar 3.12 Ukuran rangka utama ROV..... 46
Gambar 3.13 Pemasangan tabung pada rangka utama..... 46
Gambar 3.14 Diagram blok sistem ROV 47
Gambar 4.1 Keseluruhan Sistem ROV 49
Gambar 4.2 ROV Tampak Atas..... 50
Gambar 4.3 Manipulator arm ROV 50
Gambar 4.4 (a) Gambar RGB; (b) Gambar setelah gaussian blur; (c) Gambar setelah normalisasi RGB; (d) Gambar HSV; (e) Gambar objek yang telah ditandai. 52

Gambar 4.5 Gambar hasil Pendeteksian objek 53
Gambar 4.6 Perbandingan tingkat akurasi 54
Gambar 4.7 Jarak pengukuran maksimum 55
Gambar 4.8 Jarak optimal pendeteksian objek 56
Gambar 4.9 Akurasi posisi penjejakan objek 57
Gambar 4.10 *Error* posisi pada sumbu x dan sumbu y 57
Gambar 4.11 Grafik pengukuran jarak di permukaan 59
Gambar 4.12 Grafik pengukuran jarak di dalam air 59
Gambar 4.13 Respon sistem pada sumbu x pengujian di aquarium 60
Gambar 4.14 Respon sistem pada sumbu y pengujian di aquarium 61
Gambar 4.15 Respon sistem pada sumbu x pengujian kolam renang 62
Gambar 4.16 Respon sistem pada sumbu y pengujian kolam renang 62
Gambar 4.17 Pengujian beban manipulator 67
Gambar 4.18 Pengujian kesbilan *pitch* terhadap beban 68
Gambar 4.19 Kestabilan *depth* terhadap beban 69
Gambar 4.20 Efek gerakan manipulator pada kestabilan *pitch* 70
Gambar 4.21 Efek gerakan manipulator pada kestabilan *depth* 71

DAFTAR TABEL

Tabel 2.1 Kategori ROV [16] 15
Tabel 3.1 D-H Parameter 40
Tabel 3.2 Tabel kebutuhan manipulator..... 43
Tabel 3.3 Spesifikasi manipulator..... 43
Tabel 3.4 Sudut jangkauan motor servo..... 44
Tabel 4.1 Hasil pengukuran tingkat akurasi..... 54
Tabel 4.2 Hasil pembacaan sensor jarak..... 58
Tabel 4.3 Evaluasi kebutuhan manipulator..... 63
Tabel 4.4 Pengujian waktu mencapai titik koodinat 64
Tabel 4.5 Ketepatan sudut servo 1 65
Tabel 4.6 Ketepatan sudut servo 2 66
Tabel 4.7 Respon konstanta PID pada sumbu x..... 71
Tabel 4.8 Respon konstanta PID pada sumbu y..... 72
Tabel 4.9 Pengujian tingkat keberhasilan 73

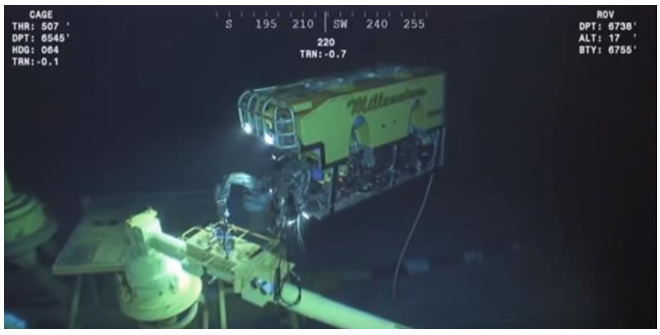
Halaman ini sengaja dikosongkan

BAB 1
PENDAHULUAN

1.1 Latar Belakang

Penggunaan pipa dalam laut menjadi hal yang penting dalam transportasi minyak, gas dan air bersih antar wilayah yang berbeda [1]. Instalasi dan proteksi pipa dalam air wajib dilakukan di negara yang bergantung pada komoditi tersebut untuk kesehatan dan tingkat ekonomi negara tersebut. Pipa dalam air yang dipergunakan untuk mengirimkan minyak atau gas dari *offshore* ke *onshore* misalnya membutuhkan perawatan dan diperlukan inspeksi berkala untuk mengecek keretakan, kebocoran atau karat untuk menjaga proses distribusi tetap berjalan dengan baik.

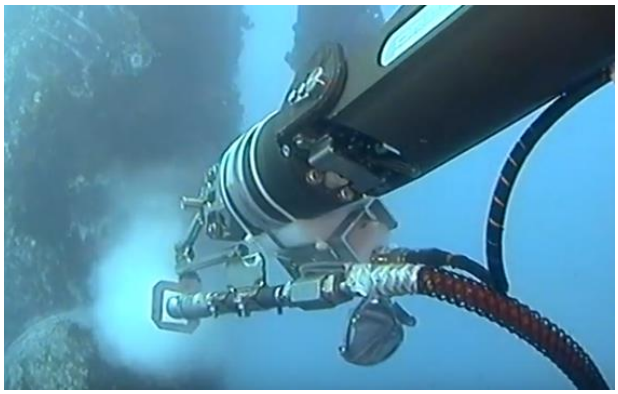
Saat pertama kali pipa dalam laut digunakan, para peneliti telah banyak mengembangkan berbagai macam metode dalam melakukan inspeksi, perbaikan dan perawatan. Pada umumnya pengembangan yang dilakukan berfokus pada area yang tidak dapat dijangkau oleh manusia [2]. Cara konvensional dalam perawatan dan penggantian pipa yang rusak masih menggunakan manusia yang menyelam di dasar laut untuk memindahkan berbagai peralatan yang digunakan dalam proses perbaikan pipa. Penggunaan penyelam untuk membawa maupun memindahkan peralatan dan bahan didasar laut sangatlah berbahaya dan sangat beresiko terhadap penyelam sebab kemampuan manusia didalam air sangat terbatas. Resiko penyelam dalam melakukan tugas penyelaman tidak lepas dari faktor lingkungan serta kelalaian yang menyebabkan terjadinya kecelakaan kerja.



Gambar 1.1 Penggunaan ROV dalam perbaikan pipa

Remotely Operated Vehicles (ROV) telah banyak digunakan dalam mengeksplorasi bawah laut termasuk untuk mengambil sampel bawah laut dan lainnya seperti yang terlihat pada Gambar 1.1. Penggunaan ROV untuk keperluan pengambilan sampel bawah laut harus dilengkapi dengan perangkat tambahan seperti manipulator yang dapat mengambil objek didasar laut [2]. Kemampuan seorang operator dalam mengendalikan ROV sangat penting sebab untuk mengontrol dan mempertahankan posisi ROV membutuhkan suatu keahlian ditambah seorang operator harus mengendalikan manipulator untuk mengembil suatu objek. Untuk mengontrol manipulator bukanlah hal yang mudah walaupun dengan operator yang berpengalaman dikarenakan untuk menyelesaikan suatu tugas tertentu operator harus mengontrol semua *joint* dan harus mempertimbangkan batas dari setiap *joint* serta relatif posisi dan orientasi dari *end-effector* [3].

Pada umumnya untuk mengendalikan sebuah ROV dibutuhkan operator lebih dari satu orang. Tugas operator pertama adalah untuk mengontrol ROV agar tetap stabil dan operator kedua bertugas untuk mengendalikan manipulator. Tugas utama dari seorang operator ROV adalah mempertahankan posisi stabil ROV untuk menghindari gangguan pergerakan manipulator [4]. Manipulator pada ROV banyak dikembangkan dengan berbagai sistem seperti yang terlihat pada Gambar 1.2. Pada umumnya sebuah manipulator menggunakan *gripper* sebagai *end effector*. Penggunaan manipulator dengan area kerja yang lebih spesifik dapat ditambahkan dengan berbagai peralatan diantaranya seperti penjepit, bor, dan lainnya untuk menunjang efektifitas dari pekerjaan yang dilakukan.



Gambar 1.2 Penggunaan ROV dalam membersihkan pipa

Dalam membangun sebuah ROV untuk melakukan *tracking* objek tertentu dibutuhkan suatu sistem yang dapat mendeteksi objek yang berada di sekitar ROV. ROV dapat mendekati objek secara *autonomous* dan kemudian mengambil objek dengan menggunakan manipulator. Banyak metode telah digunakan untuk sistem penjejakan ROV diantaranya menggunakan kamera, gelombang dan sensor laser [5]. Teknik menggunakan laser membutuhkan daya yang besar dan tempat yang luas sehingga sangat tidak efektif, menggunakan gelombang akustik juga sulit diterapkan karena metode ini sulit menghindari masalah multipel efek gelombang. Metode menggunakan kamera mengkonsumsi daya yang kecil dan dapat mengambil infomasi lingkungan yang lebih luas seperti warna, textur, geometri dll [6]. Walaupun memiliki berbagai kelebihan, metode penjejakan dengan menggunakan kamera memiliki kekurangan yaitu masalah intensitas cahaya dan kualitas air yang keruh mempengaruhi kualitas gambar.

Penelitian ini bertujuan untuk membangun sistem otomatis ROV dalam melakukan penjejakan dan mengambil objek dalam air. Konsep ini merupakan konsep awal dalam pengembangan sebuah ROV sebagai robot yang dapat membantu dan menggantikan fungsi manusia dalam melakukan pekerjaan dalam laut. Metode yang menjadi fokus penelitian ini adalah metode yang memanfaatkan fitur warna. Setelah metode berhasil mendeteksi posisi relatif objek, kemudian sistem mengirimkan data posisi untuk menentukan arah agar ROV dapat sejejar dengan objek. Setelah ROV telah sejajar terhadap objek, informasi jarak yang dikirim oleh sensor akan menggarakan ROV mendekati objek sesuai dengan jarak yang telah ditentukan kemudian menggerakan manipulator untuk mengambil objek.

1.2 Rumusan Masalah

Secara umum perumusan masalah dalam penelitian ini adalah merancang suatu sistem otomatis ROV dalam melakukan *tracking* dan mengambil objek dengan memanfaatkan kamera sebagai *visual sensor*. Adapun permasalahan secara umum dari penelitian ini adalah sebagai berikut :

1. Bagaimana membangun suatu sistem yang dapat membantu dalam mengeksplorasi dan mengambil objek dalam air?

2. Bagaimana membangun sistem yang dapat membantu operator ROV dalam navigasi mendekati objek?
3. Bagaimana membangun sistem yang dapat membantu operator ROV dalam mengambil objek secara otomatis?

1.3 Tujuan

Secara umum tujuan utama dari penelitian ini adalah merealisasikan sistem otomatis ROV dalam melakukan *tracking* dan mengambil objek dengan memanfaatkan kamera sebagai *visual sensor*. Adapun tujuan dari penelitian ini secara umum adalah sebagai berikut :

1. Merealisasikan sebuah ROV untuk membantu dalam mengeksplorasi dan mengambil objek dalam air.
2. Memanfaatkan kamera pada ROV sebagai sensor visual untuk menggerakan ROV secara *autonomous* mendekati objek.
3. Penggunaan manipulator ROV untuk mengambil objek.

1.4 Batasan Masalah

Adapun batasan-batasan masalah pada penelitian ini adalah sebagai berikut :

1. Objek yang dideteksi dan yang akan diambil tidak lebih besar dari ukuran *gripper.*
2. Objek yang dideteksi memiliki warna yang cerah dan berbeda dari objek lain disekitarnya.
3. Penelitian dilakukan di lingkungan yang terukur dengan pencahayaan, arus air dan kejernihan air yang stabil.
4. Objek yang diambil dengan posisi vertikal
5. ROV hanya diuji coba pada kolam dengan kedalman maksimal 2 m.

1.5 Kontribusi

Hasil dari penelitian ini diharapkan dapat memberikan manfaat antara lain yaitu :

1. Mempermudah operator ROV dalam menjalankan tugasnya untuk mengeksplorasi objek dalam air.
2. Membantu para penyelam untuk melakukan tugas dalam air.
3. Memberikan kontribusi ilmu pengetahuan dalam bidang elektronika khusunya pada bidang ROV.

Halaman ini sengaja dikosongkan

BAB 2
KAJIAN PUSTAKA

Pada bab ini akan dijelaskan teori dasar sebagai penunjang sistem yang akan dibangun serta metode yang dikembangkan pada sistem secara keseluruhan. Berikut adalah teori yang mendukung pada penelitian yang berhubungan dengan ROV, pengolahan citra, dan manipulator.

2.1 Kajian Penelitian Terkait

Pemanfaatan berbagai metode dalam mendeteksi maupun penjejakan objek telah banyak diaplikasikan dalam berbagai peralatan. Beberapa metode memanfaatkan fitur dalam mengenali objek. Adapun beberapa penelitian terkait mengenai metode yang memanfaatkan fitur objek serta metode penjejakan akan dijelaskan sebagai berikut.

2.1.1 Metode Pemanfaatan Fitur Objek

Dalam penjejakan onjek perlu dilakukan proses representasi bentuk objek agar dapat terbaca oleh komputer. Sifat dan karakteristik objek pada umumnya digunakan sebagai fitur penanda dari suatu objek. Hal yang penting dalam penjejakan objek adalah memilih fitur pada objek agar dapat dibedakan dengan objek lainnya. Dalam mempresentasikan suatu objek menggunakan seleksi fitur, dapat dibagi dalam beberapa fitur diantaranya *edges*, *optical flow*, warna, dan tekstur.

Pada seleksi fitur dengan menggunakan *edge* yaitu batas antar objek dan latar belakangnya pada umumnya lebih mudah untuk diaplikasikan. Garis tepi yang memiliki warna yang tegas dapat digunakan sebagai fitur sebuah objek yang memungkinkan algoritma ini melakukan penjejakan batas suatu objek. algoritma menggunakan fitur *edge* kurang sensitif dengan perubahan intensitas cahaya berbeda dengan menggunakan fitur warna [7].

Optical flow merupakan salah satu fitur yang beranggapan setiap objek bergerak mengacu pada gerakan pola yang berpindah dalam *frame* pertama dan

frame selanjutnya. Untuk menghitung dan mengidentifikasi gerakan setiap *pixel* antar bingkai digunakan berbagai metode salah satunya Lucas Kanade. Metode ini adalah metode *differensial* yang digunakan untuk mengestimasi pola gerak objek berdasarkan aliran optik.

Warna merupakan suatu fitur yang secara mudah dapat terdeteksi oleh mata dan dimiliki setiap objek. Pada umumnya sebuah warna diwakili oleh *Red, Green, Blue* (RGB) dalam pengolahan gambar namun, ruang warna *Hue Saturation Value* (HSV) lebih baik dalam mempresentasikan warna dibandingkan dengan RGB. Masalah utama dalam pengunaan warna sebagai fitur adalah sangat sensitif dengan perubahan intensitas cahaya [8].

Tekstur merupakan sifat-sifat keteraturan yang diukur dari variasi intensitas permukaan. Tekstur merupakan fitur yang dapat digunakan untuk menandai suatu objek namun perlu dilakukan pemrosesan untuk menghasilkan deskriptor. Deskriptor merupakan ukuran tekstur yang didalamnya terdapat filter untuk *level, edge, spot, wave,* dan *ripple*. Untuk dapat mengeliminasi interfensi kesamaan kontras yang menyebabkan sulit untuk mendapatkan fitur antara objek dan latar belakang, Koefisien rasio dari objek yang melingkupi lingkungan dari objek sangat penting untuk menghasilkan model objek [9].

2.1.2 Metode Penjejakan Objek

Untuk dapat melakukan penjejakan sebuah objek perlu dilakukan proses deteksi dan identifikasi dalam sebuah gambar. Pendeteksian objek memiliki mekanisme yang berbeda-beda, ada yang perlu memperhatikan metode penjejakan yang digunakan dan adapula tanpa perlu memperhatikan metode penjejakan yang digunakan namun dengan menggunakan pendekatan yang berbeda. Pendeteksian dilakukan pada setiap *frame* ketika objek pertama muncul dan dibandingkan dengan *frame* berikutnya. Ada beberapa metode yang populer dan sering digunakan seperti *point detector*, *background subtraction*, segmentasi , *supervised learning*, dan *temporal differencing*.

Detektor titik metode ini mengacu pada *point of interest* pada sebuah gambar. Titik yang dimaksud adalah bagian dari objek yang stabil walaupun terjadi perubahan intensitas cahaya dan sudut kamera. Pertemuan antara dua garis tepi

yang arahnya sama didefinisikan sebagai *corner*. Di sisi lain titik adalah dua garis tepi dalam *local neighbourhood* dari titik yang memiliki arah yang berbeda [10].

Metode *Background subtraction* merupakan pendekatan dasar yang membangun sebuah model dimana *background* atau latar belakang berfungsi sebagai referensi dan selalu diperbaruhi. Masing-masing *frame* akan dibandingkan dengan *frame* sebelumnya sehingga perubahan pada setiap *frame* dapat diketahui. Objek yang terdeteksi akan dibandingkan dengan posisi objek yang terdeteksi pada *frame* sebelumnya sehingga objek dapat ditandai [11].

Segmentasi merupakan sebuah metode membagi sebuah gambar dalam beberapa *segment* atau daerah. Pembagian sebuah gambar berdasarkan pada warna atau tekstur, hal ini membantu dalam menentukan posisi sebuah objek. Sebuah metode segmentasi dengan membagi gambar berdasarkan warna yang dipilih kemudian setiap piksel akan dibandingkan dengan nilai *threshold* sehingga menghasilkan gambar biner [8].

Supervised learning adalah metode deteksi objek dengan pembelajaran pola objek secara keseluruhan. metode ini menggunakan *some set of learning* untuk dapat menghasilkan sebuah fungsi yang dapat memetakan masukan sesuai dengan keluaran yang diinginkan. Dalam proses penjejakan objek pemilihan fitur dapat menjadi hal yang penting dalam proses klasifikasi. Adapun beberapa pendekatan pembelajaran yang ada seperti *neural network, adaptive boosting, decision trees*, dan *support vector machines.*

2.1.3 Manipulator ROV

Salah satu fitur yang harus ada pada *work class ROV* adalah manipulator. Manipulator adalah tangan mekanik pada ROV yang digunakan untuk melakukan suatu pekerjaan didasar laut. Laut dalam merupakan wilayah kerja yang sangat beresiko jika dikerjakan oleh manusia dan penggunaan ROV dengan manipulator merupakan salah satu pilihan untuk dapat melakukan suatu pekerjaan di laut dalam. Remot manipulator disebut dengan *teleoperation* karena alat ini dikendalikan dari jarak jauh. Dengan manipulator yang dapat dikendalikan dari jarak yang cukup jauh seorang operator ROV dapat memanipulasi objek di dasar laut dari permukaan.

Pada perkembangannya, manipulator pada ROV dibagi dalam dua jenis yaitu elektrik dan hidrolik. Pada elektrik manipulator terdiri dari motor servo pada setiap sendi untuk membuat pergerakan memutar atau menggunakan non elektrik linear aktuator seperti silinder. Salah satu contoh dari manipulator yang digunakan pada industri ROV adalah ARM 7E elektrik manipulator oleh ECA group [13] seperti yang terlihat pada Gambar 2.1. Manipulator ini memiliki maksimal jangkauan 1790 mm dengan kemampuan untuk mengangkat objek dengan kapasitas 40 Kg. Motor servo yang besar membutuhkan banyak ruang dibandingkan dengan penggunaan hidrolik dan hal ini akan menambah berat dari manipulator.

Hidrolik telah digunakan pada berbagai pekerjaan yang memiliki objek beban yang berat. Dengan tenaga yang besar dan berbanding dengan rasio berat, manipulator hidrolik lebih baik dibandingkan dengan manipulator elektrik. Dengan menggunakan manipulator hidrolik, semua komponen yang dibutuhkan dalam operasi seperti daya untuk hidrolik, *valve* dan *reservoir* dapat digabungkan pada satu tempat. Hal ini menjadi sangat simpel dam membuat desain sistem ROV menjadi lebih *compact.* Komponen pada hidrolik tidak membutuhkan daya besar dibandingkan dengan daya yang dibutuhkan pada manipulator elektrik [14].

Desain sistem pararel manipulator hidrolik yang dipasang pada sebuah *seabed crawlers* ROV telah diuji coba dan diimplementasikan. Komponen tambahan seperti *loader crane* sebagai aktuator digunakan pada desain sistem ini. Komponen mekanik yang telah dimodifikasi menjadikan komponen yang dijual secara umum sebagai komponen penyusun manipulator dengan 6 DoF untuk penggunaan dasar laut [15].

Penggunaan kamera pada manipulator untuk mendapatkan akurasi saat mengambil objek pada implementasinya hanya digunakan pada ROV kategori *heavy work class* [16]. Kamera yang diletakan pada manipulator untuk mendekatkan manipulator pada objek tidak dapat digunakan pada mini ROV. Hal ini disebabkan tidak semua kamera didesain untuk mendukung ROV dengan kategori *low-cost* dan *light weight* seperti mini ROV. Oleh karena itu, diperlukan sebuah desain manipulator ROV dengan mempertimbangkan komponen dan ukuran sistem. Pengolahan gambar untuk mengoptimalkan kemampuan ROV

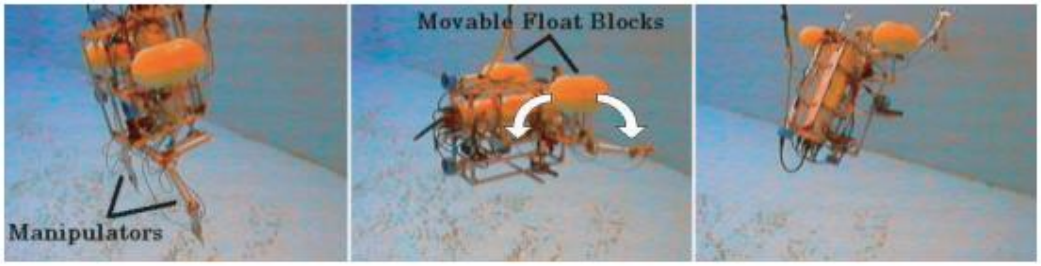
dalam menjelajah dalam air serta unit kamera yang memungkinkan untuk dipasang pada manipulator mini ROV seperti yang terlihat pada Gambar 2.2.

Pada perancangan manipulator dengan mengedepankan penghematan dalam penggunaan energi, desain manipulator ROV pada umumnya menggunakan hidrolik oli untuk instrumen kontrol manipulator namun dapat juga menggunakan teknologi *water hydraulics* [17].

Desain *dual* manipulator pada ROV yang digunakan untuk penelitian pada bidang eksplorasi biologi, geologi, dan arkeologi seperti yang terlihat pada Gambar 2.3. ROV. Manipulator memiliki dua karakteristik yang berbeda yang membedakan dengan manipulator pada umumnya yaitu sistem dual manipulator dan sistem kontrol. Ukuran manipulator yang dibuat seukuran dengan lengan manusia sehingga ROV dapat bekerja seperti pekerjaan yang biasa dilakukan oleh penyelam [18].

Gambar 2.1 Manipulator hidrolik 6 DoF

Gambar 2.2 Manipulator elektrik dengan kamera



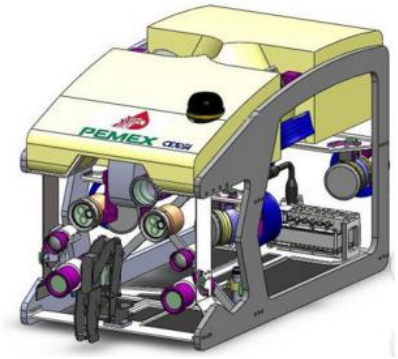
Gambar 2.3 Mini ROV dengan manipulator

2.1.4 Desain ROV

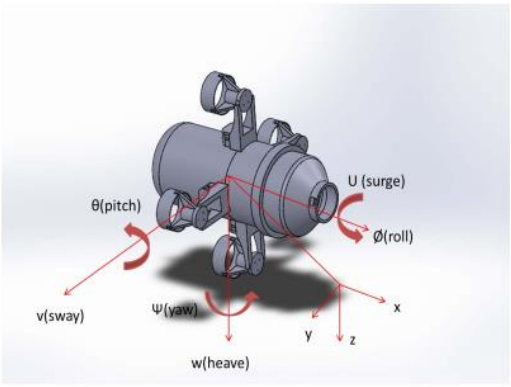
ROV kini telah berkembang dengan berbagai desain dan fitur tambahan yang berguna untuk memudahkan suatu pekerjaan. Desain ROV disesuaikan dengan kebutuhan dan kondisi lingkungan kerja ROV. Untuk sebuah ROV dengan jenis *micro* ROV yang didesain untuk melakukan inspeksi dan monitoring, desain yang digunakan harus lebih ringkas dan kecil sehingga mudah untuk dibawa dan dapat memasuki daerah-daerah yang sulit dijangkau. Adapun beberapa penelitian mengenai desain ROV adalah sebagai berikut.

Desain dan integrasi sistem ROV yang bertujuan untuk inspeksi visual di laut dalam harus dilengkapi dengan sebuah manipulator untuk dapat memanipulasi objek dalam laut seperti yang terlihat pada Gambar 2.4 [19]. Inspeksi visual yang diaplikasikan pada ROV digunakan untuk melihat pipa distribusi minyak dalam laut. Sistem pada ROV yang meliputi *Surface Control Unit* (SCU), *Launch and Recovery System* (LARS) dan *Tether Management System* (TMS). SCU merupakan sistem daya, kontrol dan sistem monitor yang digunakan oleh operator untuk mengontrol ROV. LARS merupakan perangkat seperti crane untuk melepas dan mengangkat ROV dari air. TMS merupakan penutup untuk melindungi ROV dan mengisolasi sistem dari efek gelombang dan efek arus.

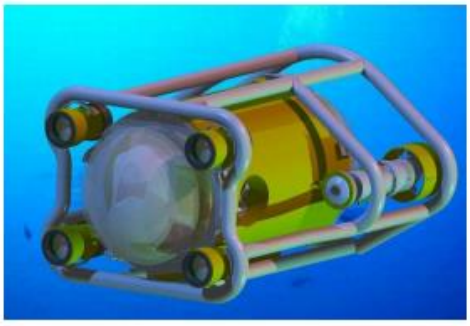
Desain *micro* ROV yang dikembangkan untuk tujuan obeservasi dilengkapi dengan kamera resolusi tinggi dan stuktur sistem mengedepankan kestabilan dalam gerakan *pitch*, *roll* dan *yaw* seperti yang terlihat pada Gambar 2.5 [20]. ROV yang dikembangkan dapat bergerak maju, mundur dan menyamping.



Gambar 2.4 Desain ROV *medium wrok class*



Gambar 2.5 Desain *micro* ROV



Gambar 2.6 Desain mini ROV untuk survei dan inspeksi

Jenis ROV dengan satu silinder dan dua penutup dirancang untuk melakukan inspeksi dan memberikan informasi visual bawah kapal seperti yang terlihat pada Gambar 2.6. Silinder digunakan untuk menyimpan semua komponen ROV dan dilindungi oleh bingkai eksternal. Bagian penutup ujung lainnya menggunakan bahan yang tembus cahaya untuk kamera. Sistem penggerak terdiri dari 4 motor. 2 motor digunakan untuk gerakan maju, mundur dan belok. 1 motor

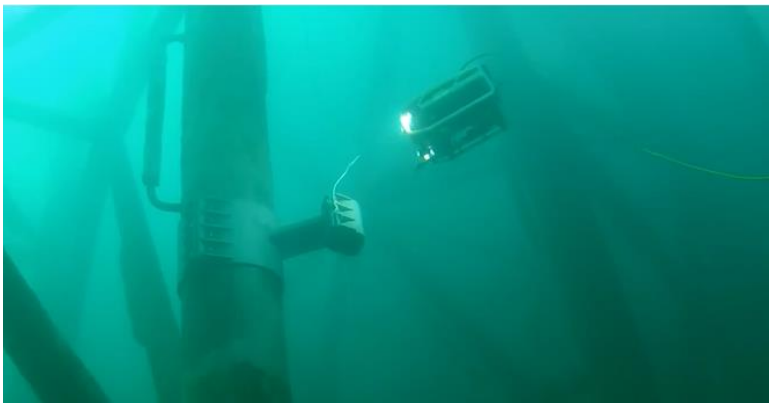
digunakan untuk gerakan naik dan turun. 1 motor digunakan untuk gerakan menyamping.

2.2 Teori Dasar

2.2.1 ROV

Remotely Operated Vehicle (ROV) adalah robot dalam air yang terhubung kabel antara robot dan stasiun dipermukaan air [12]. ROV telah banyak digunakan dalam mengeksplorasi bawah laut termasuk untuk mengambil sampel bawah laut, inspeksi pipa dalam laut, investigasi kecelakaan laut, dan sebagainya [13]. ROV memiliki kemampuan untuk mengamati benda dalam air dan dikendalikan dengan remot secara langsung dari permukaan air. Operator bertugas untuk mengendalikan ROV, mempertahankan posisi, dan mengoperasikan manipulator untuk berbagai tugas eksplorasi dalam air.

ROV telah digunakan diberbagai industri diantaranya industri minyak dan gas lepas pantai seperti yang terlihat pada Gambar 2.7. Sejak ROV digunakan diberbagai industri, desain dan kemampuan ROV lebih ditingkatkan. ROV saat ini dapat digunakan dan beroperasi di dalam laut dengan berbagai fitur tambahan seperti kamera resolusi tinggi, perangkat komunikasi yang lebih efektif, dan manipulator. Jarak antara ROV dengan operator yang berada dipermukaan air dibatasi oleh kemampuan penyaluran daya. Semakin jauh jarak ROV maka semakin besar pula resistansi yang dihasilkan oleh kabel. Hal itu menyebabkan terjadinya penurunan kecepatan ROV didalam air.



Gambar 2.7 *Remotely Operated vehicle* (ROV) [15]

2.2.2 Jenis ROV

Saat ini ROV telah dibagi dalam beberapa kategori berdasarkan ukuran, kemampuan menyelam, kekuatan motor, sensor-sensor, dan berbagai fitur seperti manipulator. Penggunaan ROV pun tidak terbatas pada inspeksi namun juga digunakan untuk keperluan yang lebih kompleks seperti perbaikan pipa dalam laut. Semakin besar sebuah ROV, maka semakin besar pula kapasitasnya. Sehingga fitur-fitur pendukung sangatlah penting guna menunjang ROV dalam melaksanakan tugas di dalam laut. Kategori ROV seperti yang terlihat pada Tabel 2.1 dibawah ini.

Tabel 2.1 Kategori ROV [16]

Kelas	Kedalaman (m)	Tipe	Tenaga (hp)
Micro observation	< 100	*Low Cost Small Electric ROV*	< 5
Mini Observation	< 300	*Mini (Small Electric ROV)*	< 10
Light/Medium work class	< 2000	*Medium (Electric ROV)*	< 100
Observation/Light work class	< 3000	*High Capacity Electric*	< 20
Heavy Work Class/Large Payload	< 3000	*High Capacity Electric*	< 300
Observation/Data collection	> 3000	*Ultra Deep*	< 25
Heavy Work Class/Large payload	> 3000	*Ultra Deep*	< 120

- *Micro observation* adalah jenis ROV dengan ukuran yang kecil dan dan berat yang relatif ringan dengan berat kurang dari 3 Kg. ROV ini digunakan untuk menggantikan penyelam ke area yang sempit dan sulit untuk dijangkau.
- *Mini observation* adalah jenis ROV dengan berat sekitar 15 Kg dan memiliki fungsi yang hampir sama dengan *micro observation*. Ukuran yang lebih besar membuat ROV ini mungkin membutuhkan sebuah boat untuk membawanya dan ROV ini lebih stabil saat terhempas gelombang.

- *General* adalah jenis ROV yang telah dilengkapi dengan manipulator. ROV ini berukuran cukup besar dengan penggunaan untuk kepentingan survei. Tipe pada umumnya didesain hanya untuk kedalaman kurang dari 1000 m.
- *Light Workclass* adalah jenis ROV yang memiliki daya kurang dari 50 hp dan dilengkapi dengan manipulator. Rangka ROV ini dibuat dari *polymers* seperti *polyethylene* berbeda dengan konvensional yang terbuat dari *stainless steel* atau aluminium. Jenis ROV ini dapat bekerja dikedalaman kurang dari 2000 m.
- *Heavy Workclass* adalah jenis ROV yang memiliki daya kurang dari 220 hp. ROV jenis ini memiliki kemampuan untuk mengangkat objek dan dilengkapi dengan 2 manipulator. ROV ini juga dapat bekerja hingga kedalaman 3000 m.

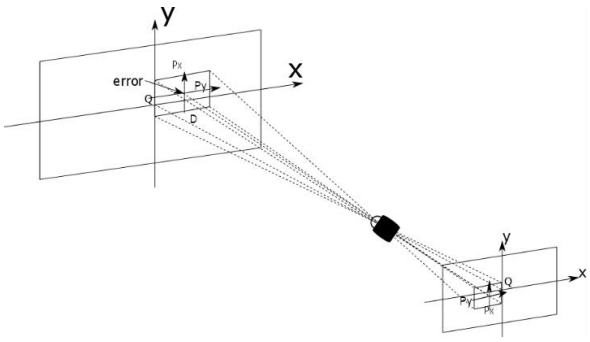
2.2.3 Sensor Visual

Sensor visual adalah perangkat yang digunakan untuk mengambil informasi permukaan benda secara optik berupa gambar. Prinsip kerja dari sensor visual adalah menangkap sebagian pemantulan cahaya dari objek yang mengarah pada kamera. Dari intensitas cahaya yang ditangkap mempresentasikan informasi objek kemudian informasi tersebut akan dibentuk menjadi citra data digital. Sensor ini berbentuk *chip* yang terletak dibelakang lensa. Semakin besar nilai *pixel* dari suatu sensor semakin detail gambar yang dihasilkan.

2.2.4 Proyeksi Kamera

Pada umumnya kamera terdiri dari sebuah lensa yang digunakan untuk memproyeksi objek 3 dimensi menjadi 2 dimensi. Karena citra yang dihasilkan dalam bentuk 2 dimensi maka informasi tentang kedalaman objek menjadi hilang. Untuk mengetahui perkiraan jarak antara objek dengan kamera perlu mengetahui model proyeksi kamera terhadap sistem kordinat kamera. Model proyeksi kamera yang ditunjukan pada Gambar 2.8. Posisi bidang 3 dimensi (x,y,z) dapat diketahui dengan perhitungan proyeksi pada sistem koordinat kamera pada bidang 2 dimensi (x,y) seperti Persamaan 2.1 dibawah ini.

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \frac{f}{z}\begin{bmatrix} x \\ y \end{bmatrix} \tag{2.1}$$



Gambar 2.8 Model proyeksi kamera

Pada model proyeksi kamera titik (x',y') merupakan titik kordinat yang tampil pada layar dengan satuan pixel. Z adalah jarak antara kamera dengan posisi objek yang diamati dan f adalah nilai *focal length* dari kamera. Untuk mencari nilai *focal length* dilakukan dengan cara menghitung nilai luasan pixel dengan perubahan jarak antara objek dan kamera.

2.2.5 Pengolahan Citra

Citra merupakan informasi visual yang diperoleh dari penangkapan cahaya yang dipantulkan oleh objek. Citra berbentuk 2 dimensi (x,y) dimana x dan y adalah sebuah kordinat spasial. Citra digital adalah representasi dari sebuah citra yang berbentuk 2 dimensi yang tersusun dari kumpulan nilai digital yang disebut elemen gambar atau *pixel*. *Pixel* sendiri adalah elemen terkecil penyusun citra yang mengandung suatu nilai. Nilai tersebut mewakili kecerahan dari suatu warna tertentu dan umumnya citra berbentuk persegi panjang. Ukuran citra dinyatakan dalam bentuk banyaknya *pixel* dan memiliki nilai kordinat. Setiap nilai digital mempresentasikan informasi yang diwakili citra tersebut.

Pengolahan citra adalah teknik manipulasi citra secara digital. Pengolahan citra membutuhkan sebuah komputer dalam pengelohannya. Tujuan dari pengolahan citra adalah untuk memperbaiki kualitas gambar atau dengan tujuan tertentu sesuai dengan keinginan pengguna. Citra digital berupa gambar yang

terususun atas *pixel* dalam baris dan kolom. Setiap *pixel* memiliki informasi tertentu yang terdiri dari informasi tingkat kecerahan warna primer.

2.2.5.1 Model warna *Red Green Blue* (RGB)

RGB adalah sutu model warna yang terdiri dari 3 warna dasar yang dapat menghasilkan berbagai macam warna. Kelebihan dari model warna RGB adalah gambar yang mudah disalin atau dipindahkan ke tempat lain tanpa harus lebih dahulu di diubah ke bentuk lain [26]. Model warna RGB merupakan model warna adaptif dengan tiga berkas cahaya yang ditambahkan bersama-sama dengan panjang gelombang untuk membuat spektrum warna akhir.

2.2.5.2 Filter *Gaussian Blur*

Filter *Gausian Blur* adalah filter yang digunakan mengurangi *noise* (gangguan) dengan efek *smoothing* atau *blur* [15]. Gambar hasil filter gaussian merupakan hasil dari konvolusi setiap *pixel* dalam matriks *gaussian*. Proses ini menghasilkan susunan nilai matriks yang baru. Fungsi filter gaussian blur ditunjukan pada Persamaan 2.2 berikut :

$$g(x,y) = G_\sigma(x,y) * f(x,y) \tag{2.2}$$

Dimana

$$G_\sigma = \frac{1}{\sqrt{2\pi\sigma^2}} exp\left(-\frac{x^2+y^2}{2\sigma^2}\right) \tag{2.3}$$

Gambar 2.9 a). Gambar sebelum difilter b). Gambar hasil filter

2.2.5.3 Normalisasi RGB

Normalisasi RGB adalah salah satu metode pengolahan citra yang digunakan untuk mengatasi perubahan intensitas cahaya. Metode ini meminimalisir terjadinya perubahan warna yang sangat drastis akibat perubahan intensitas cahaya [28]. Metode ini dilakukan dengan menghitung masing-masing *channel* atau kanal warna R, G, dan B pada suatu citra kemudian dilakukan penjumlahan yang menghasilkan nilai (*I*). Masing-masing kanal kemudian dibagi dengan nilai (*I*) dan dikalikan dengan 255. Hasil dari proses itu adalah *R'*, *G'*, dan *B'*. Dengan menggunakan metode ini sebuah objek yang memiliki warna tertentu dapat dideteksi baik dan pengaruh perubahan intensitas cahaya terhadap warna objek dapat diminimalisir. Adapun Persamaan dari metode ini dapat ditulis sebagai berikut :

$$f(x,y) = (R,G,B)$$

$$I = (R+G+B) \tag{2.4}$$

$$R' = \frac{R}{I}\, x\, 255 \tag{2.5}$$

$$G' = \frac{G}{I}\, x\, 255 \tag{2.6}$$

$$B' = \frac{B}{I}\, x\, 255 \tag{2.7}$$

$$g(x,y) = (R',G',B')$$

Normalisasi RGB dapat menghilangkan bagian warna yang terlalu cerah, bayangan, dan lebih memudahkan dalam pendeteksian objek. Kelemahan dari metode ini adalah tidak dapat membedakan warna hitam dan putih karena memiliki presentasi nilai RGB yang sama.

2.2.5.4 Model warna *Hue Saturation Value* (HSV)

HSV adalah representasi dari silinder kordinat dari nilai warna model RGB [29]. Model warna ini memiliki 3 karakteristik pokok yaitu *Hue, Saturation*, dan *Value*.

- *Hue* merupakan representasi pemilihan warna pada *color wheel*.
- *Saturasi* untuk menentukan presentase kepekatan warna.
- *Value* untuk menentukan tingkat kecerahan warna yang akan merubah tingkat kecerahan pada RGB.

$$V = \max(R, G, R) \tag{2.8}$$

$$S = \begin{cases} \dfrac{V - \min(R, G, B)}{V} & if\ V\ \neq 0 \\ 0 \qquad otherwise \end{cases} \tag{2.9}$$

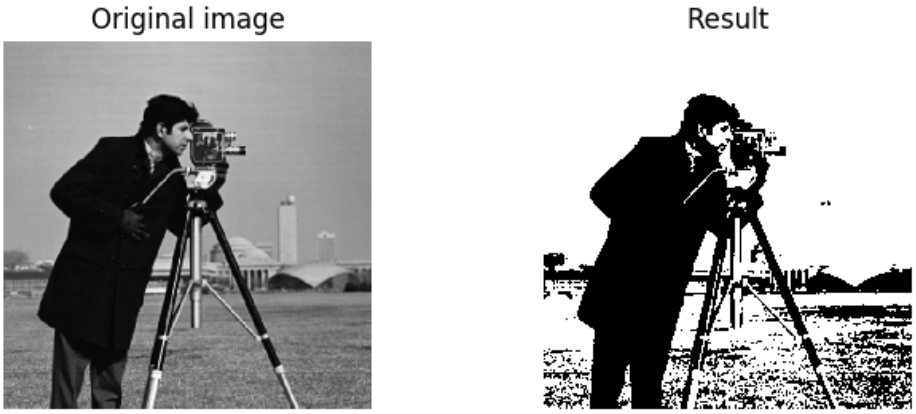
$$H = \begin{cases} \dfrac{60(G - B)}{(V - \min(R, G, B)} & if\ V = R \\ \dfrac{120 + 60(B - R)}{(V - \min(R, G, B)} & if\ V = G \\ \dfrac{240 + 60(R - G)}{(V - \min(R, G, B)} & if\ V = B \end{cases} \tag{2.10}$$

2.2.5.5 Metode *Threshold*

Threshold adalah metode untuk mengubah citra warna menjadi citra biner dengan cara mengelompokkan nilai setiap *pixel* ke dalam dua warna yaitu hitam dan putih [30]. Pada metode *threshold* nilai-nilai tiap *pixel* warna dipresentasikan dalam nilai 0 sebagai warna hitam dan 1 sebagai putih. Metode *threshold* ditunjukan seperti Persamaan 3.1 dibawah ini.

$$g(x, y) = \begin{cases} 1 & if\ \ f(x, y) > T \\ 0 & if\ \ f(x, y) \leq T \end{cases} \tag{2.11}$$

Nilai $g(x, y)$ merupakan hasil dari metode *threshold* untuk setiap *pixel* warna pada $f(x, y)$. T merupakan nilai ambang batas dengan kata lain jika nilai *pixel* suatu gambar berada di bawah nilai T maka *pixel* itu akan menjadi warna hitam, Sedangkan jika pixel suatu gambar berada di atas nilai T maka *pixel* akan menjadi warna putih. Hasil dari penggunaan metode *threshold* dapat terlihat seperti pada Gambar 2.11 dibawah ini.



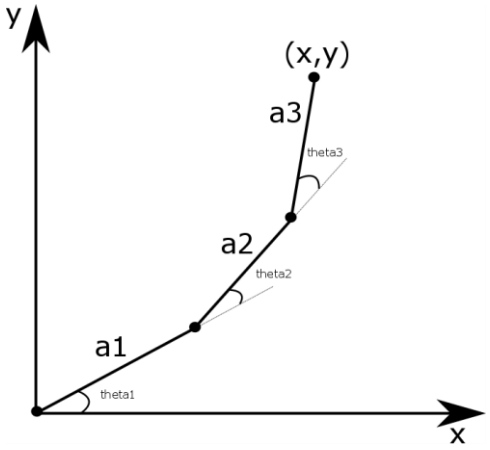
Gambar 2.10 Gambar Threshold

2.2.6 Kinematika Robot

Kinematika adalah cabang ilmu dari mekanika klasik mengenai gerak benda dan sistem benda tanpa mempersoalkan gaya yang menyebabkan benda tersebut bergerak. Ilmu ini mempelajari dan mengaplikasikan sistem kerangka koordinat geometri untuk menyusun struktur sistem sebuah robot. Geometri pada kinematika robot yaitu hubungan antar bagian robot yang dimodelkan sebagai benda tegar yang setiap sambungan antar benda tegar menghasilkan gerakan bebas secara translasi atau rotasi. Secara umum benda tegar yang dimaksud adalah bagian robot yang dapat bergerak dan tidak mengalami perubahan ukuran ketika mendapatkan gaya tertentu. Kinematika robot secara khusus mempelajari hubungan antara dimensi dan sambungan (*joint*) dari beberapa bagian (*link*). Dalam kinematika robot dibutuhkan posisi, kecepatan, dan percepatan untuk mengukur dan menghitung besarnya gaya yang berhubungan dengan torsi dari aktuator.

Robot terdiri dari *link* yang dihubungkan oleh *joint* (sendi) dan pada umumnya terdapat instrumentasi untuk mengukur posisi relatif *link*. Pada *joint* yang berputar, perpindahan posisi *link* disebut sudut *joint*. Setiap bagian *joint* akan menghasilkan tingkat kebebasan robot dalam bergerak. Jumlah yang menunjukan tingkat kebebasan disebut sebagai *Degree of Freedom* (DoF).

2.2.7 Inverse Kinematics



Gambar 2.11 *Inverse kinematics*

Inverse kinematics adalah fungsi yang menentukan nilai sudut dengan posisi *end effector* yang telah diketahui, *inverse kinematics* adalah kebalikan dari *forward kinematics*. *Inverse kinematics* menggunakan persamaan kinematik untuk menghitung nilai sudut berdasarkan posisi *end effector* yang telah diketahui seperti yang terlihat pada Gambar 2.12. Secara umum pada *inverse kinematics* nilai kordinat posisi *end effector* sebagai masukan dan nilai sudut sebagai keluaran. Posisi titik p dihitungndari titik (x,y) seperti persamaan berikut.

$$p_x = x - a_3 \, cos \, \emptyset \tag{2.12}$$

$$p_y = y - a_3 \, sin \, \emptyset \tag{2.13}$$

Untuk mendapatkan nilai θ_2 menggunakan aturan cos dengan persamaan awal adalah sebagai berikut:

$$p_x^2 + p_y^2 = a_1^2 + a_2^2 + 2a_1a_2 \cos\theta_2 \tag{2.14}$$

Nilai a merupakan panjang *link* pada lengan. Untuk mendapatkan nilai θ_2 perlu diketahui gerakan robot mengarah ke atas atau ke bawah. Untuk gerakan ke atas menggunakan pendekatan *elbow up* sedangkan untuk gerakan ke bawah menggunakan pendekatan *elbow down*. Persamaan elbow down seperti yang ditunjukan pada Persamaan 3.5 berikut.

$$\theta_2 = cos^{-1}\left(\frac{p_x^2 + p_y^2 - a_1^2 - a_2^2}{2a_1a_2}\right) \tag{2.15}$$

Untuk mendapatkan nilai θ_1 diperlukan hasil dari θ_2 dengan memasukan nilai a_1 yang merupakan panjang link 1 dan a_2 yang merupakan panjang link 2. Nilai θ_1 didapatkan dengan menggunakan persamaan sebagai berikut:

$$\theta_1 = tan^{-1}\left(\frac{p_y}{p_x}\right) - cos^{-1}\left(\frac{a_2\ sin\ \theta_2}{a_1 + a_2 cos\ \theta_2}\right) \quad (2.16)$$

Untuk mendapatkan θ_3 dengan cara mengurangi nilai $\emptyset$ yaitu orientasi dengan nilai sudut θ_1 dan θ_2 yang telah didapatkan sebelumnya seperti persamaan sebagai berikut:

$$\theta_3 = \emptyset - \theta_1 - \theta_2 \quad (2.17)$$

2.2.8 PID (*Propotional Integral Derivative*)

PID adalah salah satu metode kontrol yang terdiri atas kontrol *propotional*, *integral*, dan *derivatif*. PID juga merupakan kontrol klasik dan berhasil bertahan dari perkembangan teknologi dan tetap menjadi kontrol yang banyak digunakan [19]. PID merupakan kontroler untuk menentukan presisi suatu sistem instrumentasi dengan adanya umpan balik pada sistem seperti yang terlihat pada Gambar 2.13. PID memberikan aksi terhadap kontrol berdasarkan besarnya *error* yang didapatkan. *Set point* merupakan nilai perintah yang diberikan lalu dikurangi dengan output aktual sehingga menghasilkan *error*.

Perhitungan pada parameter PID, nilai *propotional* mempengaruhi reaksi pada *error* untuk proses yang sedang berlangsung, *integral* mempengaruhi nilai penjumlahan *error* terbaru sedangkan *derivatif* untuk menentukan reaksi kecepatan perubahan *error.*

Keluaran kontrol PID adalah penjumlahan dari nilai *propotional*, *integral* dan *derivative* dinyatakan dengan Persamaan 2.18 sebagai berikut.

$$\text{Output } PID = K_p\ [E + K_i\ \int E\ dt + K_d\ \frac{dE}{dt}] \quad (2.18)$$

Dimana

K_p = *Gain Propotional*

K_i = *Gain Integral*

K_d = *Gain Derivative*

Pengaturan nilai *gain* dapat mempengaruhi kestabilan keluaran PID. Apabila nilai dari parameter *gain* tidak sesuai maka sistem akan tidak stabil. Nilai K_p yang terlalu besar dapat menyebabkan *steady state error* yang besar. Nilai *gain* K_i akan mempercepat respon proses menuju *set point* namun karena K_i merespon akumulasi *error* dari nilai sebelumnya maka akan menyebabkan terjadinya osilasi pada output sistem. Nilai K_d akan memperlambat laju perubahan keluaran dan memberi efek mendekati nilai *set point*. K_d bertujuan untuk mengurangi *magnitude overshoot* yang dihasilkan oleh proses sebelumnya dan meningkatkan kestabilan proses kontrol.

Parameter-parameter penyusun kontrol PID dan pengaruh perubahan konstatnta pada sistem adalah sebagai berikut:

- *Proportional*

  Pada sistem, *proportional* memiliki pengaruh untuk mempercepat respon dan mengurangi *staedy state error*. Persamaan *proportional* seperti yang terlihat pada Persamaan 3.9 berikut:

  $$U(t) = K_p . E \tag{2.19}$$

- *Integral*

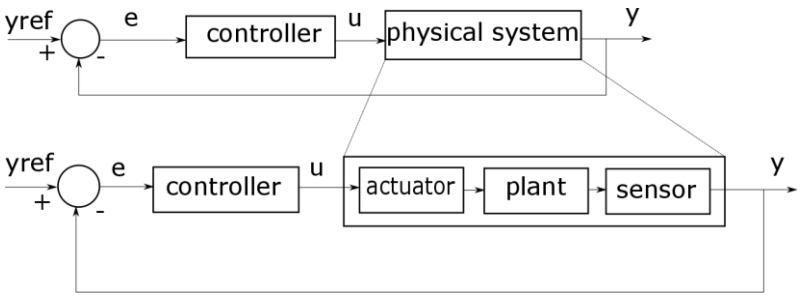
  Pada sistem, *integral* memiliki pengaruh menghilangkan *steady state error* namun dapat menimbulkan *osilasi* pada sistem. *Integral* juga memberikan efek rise time atau respon yang lambat. Persamaan *Integral* seperti yang terlihat pada Persamaan 3.10 berikut:

  $$U(t) = K_p \, K_i \int_o^t E \, dt \tag{2.20}$$

- *Derivative*

  Pada sistem, *derivative* memiliki pengaruh untuk mengurangi osilasi. Pada *derivative* jika *error* terjadi secara statis maka tidak akan berpengaruh pada sistem. *Derivative* akan bekerja saat terjadi perubahan *error*. Persamaan *derivative* seperti yang terlihat pada Persamaan 3.11 berikut:

  $$U(t) = K_p \, K_d \, \frac{dE}{dt} \tag{2.21}$$



Gambar 2.12 Diagram blok kontrol PID

2.2.9 Sensor MS5803

Sensor MS5803 merupakan sensor tekanan dengan resolusi tinggi. Komunikasi sensor ini menggunakan komunikasi *inter integrated circuit* (I2C). Sensor ini dapat mengoptimalkan kecepatan konversi dan dapat dihubungkan ke hampir semua mikrokontroler. Sensor ini memiliki protokol komunikasi sederhana tanpa perlu memprogram *internal register*. Pelindung gel dan tutup *stainless steel* memungkinkan penggunaan dengan kedalaman 100 m. Bentuk sensor ini seperti yang terlihat pada Gambar 2.14.

2.2.10 Sensor *Inertial Measurement Unit* (IMU)

Sensor IMU merupakan sensor yang digunakan untuk mengukur akselerasi, kemiringan, rotasi, getaran, dan goncangan. Penggunaan secara umum dari sensor ini adalah untuk menentukan orientasi dari sebuah benda dengan pendekatan sudut guling (*roll),* angguk (*pitch),* dan geleng *(yaw).* Bentuk dari sensor ini dapat terlihat seperti pada Gambar 2.14.

- ***Accelerometer***

Accelerometer adalah sensor yang digunakan untuk mengukur percepatan suatu objek. Percepatan yang diukur dapat berupa percepatan dinamis maupun statis. Pengukuran percepatan dinamis yaitu pengukuran pada benda yang bergerak sedangkan pengukuran statis yaitu pengukuran benda terhadap gravitasi bumi.

- ***Gyroscope***

Gyroscope adalah sensor untuk mengukur atau mepertahankan orientasi. Sensor ini menggunakan gaya gravitasi bumi untuk menentukan arah orientasi. Sensor ini secara mekanis berbentuk seperti piringan yang memiliki poros bebas untuk semua orientasi.

Gambar 2.13 Sensor tekanan MS5803

Gambar 2.14 Sensor IMU MPU6050

2.2.11 Sensor Sharp GP2Y0A21

Sharp GP2Y0A21 merupakan sensor yang digunakan untuk mengukur jarak dengan memanfaatkan pemantulan cahaya infra merah. Sensor ini terdiri dari kombinasi *Position Sensitive Distance* (PSD), *Infrared Emitting Diode* (IRED), dan sirkuit pemrosesan sinyal yang saling terintegrasi. Sensor ini tidak terpengaruhi suhu lingkungan dalam pendeteksian jarak karean mengadopsi metode triangulasi. Keluaran dari sensor ini adalah tegangan yang linier terhadap jarak objek yang terdeteksi. Sensor ini dapat digunakan sebagai sensor *proximity*. Bentuk sensor ini seperti yang terlihat pada Gambar 2.16.

2.2.12 Raspberry Pi 3

Raspberry Pi 3 adalah komputer berukuran kecil yang memiliki pin I/O atau sering disebut *General-purpose input/output* (GPIO). Mini komputer ini bekerja menggunakan sistem operasi linux dengan *system on chip* (SoC) broadcom BCM2837 dan CPU ARM cortex-A53 1.2 GHz. Mini komputer ini dibekali dengan *random acces memory* (RAM) sebesar 1 GB dan *graphics processing unit* (GPU) broadcom videocore IV. Komputer kecil ini membutuhkan tegangan kerja 5 V DC dan arus minimal 2 A. Peningkatan dari versi sebelumnya raspberry pi 3 telah

dilengkapi dengan *ethernet*, 2.4 GHz 802.11n wireless. Untuk penyimpanannya telah menggunakan microSD. Bentuk dari komputer kecil ini seperti yang terlihat pada Gambar 2.16.

2.2.13 Arduino Mega 2560

Arduino mega 2560 adalah pengembangan *board* mikrokontroler berbasis arduino yang ditanamkan chip ATmega2560. Mikrokontroler ini memiliki pin I/O sejumlah 54 pin. Mikrokontroler ini bekerja dengan *oscillator* sebesar16 Mhz dan tegangan operasi 5 V DC. *Board* arduino mega telah dilengkapi dengan *port* USB, *power jack* DC, ICSP *header* dan tombol reset. Bentuk dari mikrokontroler ini seperti yang terlihat pada Gambar 2.17.

2.2.14 Raspberry Pi kamera modul

Raspbery pi kamera modul merupakan kamera keluaran raspberry yang didesain khusus untuk produk raspberry. Kamera ini diproduksi dalam dua jenis yaitu kamera biasa dan kamera tanpa infra merah. Kamera ini mampu memberikan gambar yang jernih dengan ukuran 5 MP. Kamera ini juga dapat merekam video dengan kualitas 1080p dengan kecepatan 30fps dan 720p dengan kecepatan 60 fps. Sensor gambar yang digunakan pada kamera ini adalah omnivision 5647. Antarmuka komunikasi yang digunakan adalah serial interface CSI 15 pin. Bentuk dari kamera ini seperti yang terlihat pada Gambar 2.18.

Gambar 2.15 Sensor Sharp GP2Y0A21

Gambar 2.16 Raspberry Pi 3

2.2.15 Motor *Bilge Pump*

Motor *bilge pump* merupakan pompa air yang digunakan untuk menyedot air dilambung kapal. Motor ini dilengkapi dengan penutup yang kedap air sehingga motor dapat digunakan dalam air. Motor bilge pump bekerja dengan supply 12 V DC dengan arus maksimal 3 A dengan tenaga sebesar 1100 GPH. Bentuk dari motor ini seperti yang terlihat pada Gambar 2.19.

Gambar 2.17 Arduino Mega 2560

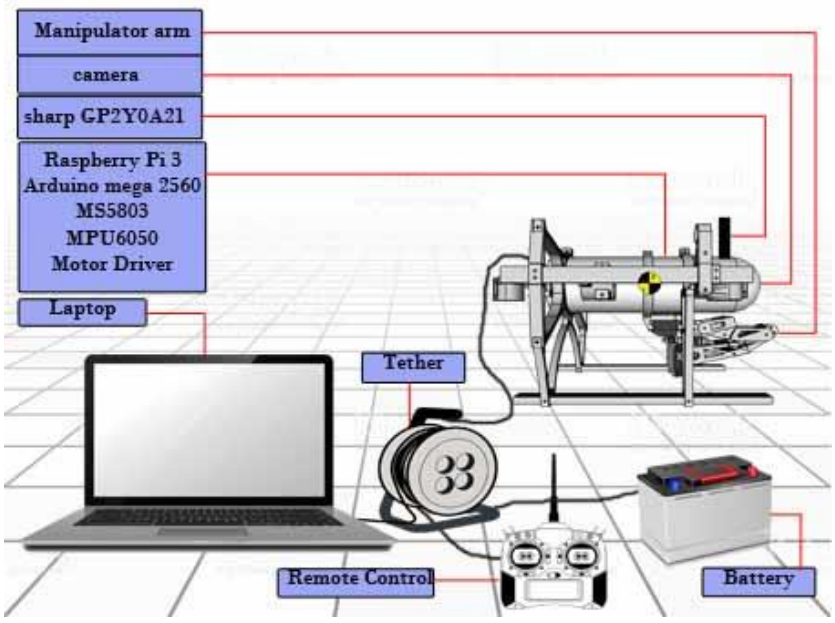
Gambar 2.18 Raspberry Pi modul kamera

Gambar 2.19 Motor bilge pump

BAB 3
METODOLOGI PENELITIAN

Pada penelitian ini berfokus pada membangun sistem otomatis ROV untuk mengambil objek menggunakan menipulator dengan memanfaatkan kamera sebagai *visual sensor*. Pada manipulator ROV menggunakan *end-effector* berupa penjepit untuk menggenggam objek yang dipilih. Komponen keseluruhan sistem ini seperti yang terlihat pada Gambar 3.1.

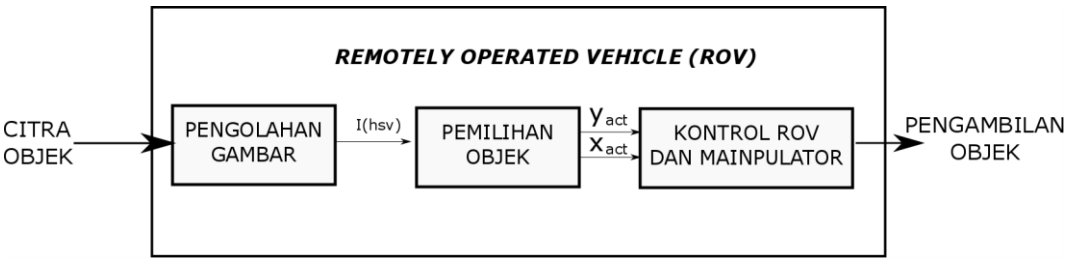
Sistem otomatis yang dibangun pada penelitian ini yaitu proses ROV mendekati objek dan menggerakan manipulator untuk mengambil objek. Gambar dalam air yang dikirim ROV ke stasiun kendali menjadi informasi visual oleh operator untuk melihat kondisi lingkungan kerja ROV. Operator dapat memberikan perintah pada ROV untuk mendekati dan mengambil objek. Perintah diberikan dengan cara operator memilih objek yang tampil pada layar komputer. Kemudian objek yang telah dipilih tersebut diproses oleh komputer ROV untuk menghasilkan koordinat relatif objek. Secara otomatis ROV akan bergerak mendekati objek tersebut. Saat posisi dan jarak aktual antara ROV dan objek telah sesuai, manipulator ROV akan mengambil objek tersebut.



Gambar 3.1 Keseluruhan sistem ROV

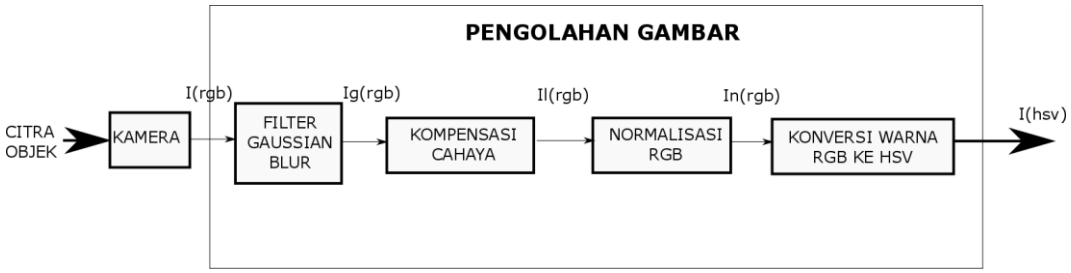
Pengontrolan posisi ROV menggunakan kontrol PID dan pengontrolan gerak manipulator menggunakan *inverse kinematics*. Manipulator membentuk posisi *standby* saat tidak ada objek yang terdeteksi. Saat objek terdeteksi dan berada tepat didepan ROV, maka manipulator akan membentuk posisi tegak lurus. Posisi manipulator saat mendeteksi dan tidak mendeteksi objek akan dijelaskan secara detail pada sub bab selanjutnya. Saat objek telah berada pada posisi dan jarak yang sesuai maka jepit akan menjepit objek.

Pengontrolan posisi ROV dibagi menjadi dua mode yaitu mode manual dan mode otomatis. Mode manual menggunakan remot kontrol yang dikendalikan secara manual oleh operator. Operator dapat mengendalikan pergerakan ROV dan manipulator sesuai arah yang dikehendaki. Untuk mempertahankan posisi stabil kedalaman menggunakan nilai yang dikirim oleh sensor tekanan. Pada mode otomatis ROV bergerak berdasarkan posisi objek yang telah dipilih. Untuk mempertahankan posisi, ROV menggunakan nilai koordinat relatif objek yang terdeteksi. Jarak aktual ROV dan objek berdasarkan pembacaan sensor jarak yang diletakan pada bagian depan tabung ROV. Saat proses mendekati objek, ROV bergerak dengan kecepatan konstan. Manipulator akan aktif pada saat posisi aktual ROV telah sejajar dengan objek dan jarak minimum yang telah ditentukan sudah terpenuhi.



Gambar 3.2 Diagram blok sistem secara keseluruhan

3.1 Pengolahan Gambar



Gambar 3.3 Diagram blok pengolahan gambar

Pada tahap ini dilakukan proses pengolahan gambar dengan resolusi 640x480 px. Pemrograman dilakukan dengan menggunakan IDE Qt bahasa pemrograman C++. Pemrograman dilakukan pada sistem operasi Linux dengan menggunakan OpenCV. Untuk dapat memisahkan objek dengan latar belakangnya maka perlu dilakukan beberapa proses agar dapat meningkatkan kualitas serta mengurangi *noise* pada gambar sehingga objek dapat terdeteksi dengan baik. Pencahayaan menjadi faktor yang penting dalam pemrosesan gambar. Untuk meminimalisir efek dari perubahan intensitas cahaya yang drastis dan perubahan sudut datang cahaya maka dilakukan beberapa proses seperti yang ditunjukan pada Gambar 3.3.

A. Filter *Gaussian Blur*

Kamera yang diletakan dibagian depan ROV digunakan untuk mengambil gambar secara *continue* saat ROV berada didalam air. Gambar 3 kanal dalam bentuk RGB yang disimbolkan dengan I(rgb) kemudian difilter untuk mengurangi *noise*. Gambar difilter menggunakan *gaussian blur* sehingga menghasilkan gambar yang lebih halus. Titik-titik hitam pada gambar akan menghilang. Filter *gaussian blur* juga memberikan efek warna yang kontras akan terlihat lebih halus. Filter *gaussian blur* menggunakan *kernel* 7x7 dan standar deviasi yang digunakan untuk sigma x dan sigma y adalah 3. Gambar hasil filter disimbolkan dengan $I_g(rgb)$.

B. Kompensasi Cahaya

Gambar $I_g(rgb)$ hasil filter selanjutnya diproses untuk mendapatkan warna yang lebih tahan terhadap perubahan cahaya. Proses ini disebut dengan kompensasi

cahaya. Tahap pertama, gambar $I_g(rgb)$ dipisah menjadi 3 kanal r, g dan b. Kemudian masing-masing kanal dirata-ratakan dengan cara membagi masing-masing kanal dengan jumlah ukuran kanal itu sendiri. Seperti yang ditunjukan pada persamaan dibawah ini :

$$I_l(r) = \frac{1}{N} \sum_{i,j} I_r(i,j) \tag{3.1}$$

$$I_l(g) = \frac{1}{N} \sum_{i,j} I_g(i,j) \tag{3.2}$$

$$I_l(b) = \frac{1}{N} \sum_{i,j} I_b(i,j) \tag{3.3}$$

C. Normalisasi RGB

Setelah dilakukan kompensasi cahaya pada gambar yang disimbolkan dengan $I_l(rgb)$, proses selanjutnya adalah normalisasi RGB. Normalisasi RGB dilakukan untuk membuat warna lebih merata dan halus. Perubahan intensitas cahaya yang mengenai permukaan objek membuat warna lebih terang atau lebih gelap dibeberapa sisi. Untuk dapat diminimalisir perubahan warna tersebut maka dilakukan normalisasi RGB dengan persamaan sebgai berikut :

$$I = I_{lr}(i,j) + I_{lg}(i,j) + I_{lb}(i,j) \tag{3.4}$$

$$I_n(r) = \frac{I_{lr}(i,j)}{I}\ 255 \tag{3.5}$$

$$I_n(g) = \frac{I_{lg}(i,j)}{I}\ 255 \tag{3.6}$$

$$I_n(b) = \frac{I_{lb}(i,j)}{I}\ 255 \tag{3.7}$$

D. Konversi Warna RGB ke HSV

Gambar yang telah dinormalisasi selanjutnya diubah kedalam bentuk HSV karena ruang warna HSV lebih baik dalam mempresentasikan warna. *Hue* merupakan sudut yang mempresentasikan warna, *Saturation* adalah untuk

mempresentasikan kepekatan warna dan *value* untuk gelap dan terangnya warna. Pada proses ini warna yang didefinisikan diwakili oleh *hue*, *saturation* dan *value* disimbolkan dengan $I(hsv)$. Pada proses konversi warna RGB ke HSV menggunakan persamaan seperti yang terlihat dibawah ini.

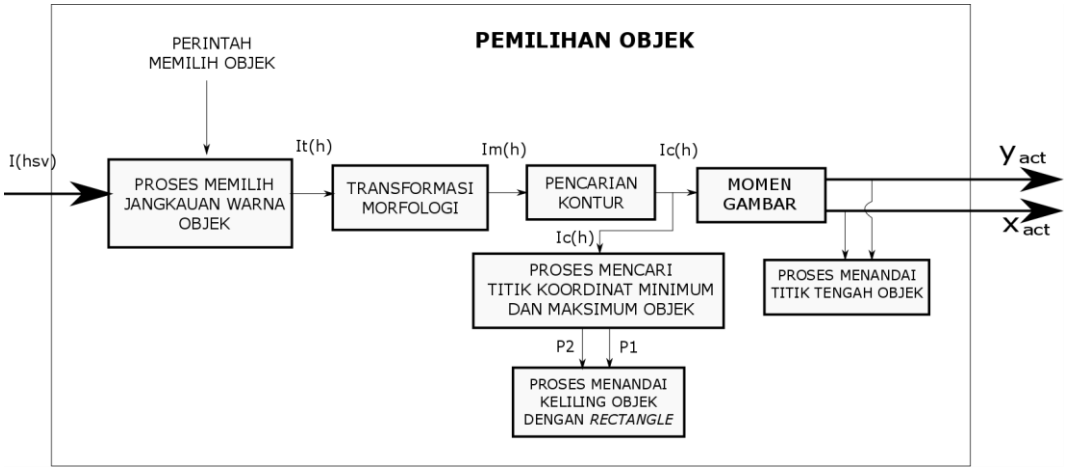
$$V = \max I_n(rgb) \tag{3.3}$$

$$S = \begin{cases} \dfrac{V - \min I_n(rgb)}{V} & if\ V\ \neq 0 \\ 0 \qquad otherwise \end{cases} \tag{3.4}$$

$$H = \begin{cases} \dfrac{60(I_n(g) - I_n(b))}{(V - \min I_n(rgb)} & if\ V = I_n(r) \\ \dfrac{120 + 60(I_n(b) - I_n(r))}{(V - \min I_n(rgb)} & if\ V = I_n(g) \\ \dfrac{240 + 60(I_n(r) - I_n(g))}{(V - \min I_n(rgb)} & if\ V = I_n(b) \end{cases} \tag{3.5}$$

3.2 Pemilihan Objek

Pemilihan objek adalah proses untuk memilih *region of interest* (ROI) pada gambar. Proses ini memisahkan objek yang menjadi ROI dengan objek lainnya. Pada proses pemilihan objek, terdapat beberapa proses yang meliputi pemilihan jangkauan warna, transformasi morfologi, pencarian kontur, dan momen gambar. Proses pemilihan objek secara keseluruhan dapat dilihat pada Gambar 3.4.



Gambar 3.4 Diagram blok pemilihan objek

A. Proses Memilih Jangkauan Warna Objek

Proses memilih jangkauan warna objek bertujuan untuk memisahkan objek dan latar belakang berdasarkan warna objek yang telah dipilih. Proses ini dilakukan dengan cara melakukan klik pada objek yang tampil pada layar. Kemudian warna yang dipilih akan disimpan pada sebuah *array* dan menjadi nilai referensi. Nilai warna yang tersimpan akan dicari nilai minimum dan maksimumnya. Nilai minimum dan maksimum digunakan sebagai jangkauan nilai warna objek. Nilai warna objek hasil seleksi warna disimpan sebagai nilai referensi seperti Persamaan 3.5 berikut:

$$I(i,j) = \begin{cases} \dfrac{60(I_n(g) - I_n(b))}{(V - \min In(rgb)} & if\ V = I_n(r) \\ \dfrac{120 + 60(I_n(b) - I_n(r))}{(V - \min In(rgb)} & if\ V = I_n(g) \\ \dfrac{240 + 60(I_n(r) - I_n(g))}{(V - \min In(rgb)} & if\ V = I_n(b) \end{cases} \quad (3.5)\ (3.6)$$

Dalam HSV tidak semua kanal digunakan sebagai referensi warna. Dalam proses ini hanya diwakili oleh *Hue* karena yang dibutuhkan dalam proses pemilihan objek adalah warna objek tersebut. Untuk menghilangkan *noise* pada daerah sekitar objek maka perlu dilakukan filter dengan mengatur nilai jangkauan warna minimum dan maksimum. Nilai Th merupakan nilai warna saat dilakukan klik warna pada objek. Dengan mengatur nilai jangkauan warna Th, warna objek yang terseleksi lebih sempit sehingga warna lain yang mendekati nilai warna objek tidak dapat terdeteksi. Hasil dari proses ini adalah gambar biner yang disimbolkan dengan $I_t(h)$. Proses *threshold* ini menghasilkan bentuk objek dan memisahkan objek dari *background* atau objek lain disekitarnya.

$$I_t(h) = \begin{cases} 1\,; Th - 10 < I(i,j) < Th + 10 \\ 0\,; Th - 10 \geq I(i,j) \geq Th + 10 \end{cases} \quad (3.7)$$

Hasil dari proses ini adalah gambar biner $I_t(h)$ yang telah berbentuk objek namun masih dipengaruhi *noise* dan bagian-bagian objek yang tidak terhubung. Untuk menghilangkan noise dan agar bentuk objek menyerupai bentuk aktual maka perlu dilakukan proses selanjutnya yaitu transformasi morphologi.

B. Transformasi Morfologi

Tujuan dari proses transformasi morfologi adalah untuk menghilangkan *noise*, mengisolasi elemen *pixel* dan menghubungkan bagian yang terputus. Jika *noise* pada gambar terlalu banyak maka akan menggangu proses pendeteksian objek. Untuk menghilangkan *noise* maka dilakukan transformasi morfologi dengan menggunakan erosi dan dilasi. Pada gambar biner dilasi memberikan efek objek putih menjadi lebih besar dan erosi memberikan efek objek putih akan menjadi lebih kecil. Untuk mendapatkan perkiraan ukuran objek maka harus diatur ukuran pada erosi dan dilasi. Pada proses ini menggunakan fungsi dengan *size* pada erosi adalah 1 dan *size* pada dilasi adalah 5. Hasil pada proses ini disimbolkan dengan $I_M(h)$..

C. Pencarian Kontur

Proses pencarian kontur dilakukan dengan tujuan untuk mendapatkan luas permukaan objek. Proses ini dilakukan setelah objek melewati beberapa proses sehingga bentuk kontur objek tidak terinterfensi dengan *noise*. Hasil pendeteksian kontur berupa titik piksel disekeliling objek yang membentuk seperti rantai yang disimbolkan dengan $I_c(h)$. Pada proses ini digunakan fungsi yang telah tersedia. Kemudian untuk menentukan luasan daerah objek yang terdeteksi maka perlu dilakukan perbandingan antara minimal luasan objek, maksimal luasan objek, dan luas daerah referensi. Hal ini dilakukan untuk menghindari banyaknya objek yang terdeteksi jika pada gambar banyak objek yang memiliki warna yang sama. Minimal luasan objek yang digunakan pada penelitian ini adalah 60x60 px. Hal ini mengakibatkan objek dengan ukuran dibawah nilai minimum tidak akan terdeteksi. Untuk nilai maksimum digunakan 320x280 px, sehingga apabila warna referensi menutup semua bagian kamera tidak akan terdeteksi sebagai objek.

D. Proses Mencari Titik Koordinat Minimum dan Maksimum Objek

Untuk menandai objek yang telah terseleksi dari objek lainnya adalah dengan mencari koordinat nilai titik paling awal objek dan nilai titik paling akhir objek menggunakan sebuah fungsi program *boundingRec()*. Pada proses ini digunakan luasan objek dengan sistem koordinat. Dengan kata lain, titik min(x),

min(y) dan max(x), max(y) pada objek yang telah terseleksi digunakan sebagai titik batas atas dan titik batas bawah pada fungsi *rectangle()*. Untuk titik batas atas yang telah didapatkan disimbolkan dengan P1 dan titik batas bawah disimbolkan dengan P2.

E. Proses Menandai Keliling Objek Dengan *Rectangle*

Proses selanjutnya adalah menandai objek dalam sebuah bingkai kotak (*rectangle*) berdasarkan nilai titik P1 dan titik P2 pada sumbu x dan sumbu y yang telah diketahui. Proses ini menggunakan fungsi *rectangle()* yang telah tersedia dengan tujuan untuk menandai objek yang menjadi ROI saat ditampilkan pada layar komputer.

F. Momen Gambar

Momen gambar adalah proses untuk mendapatkan titik tengah objek. Proses ini menggunakan kontur objek yang telah diperoleh dari proses sebelumnya. Proses momen gambar menghasilkan posisi objek berdasarkan koordinat sumbu x dan sumbu y. Dalam menentukan posisi objek, titik yang menjadi informasi posisi adalah titik tengah objek. Untuk mendapatkan nilai luasan dan titik tengah objek menggunakan fungsi *moments()* dengan persamaan sebagai berikut:

Luas area objek

$$M_{00} = \sum_{x}\sum_{y} I_t(x,y) \tag{3.8}$$

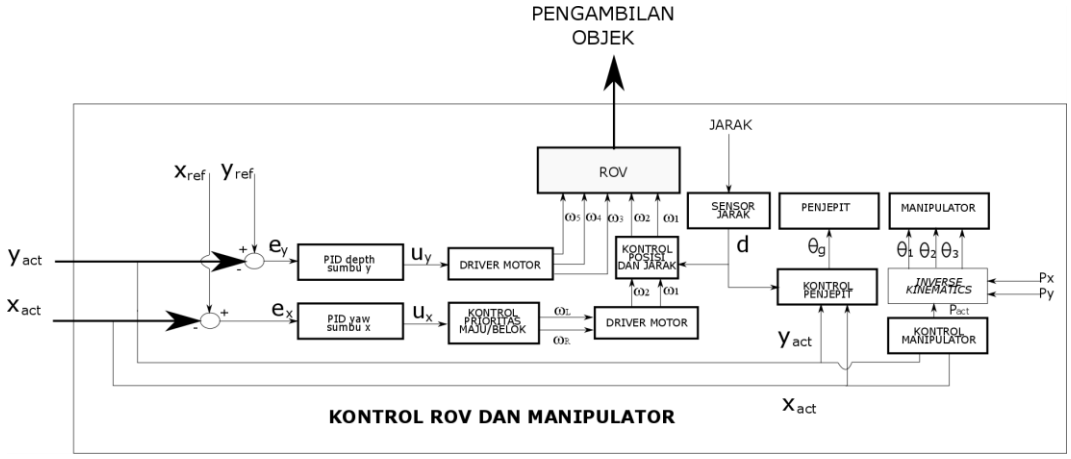
Luas area x dan y

$$M_{10} = \sum_{x}\sum_{y} x\, I_t(x,y)$$
$$M_{01} = \sum_{x}\sum_{y} y\, I_t(x,y) \tag{3.9}$$

Koordinat objek

$$x_{act} = \frac{M_{10}}{M_{00}}, y_{act} = \frac{M_{01}}{M_{00}} \tag{3.10}$$

3.3 Kontrol ROV dan Manipulator

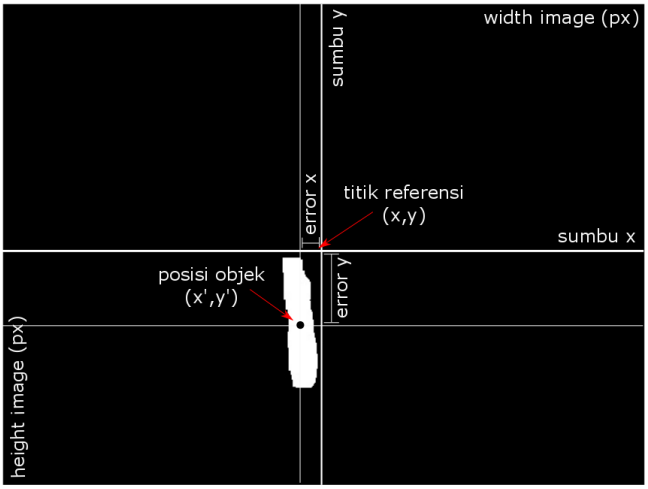


Gambar 3.5 Diagram blok kontrol ROV dan manipulator

Kontrol ROV dibagi menjadi dua yaitu manual dan otomatis. Untuk membuat pergerakan yang stabil digunkan kontrol PID dengan nilai *setpoint* untuk *roll* dan *pitch* adalah 0 derajat. Pada kontrol manual pergerakan ROV dikendalikan dengan menggunakan remot kontrol dan pada sistem otomatis menggunakan data koordinat yang dikirim oleh komputer ROV.

A. Kontrol PID *Depth* dan *Yaw*

Kontrol PID *depth* dan *yaw* merupakan kontrol untuk menggerakan ROV sesuai dengan posisi objek. Titik referensi x_{ref} dan y_{ref} yang digunakan pada sistem ini adalah (320,240). Titik ini merupakan titik tengah layar dengan resolusi 640x480 px. Titik referensi tersebut kemudian dibandingkan dengan titik aktual objek x_{act} dan y_{act}untuk menghasilkan besarnya e_x dan e_y seperti yang terlihat pada gambar 3.6. Nilai *error* yang didapatkan menjadi masukan pada kontrol PID *depth* (y) dan kontrol PID *yaw* (x). Kontrol PID menghasilkan keluaran U_x dan U_y untuk menggerakan motor ROV. U_y menghasilkan kecepatan sudut ω_3, ω_4 dan ω_5 untuk menggerakan 3 motor ROV. Ketiga motor tersebut mengatur kedalaman posisi ROV dengan kata lain ROV dapat bergerak ke atas atau ke bawah pada sumbu y. U_x menghasilkan kecepatan sudut ω_1 dan ω_2 untuk menggerakan 2 motor ROV. Kedua motor tersebut mengatur posisi ROV belok kanan, belok kiri, atau maju. Untuk menentukan posisi ROV harus maju atau belok dikontrol oleh proses kontrol prioritas. Kontrol ini akan dijelaskan lebih mendalam pada sub bab selanjutnya.



Gambar 3.6 Estimasi posisi objek berdasarkan sudut pandang kamera ROV

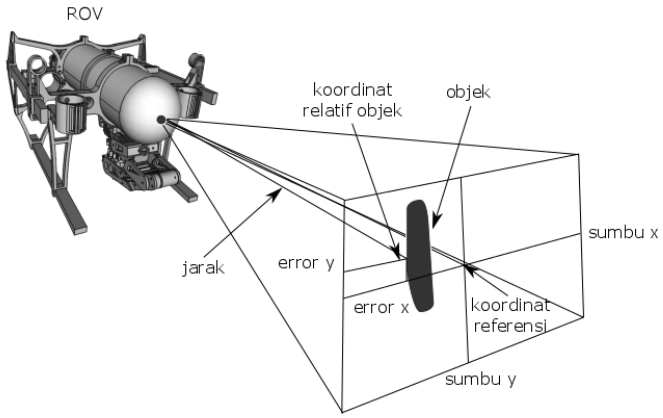
B. Kontrol Prioritas

Kontrol prioritas merupakan sebuah syarat yang digunakan untuk memberikan masukan pada *driver* motor ω_1 dan ω_2. Pada proses ROV mendekati objek, untuk menggerakan motor 1 dan motor 2 bergerak maju maka nilai e_x dan e_y harus lebih kecil dari 20 dan lebih besar dari -20. Hal ini bertujuan agar ROV tidak bergerak maju saat posisi x_{act} dan y_{act} belum mengarah pada objek. Setelah nilai e_x dan e_y sesuai dengan syarat yang telah ditentukan maka ROV akan bergerak maju. Saat posisi objek belum berada pada titik tengah sudut pandang ROV, motor ROV akan menyesuaikan posisi hingga sejajar pada objek.

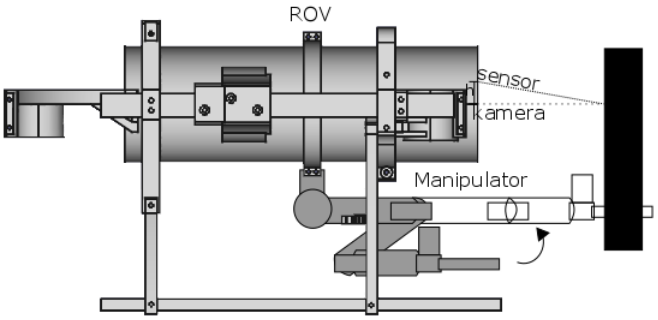
C. Kontrol Posisi dan Jarak

Kontrol posisi dan jarak merupakan proses pengontrolan posisi dan jarak ROV terhadap objek. Posisi adalah letak atau kedudukan yang mempresentasikan keberadaan suatu titik terhadap titik lainnya. Sedangkan jarak adalah angka yang menunjukan seberapa jauh objek berubah posisi pada suatu lintasan. Titik referensi x_{ref} dan y_{ref} harus sejajar dengan titik aktual x_{act} dan y_{act} agar ROV dapat mengestimasi jarak objek. Objek harus berada pada daerah yang teramati oleh kamera (*field of view*) sehingga luasan objek dapat teramati secara penuh. Pada umumnya titik dapat dijabarkan menggunakan jarak namun pada penelitian ini menggunakan satuan *pixel* dalam mempresentasikan posisi antara titik referensi dan titik koordinat aktual. Sensor yang digunakan adalah sensor sharp GP2Y0A21

dengan keluaran analog yang dikonversi dalam satuan cm. Sensor ini diletakan dibagian dalam tabung ROV dengan arah menghadap objek. Jarak antara objek dan ROV yang telah ditetapkan adalah 22 cm. Jarak tersebut berhubungan dengan jangkauan maksimal manipulator saat dalam posisi mengambil objek. ROV akan berhenti berdasarkan syarat yang ditetapkan apabila objek berada pada range 270-370 px pada sumbu x, 190-290 px pada sumbu y, dan jarak yang terukur kurang dari 22 cm.. Estimasi posisi dan jarak objek seperti yang dapat dilihat pada Gambar 3.7 dan 3.8.



Gambar 3.7 Ilustrasi estimasi posisi objek



Gambar 3.8 Estimasi jarak objek

D. Kontrol Penjepit

Kontrol Penjepit merupakan syarat yang digunakan untuk menggerakan servo penjepit dengan sudut tertentu. Sudut tertentu yang dimaksud adalah gerakan menjepit dan melepas. Syarat yang ditentukan untuk penjepit dapat menjepit adalah saat posisi objek berada diantara 270 px sampai 370 px pada sumbu x, 100 px sampai 370 px pada sumbu y, dan jarak yang terdeteksi pada sensor adalah 22 cm. Diluar syarat yang telah ditentukan posisi penjepit adalah melepas.

E. Kontrol Manipulator

Kontrol manipulator adalah kontrol pergerakan manipulator pada saat posisi tidak mengambil objek dan posisi saat akan mengambil objek. Saat ROV telah berada diposisi lurus terhadap objek maka manipulator akan bergerak pada posisi siap untuk mengambil objek. Kontrol manipulator memberikan instruksi untuk mengaktifkan manipulator berdasarkan syarat yang ditetapkan. Apabila objek telah berada pada jangkauan 270-370 px pada sumbu x dan 190-290 px pada sumbu y maka manipulator berubah posisi siap untuk mengambil objek.

F. Inverse Kinematics

Pada perancangan gerak manipulator hal pertama yang perlu ditentukan adalah posisi awal manipulator. Posisi ini merupakan posisi saat manipulator dalam keadaan tidak aktif sehingga tidak akan mengganggu pergerakan ROV. Untuk menentukan posisi awal manipulator, maka perlu mengetahui titik koordinat *end effector* Py dan Px saat berada pada posisi awal. Koordinat Py dan Px tersebut dimasukan ke dalam persamaan *inverse kinematics* untuk menghasilkan nilai sudut θ_1, θ_2 dan θ_3. Nilai sudut tersebut digunakan untuk menggerakan motor servo.

Tabel 3.1 D-H Parameter

Joint, *i*	α_i	a_i	d_i	θ_i
1	0	10	0	θ_1
2	0	8.5	0	θ_2
3	0	7	0	θ_3

Pada perancangan *inverse kinematics* perlu memperhitungkan panjang setiap lengan. Sehingga dapat diketahui jarak maksimal jangkauan manipulator. Dalam perhitungan *inverse kinematics* gerak rotasi dan gerak penjepit tidak perlu diperhitungkan karna perhitungan ini hanya mencari besarnya sudut $\theta_1, \theta_2\ dan\ \theta_3$. Untuk dapat mencari nilai θ diasumsikan bahwa nilai kordinat telah ditentukan berdasarkan posisi pada sumbu x (q_x) dan sumbu y (q_y) dan panjang masing-masing *link*. Untuk mendapatkan nilai p_x dan p_y mengikuti Persamaan 3.9 berikut.

$$p_x = q_x - a_3\ cos\ \emptyset \tag{3.11}$$

$$p_y = q_y - a_3\ sin\ \emptyset \tag{3.12}$$

Untuk mendapatkan nilai θ_2 digunakan aturan cos dengan persamaan awal adalah sebagai berikut.

$$p_x^2 + p_y^2 = a_1^2 + a_2^2 + 2a_1a_2 \cos\theta_2 \tag{3.13}$$

Pada persamaan diatas p_x merupakan nilai koordinat x sedangkan p_y merupakan nilai koordinat y. Karena arm pada θ_2 bergerak ke arah atas pada bagian *elbow* maka digunakan pendekatan *elbow up* sebagai berikut:

$$\theta_2 = -cos^{-1}\left(\frac{p_x^2 + p_y^2 - a_1^2 - a_2^2}{2a_1a_2}\right) \tag{3.14}$$

Untuk mendapatkan nilai θ_1 diperlukan hasil dari θ_2 dengan memasukan nilai a_1 yang merupakan panjang link 1 dan a_2 yang merupakan panjang link 2. Nilai θ_1 didapatkan dengan menggunakan persamaan sebagai berikut:

$$\theta_1 = tan^{-1}\left(\frac{p_y}{p_x}\right) - cos^{-1}\left(\frac{a_2\ sin\ \theta_2}{a_1 + a_2 cos\ \theta_2}\right) \tag{3.15}$$

θ_3 didapatkan dengan cara mengurangi nilai $\emptyset$ yaitu orientasi dengan nilai sudut θ_1 dan θ_2 yang telah didapatkan sebelumnya seperti persamaan sebagai berikut:

$$\theta_3 = \emptyset - \theta_1 - \theta_2 \tag{3.16}$$

Nilai yang dihasilkan masih dalam satuan radian untuk itu agar bisa digunakan pada arm maka nilai harus dirubah dalam satuan derajat dengan persamaan sebagai berikut:

$$\theta_1 = (\theta_1 * 4068)/71 \quad (3.17)$$

$$\theta_2 = (\theta_2 * 4068)/71 \quad (3.18)$$

$$\theta_3 = (\theta_3 * 4068)/71 \quad (3.19)$$

Nilai konstanta yang didapatkan berdasarkan pada 1 rad = 57.2957746. Nilai θ yang didapatkan digunakan sebagai masukan pada motor servo. Pada posisi awal posisi manipulator melipat kearah bawah dengan nilai Px = 9.638784 dan Py = -4.25. Seperti yang ditunjukan pada Gambar 3.9, posisi untuk mengambil objek membentuk posisi tegak lurus ke depan dengan sudut maksimal yang dibentuk adalah 0^o untuk θ_2 dan θ_3. Untuk nilai x dan y pada saat posisi untuk mengambil objek adalah 25.5 dan 0.

3.4 Perancangan Manipulator Arm

Pada setiap perancangan dimulai dengan persyaratan atau pedoman yang harus diikuti sehingga desain dapat sesuai dengan apa yang diinginkan. Pedoman sendiri terbagi menjadi 2 yaitu fungsional dan non-fungsional. Fungsional yaitu menggambarkan perilaku sistem atau hal yang dapat diukur dari sistem. Non-fungsional yaitu terkait hal-hal diluar teknis seperti biaya, keamanan dan teknologi yang digunakan. Dalam perancangan manipulator ini juga berfokus pada pedoman non-fungsional dimana ada hal-hal yang tidak tercakup seperti pengaruh lingkungan terhadap kerja sistem. Manipulator yang dirancang pada penelitian ini adalah manipulator dengan 3 DOF (*Degrees of Fredom*) yang dapat bergerak kearah sumbu x dan sumbu y seperti yang terlihat pada Gambar 3.9. Adapun beberapa tahap dalam pembuatan manipulator akan dijelaskan pada bagian ini.

3.4.1 Kebutuhan Manipulator

Kebutuhan pada perancangan desain manipulator dapat dijabarkan dalam tabel 3.2 sebagai berikut.

Tabel 3.2 Tabel kebutuhan manipulator

Kebutuhan	Tipe	Deskripsi
Req 1	Fungsional	Manipulator dapat digunakan dalam air dengan kedalaman minimal 1 m.
Req 2	Fungsional	Manipulator 3 DoF yang dapat bergerak pada sumbu x dan y.
Req 3	Non-Fungsional	Aktuator yang dibuat berbentuk *gripper.*
Req 4	Non-Fungsional	Manipulator harus tahan korosi.
Req 5	Non-Fungsional	Motor servo harus kedap air.

Tabel 3.2 diatas adalah syarat yang menjadi acuan dalam membuat manipulator. Pedoman fungsional maupun non-fungsional menjadi hal yang harus dilaksanakan agar sesaui dengan hasil yang diinginkan. Terjadinya masalah diluar hal-hal yang telah dipilih akibat faktor lingkungan akan menjadi pembahasan khusus.

3.4.2 Spesifikasi Manipulator

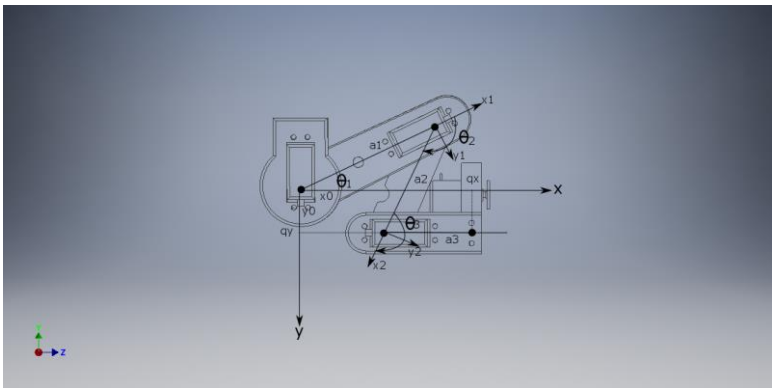
Manipulator terdiri dari 5 buah servo yang masing-masing terdapat pada bagian *shoulder, elbow, wrist, roll* dan *grip*. Pada bagian *roll* dan *grip* didesain agar dapat menjepit dan memutar pergelangan. Spesifikasi ukuran panjang lengan serta *end effector* dapat dilihat pada tabel dibawah ini.

Tabel 3.3 Spesifikasi manipulator

Bagian Manipulator	Panjang (cm)
Lengan 1	10
Lengan 2	8.5
Lengan 3	7
End Effector	15

Tabel 3.4 Sudut jangkauan motor servo

notasi	Sudut jangkauan (derajat)
θ_1	0 - 90
θ_2	30 - 150
θ_3	30 -150
End effector	20 - 50



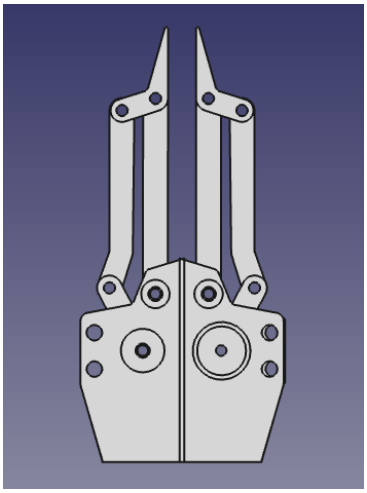
Gambar 3.9 Perencanaan posisi awal manipulator pada ROV

3.4.3 Manipulator Struktur

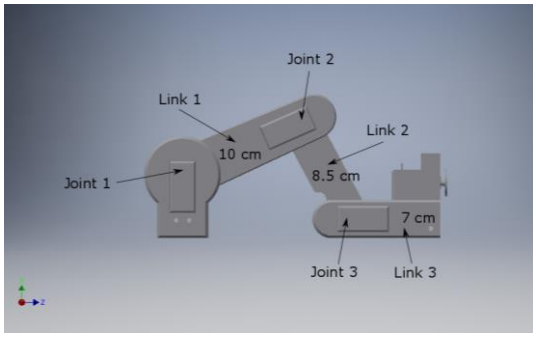
Manipulator dapat diaplikasikan pada sebuah ROV namun dengan desain yang berbeda. Setiap sendi manipulator memungkinkan untuk pergerakan pada *kartesian space*. Pada penelitian ini dirancang manipulator dengan 3 Dof seperti yang terlihat pada gambar 3.11. Lengan 1 adalah panjang lengan dari *shoulder* servo 1 ke *elbow* servo 2. Lengan 2 adalah panjang lengan dari *elbow* servo 2 ke *wrist* servo 3. Lengan 3 adalah panjang lengan dari *wrist* servo 3 ke *end effector*. Manipulator pada penelitian ini dirancang dengan pergerakan 0^o sampai dengan 180^o. Batas minimum jangkauan servo membentuk posisi awal manipulator sedangkan batas maksimum membentuk posisi tegak lurus.

3.4.4 Peralatan *End effector*

Manipulator pada penelitian ini dirancang untuk mengambil benda atau objek. Oleh karena itu *end effector* didesain berbentuk penjepit agar mudah dikontrol untuk menggenggam objek. Bentuk dan dimensi penjepit yang dibuat seperti yang terlihat pada Gambar 3.10 berikut.



Gambar 3.10 Desain gripper [28]



Gambar 3.11 Desain dan ukuran manipulator

3.4.5 Bahan Manipulator

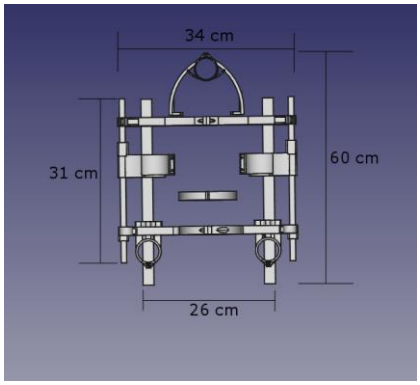
Material yang digunakan pada penelitian ini harus ringan dan kuat sehingga mampu menahan beban motor servo. Manipulator dibuat menggunakan printer 3D dengan bahan pelastik PLA dengan ketebalan 20 mm. Pada bagian sendi digunakan *bearing* berbahan besi dengan ukuran 8mm x 16mm x 5mm.

3.5 Perancangan Sistem dan Konstruksi ROV

Pada pembuatan ROV perlu diperhatikan bahan dan model ROV yang akan dibuat sehingga hasil yang diinginkan dapat terealisasi dengan baik. Adapun hal-hal yang menjadi fokus utama dalam membuat ROV seperti rangka, penggerak dan elektronik akan dijabarkan lebih mendalam.

3.5.1 Rangka Utama

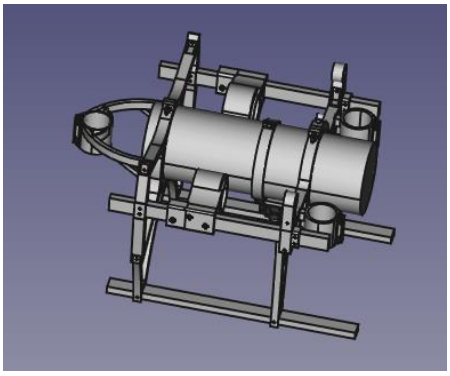
Dalam mendesain rangka ROV yang perlu diperhatikan adalah posisi ideal dari tabung dan motor penggerak yang digunakan. ROV yang didesain menggunakan 5 motor yang berada dibagian tengah, depan dan belakang. Tabung berada dibagian tengah sehingga memudahkan delam mencari CG *(center of Gravity*).



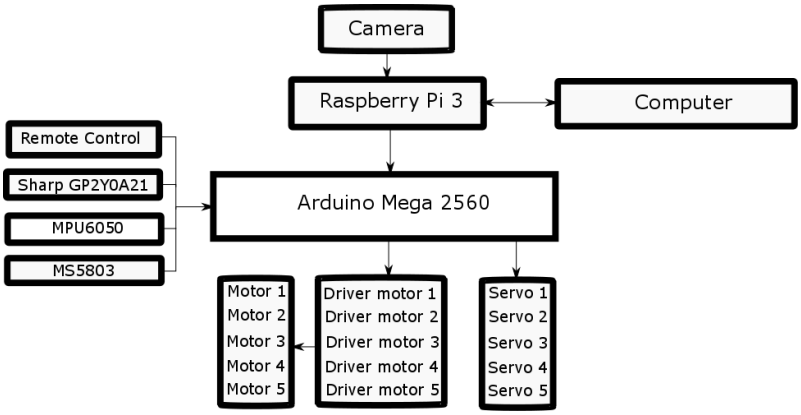
Gambar 3.12 Ukuran rangka utama ROV

3.5.2 Tabung Utama

Tabung utaman merupakan bagian yang berbentuk tabung yang berisi komponen elektronik ROV. Tabung terbuat dari pipa PVC dengan diameter 120 mm dan panjang 40 cm. Panjang tabung disesuaikan dengan panjang susunan perangkat elektronik yang terpasang dalam tabung. Pemasangan tabung ROV seperti yang terlihat pada Gambar 3.13. Perangkat elektronik yang berada dalam tabung adalah mini komputer, mikrokontroler, sensor IMU, dan *driver* motor seperti yang ditunjukan pada Gambar 3.14



Gambar 3.13 Pemasangan tabung pada rangka utama



Gambar 3.14 Diagram blok sistem ROV

ROV digerakan dengan menggunakan lima buah motor *bilge pump*. Dua motor diletakan pada bagian depan, satu pada bagian belakang dan dua pada bagian tengah. Motor pada bagian tengah dan belakang berfungsi untuk menggerakan ROV pada sumbu y ke atas dan ke bawah. Motor pada bagian tengah berfungsi untuk menggerakan ROV pada sumbu x kearah kanan, kiri, depan dan belakang. Manipulator pada ROV terdiri dari 5 buah motor servo yang dikendalikan oleh mikrokontroller.

Untuk melakukan komunikasi antara komputer dipermukaan dengan komputer ROV digunakan kabel LAN. Untuk dapat terhubung dengan komputer ROV, komputer dipermukaan dapat menggunakan aplikasi X11VNC. Aplikasi ini digunakan sebagai penghubung komputer ROV dengan komputer dipermukaan sebagai media penampil kamera ROV. Program akan mengaktifkan kamera untuk mengambil gambar secara *realtime*. Kamera mengirimkan gambar yang ditampilkan pada komputer permukaan kemudian pengguna memilih objek yang tertangkap pada kamera untuk dilakukan proses pendeteksian posisi objek.

Koordinat posisi yang dikirim tidak selalu akurat saat metode gagal mendeteksi objek maka kordinat yang dikirim adalah 0. Untuk menghindari kesalahan pembacaan data dan memisahkan data kordinat sumbu x dan sumbu y maka perlu dilakukan pasring data. Parsing data adalah mengolah paket data dengan cara memisahkan data-data tersebut ke dalam uraian-uraian data. Cara ini digunakan dengan tujuan mempermudah komunikasi data karena metode parsing hanya perlu memberi satu line data yang didalamnya mencakup banyak data. Proses parsing data yang dilakukan pada mikrokontroler.

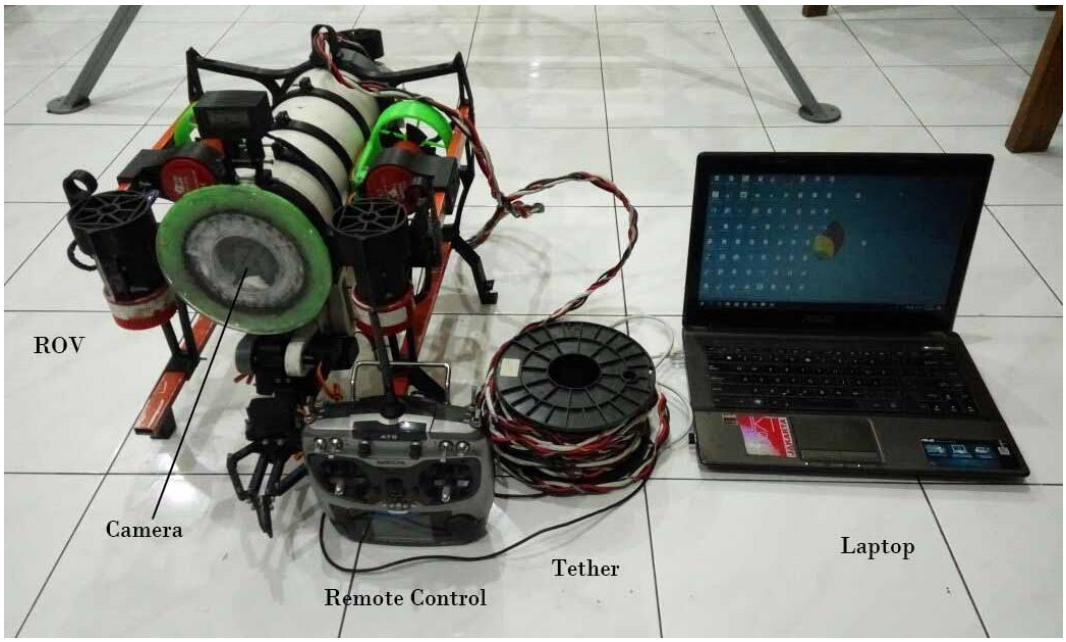
Halaman ini sengaja dikosongkan

BAB 4
HASIL DAN PEMBAHASAN

Pada bagian ini akan dibahas mengenai mekanisme, tahapan pengujian sistem, dan analisa. Adapun beberapa pengujian yang akan dilakukan untuk memvalidasi kemampuan dari sistem yang dibangun. Pada bagian ini akan disajikan data-data hasil pengujian.

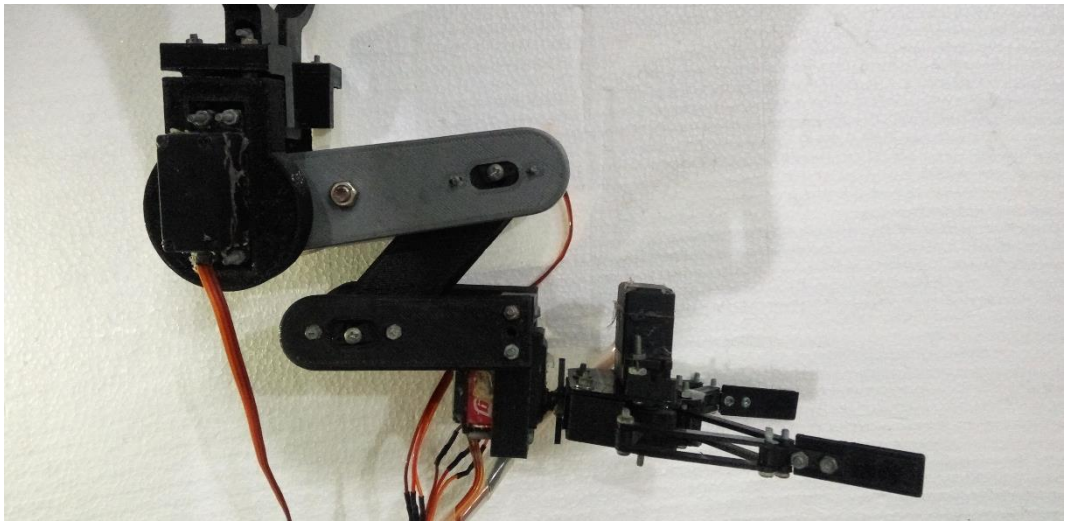
4.1 Keseluruhan Sistem ROV

ROV yang dirancang telah berhasil direalisasikan dan diuji coba pada aquarium dan kolam renang. Peralatan ROV terdiri dari laptop, remot kontrol, *tether*, dan aki sebagai catu daya seperti yang terlihat pada Gambar 4.1. ROV didesain dengan kondisi melayang dalam air. Sehingga tidak membebani motor saat akan bergerak didalam air. Bagian depan tabung menggunakan akrilik bening dengan tujuan agar kamera dapat mengambil gambar dengan baik.



Gambar 4.1 Keseluruhan Sistem ROV

Gambar 4.2 ROV Tampak Atas



Gambar 4.3 Manipulator arm ROV

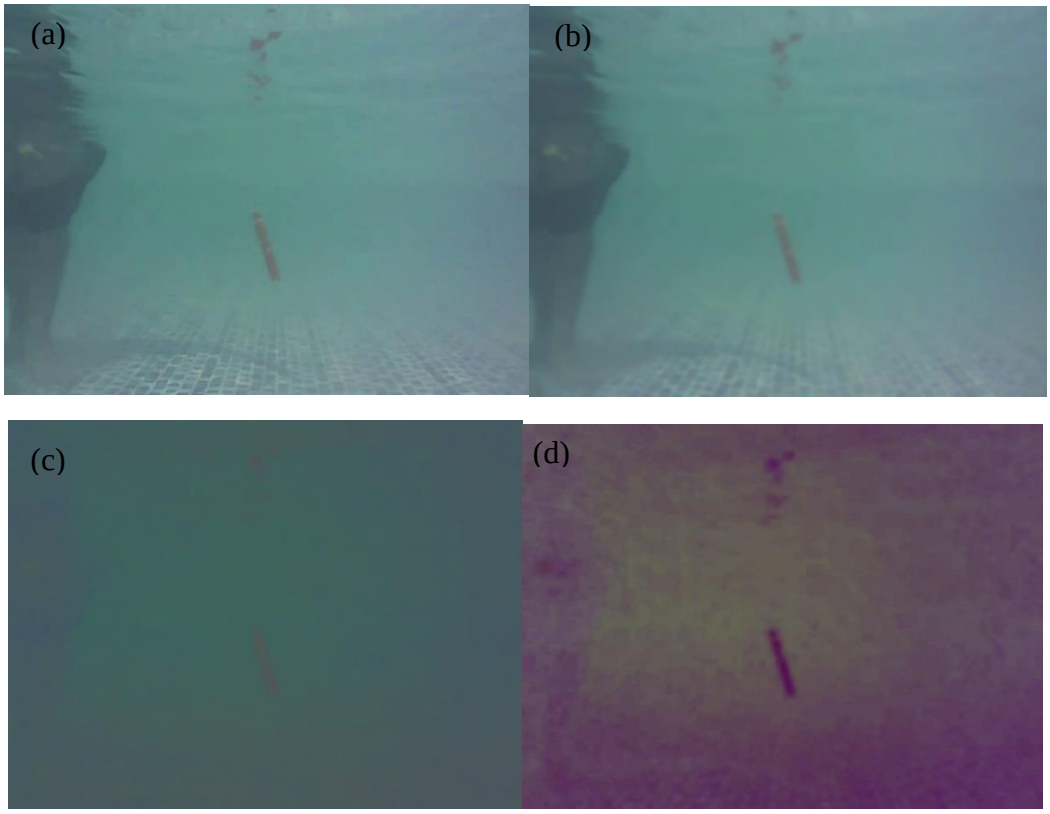
4.2 Pengujian Pendeteksian Objek

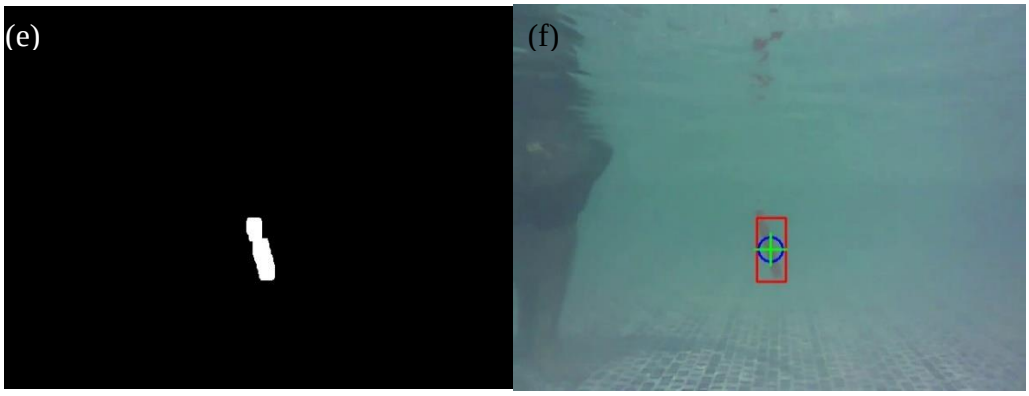
Pengujian ini bertujuan untuk mengetahui performa metode yang dikembangkan dalam mendeteksi objek yang bergerak. Pengujian ini menggunakan kamera Logitech C170 dalam lingkungan pengujian yang terkontrol. Pengujian ini dibagi dalam beberapa bagian yaitu pengujian pendeteksian kontur objek, pengujian akurasi penjejakan objek dengan variabel kecepatan, pengujian

penjejakan objek dengan variabel jarak, pengujian akurasi penjejakan objek, dan respon ROV saat penjejakan objek. Pada pengujian ini menggunakan *scene* yang sama. Pengujian dilakukan di aquarium percobaan laboratorium dan kolam renang.

4.2.1 Pengujian Pendeteksian Kontur Objek

Pada pengujian ini bertujuan untuk mengetahui kemampuan metode yang dikembangkan dalam mengekstraksi bentuk dan posisi objek dalam air. Pengujian ini dilakukan dengan cara melakukan klik pada video dalam air. Tujuan melakukan klik pada gambar adalah untuk menentukan objek yang akan dijejaki. Objek yang digunakan berbentuk pipa dengan panjang 10 cm berwarna *orange* seperti yang ditunjukan pada Gambar 4.4.





Gambar 4.4 (a) Gambar RGB; (b) Gambar setelah gaussian blur; (c) Gambar setelah normalisasi RGB; (d) Gambar HSV; (e) Gambar objek yang telah ditandai.

Berdasarkan hasil pengujian pendeteksian kontur objek dengan menggunakan *scene* objek yang berada dalam air, dapat disimpulkan bahwa metode berhasil mendeteksi objek dalam air. Metode dapat mengikuti objek walaupun digerakan secara tidak beraturan. Saat objek berhasil terdeteksi, luasan kontur memberikan informasi titik tengah objek sebagai posisi aktual pada sumbu x dan sumbu y.

4.2.2 Pengujian Akurasi Pendeteksian Objek Dengan Variabel Kecepatan

Pengujian ini bertujuan untuk mengetahui akurasi pendeteksian saat objek digerakan dengan kecepatan tertentu. Gagalnya proses pendeteksian akan mempengaruhi data yang dikirim ke mikrokontroler. Untuk menghindari terjadinya kesalahan penjejakan maka perlu dilakukan pengujian untuk dapat mengetahui batas maksimal kecepatan objek bergerak. Semakin cepat objek bergerak maka semakin besar kemungkinan metode untuk gagal dalam mendeteksi.

Pengujian ini dilakukan dengan cara merekam video pada akuarium, kemudian objek digerakan dengan berbagai variasi kecepatan. Pada program masing-masing metode, setiap *frame* yang mendeteksi objek ditandai dengan nilai 1 dan *frame* yang tidak mendeteksi objek ditandai dengan nilai 0. Kemudian nilai hasil deteksi disimpan pada sebuah file dan dihitung tingkat keberhasilannya. Tingkat keberhasilan dihitung berdasarkan total jumlah *frame* dan *frame* yang terdeteksi. Pengujian dilakukan di akuarium percobaan dengan objek yang sama

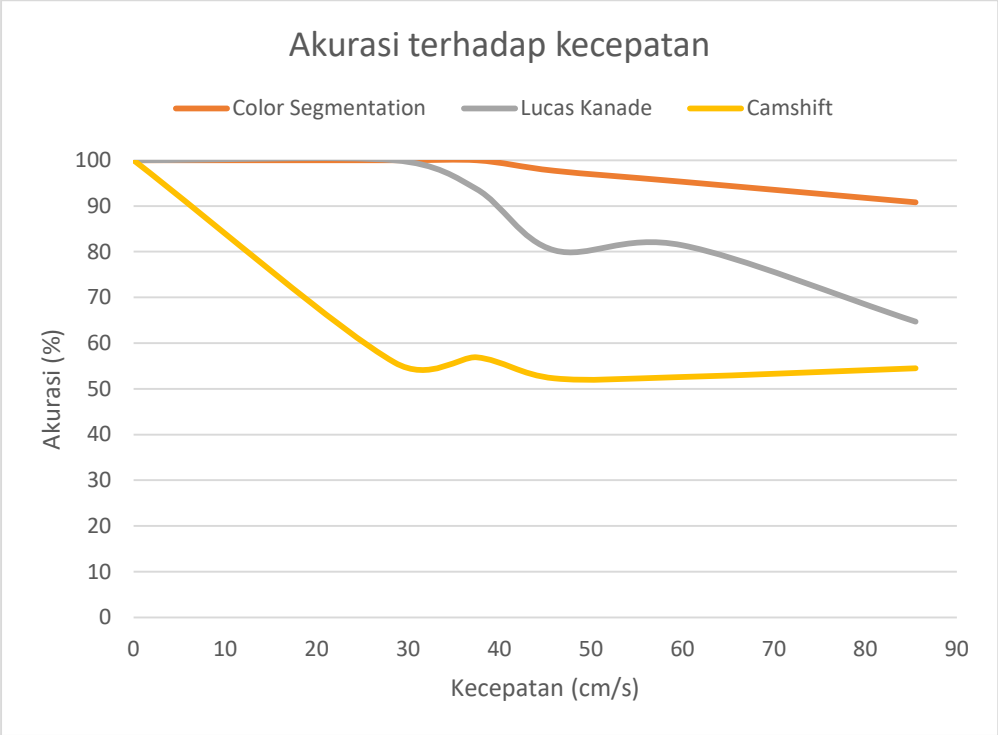
yaitu pipa berukuran panjang 10 cm dengan warna *orange*. Objek berjarak 60 cm dari kamera dan digerakan dengan diatur kecepatannya.

Dapat terlihat dari pada gambar 4.5 metode yang dikembangkan berhasil mendeteksi objek yang bergerak dengan kecepatan tertentu. Karena keterbatasan kemampuan kamera saat mengambil gambar yang bergerak dengan pencahayaan terbatas, mengakibatkan bentuk objek yang terdeteksi mengalami perubahan bentuk. Untuk memvalidasi kemampuan dari metode yang dikembangkan maka dilakukan perbandingan dengan beberapa metode penjejakan seperti yang terlihat pada Tabel 4.1.

Gambar 4.5 Gambar hasil Pendeteksian objek

Tabel 4.1 Hasil pengukuran tingkat akurasi

Metode	scene video										Akurasi %
	Jarak (cm)	60	Jarak (cm)	60	Jarak (cm)	60	Jarak (cm)	60	Jarak (cm)	60	
	Waktu (s)	2.1	Waktu (s)	1.6	Waktu (s)	1.3	Waktu (s)	1	Waktu (s)	0.7	
	Kecepatan objek (cm/s)	28.6	Kecepatan objek (cm/s)	37.5	Kecepatan objek (cm/s)	46.154	Kecepatan objek (cm/s)	60	Kecepatan objek (cm/s)	85.71	
Color Segmentation	100		100		97.7		95.3		90.8		96.76
Lucas Kanade	100		93.7		80.3		81.4		64.7		84.02
Camshift	55.8		70		52.3		52.6		54.5		57.04



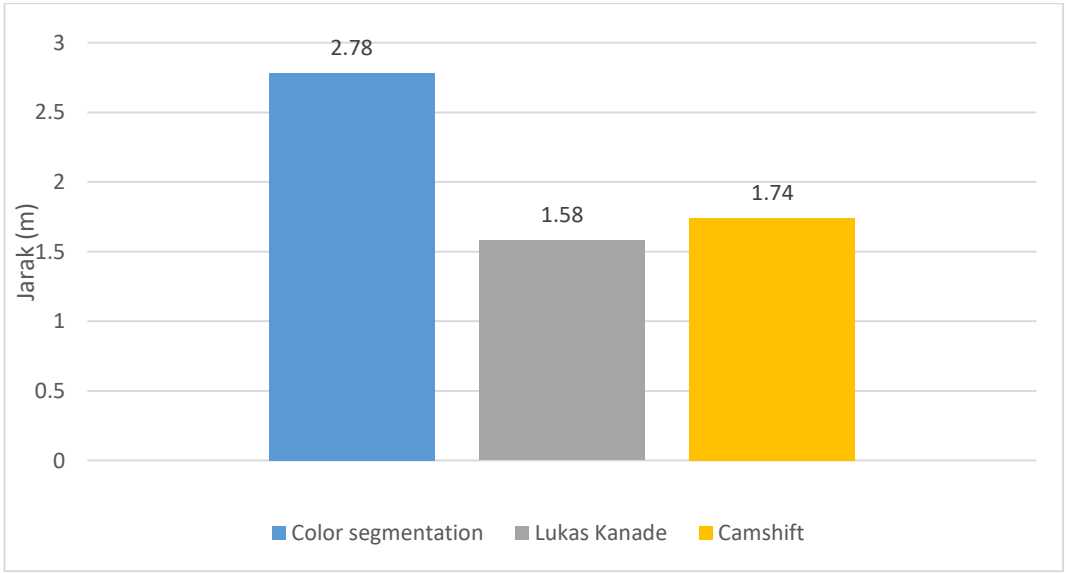
Gambar 4.6 Perbandingan tingkat akurasi

Pada tabel 4.6 diatas dapat terlihat kemampuan dari metode yang dikembangkan dengan presentase keberhasilan dalam mendeteksi objek. Percobaan dilakukan dengan 5 variasi kecepatan yaitu 28.6 cm/s, 37.5 cm/s, 46.15 cm/s, 60 cm/s dan 85.7 cm/s. Semakin cepat pergerakan benda maka semakin berkurang tingkat akurasinya seperti yang terlihat pada Gambar 4.6. Saat dilakukan percobaan dengan kecepatan rendah masing-masing metode yang diuji mampu mendeteksi objek dengan baik. Saat kecepatan objek diubah lebih cepat maka masing-masing metode akan mengalami penurunan akurasi.

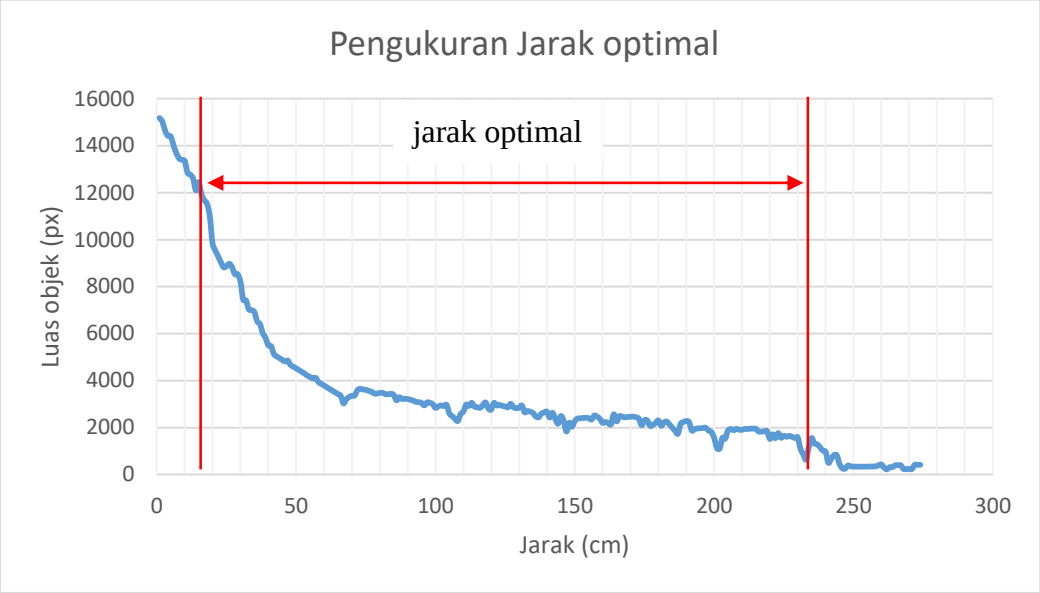
4.2.3 Pengujian Akurasi Pendeteksian Objek Dengan Variabel Jarak

Pengujian selanjutnya adalah pengujian akurasi pendektesian objek dengan variabel jarak. Pengujian ini bertujuan untuk mengetahui seberapa jauh jarak maksimal objek yang dapat dideteksi. Percobaan dilakukan dengan mengambil video objek yang berada dikolam renang kemudian objek akan digerakan menjauh dari kamera secara perlahan. Video yang direkam menunjukan jarak lintasan hingga dapat diketahui jarak maksimal metode saat tidak dapat lagi mendeteksi posisi objek. Hasil video kemudian digunakan untuk menguji masing-masing metode. Percobaan dilakukan dengan menggunakan *scene* yang sama dan dibandingkan dengan beberapa metode penjejakan. Percobaan ini dilakukan di kolam renang pada waktu siang hari. Objek yang digunakan adalah sebuah pipa dengan panjang 10 cm berwarna *orange*.

Dari hasil pengujian dapat terlihat bahwa metode yang dikembangkan dapat mendeteksi objek dengan jarak lebih jauh dari metode pembanding. Pengujian yang dilakukan menghasilkan jarak maksimal pengukuran yaitu 2.78 m.



Gambar 4.7 Jarak pengukuran maksimum

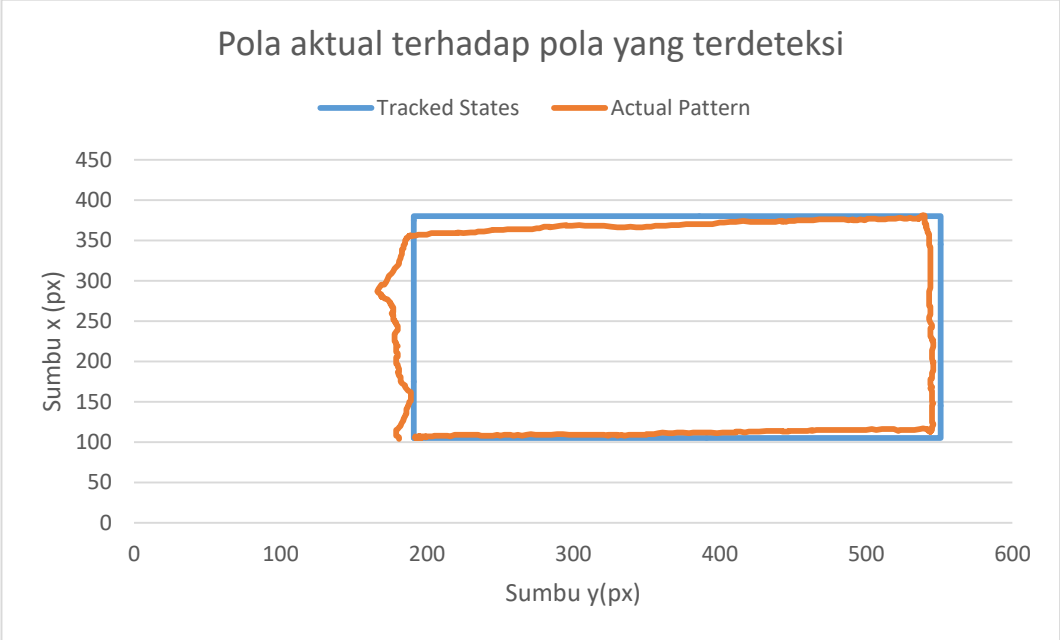


Gambar 4.8 Jarak optimal pendeteksian objek

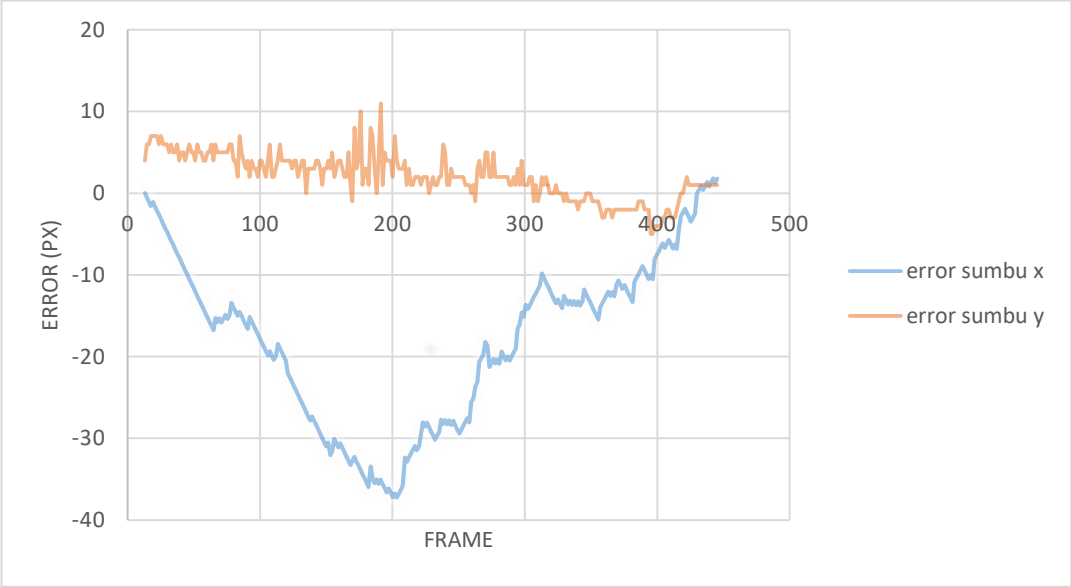
Jarak optimal yang diperoleh, menggambarkan luasan objek yang dapat terdeteksi saat objek berada dijarak terdekat dari kamera dan saat objek berada dijarak terjauh dari kamera. Dari pengujian yang dilakukan dapat diketahui jarak optimal dalam pendeteksian objek adalah 20 cm sampai 2.4 m. Luasan permukaan objek yang terdeteksi mempengaruhi jarak maksimal yang dapat dideteksi. Seperti yang dapat terlihat pada Gambar 4.8 diatas semakin jauh jarak kamera dengan objek maka semakin kecil luasan permukaan objek yang terdeteksi. Begitu pula sebaliknya semakin dekat jarak objek dengan kamera maka semakin besar luasan permukaan objek yang terdeteksi.

4.2.4 Pengujian Akurasi Posisi Penjejakan Objek

Pengujian akurasi penjejakan objek adalah pengujian untuk menghitung tingkat akurasi posisi yang terdeteksi terhadap posisi aktual objek. Pengujian ini dilakukan dengan membuat jalur gerakan objek yang simetris dan presisi kemudian objek digerakan mengikuti jalur tersebut. Sumber pencahayaan yang digunakan merupakan sumber cahaya tetap yang berada diatas permukaan objek. Secara teknis hasil deteksi akan membentuk pola yang sesuai dengan pola jalur yang telah dibuat. Keakuratan posisi pola jalur yang terbentuk akan mempengaruhi seberapa akurat metode dalam mendeteksi posisi aktual objek.



Gambar 4.9 Akurasi posisi penjejakan objek



Gambar 4.10 *Error* posisi pada sumbu x dan sumbu y

Gambar 4.9 menunjukan hasil pengujian perbandingan antara jalur pola aktual dengan jalur pola hasil penjejakan. Berdasarkan data yang diperoleh semakin jauh jarak kamera ROV terhadap objek maka semakin besar kesalahan perkiraan posisi terhadap pola jalur yang sebenarnya. Hal ini berkaitan dengan pergeseran titik tengah objek. Pada saat objek bergerak ke arah kiri, cahaya yang mengenai permukaan objek menjadi tidak merata. Hal ini menyebabkan pencahayaan menjadi

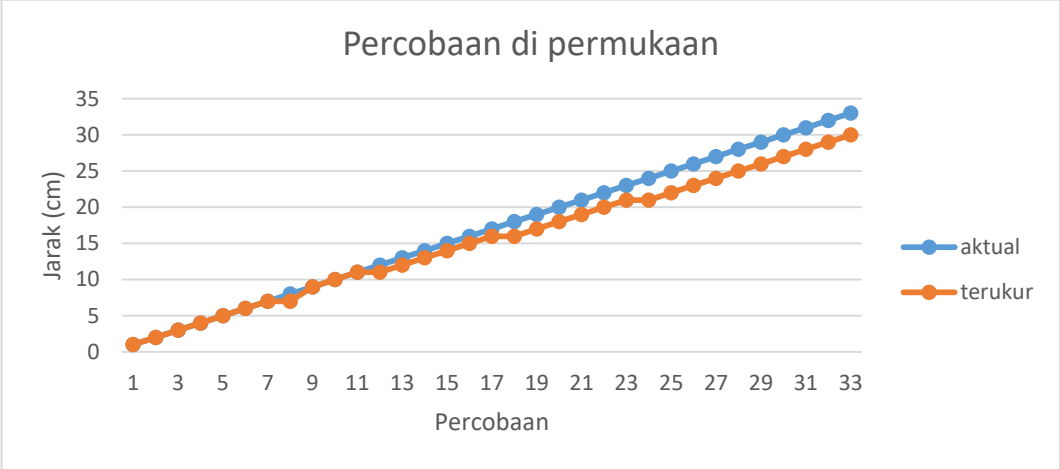
tidak merata dan menghasilkan sisi objek yang lebih gelap daripada sisi lainnya. Penyebaran cahaya yang tidak merata ini menyebabkan perubahan bentuk kontur objek. Secara tidak langsung titik tengah objek mengalami pergeseran secara terus menerus saat objek bergerak menjauh dari sumber cahaya. Pada gambar 4.10 dapat terlihat persebaran *error* posisi pada setiap *frame* masing-masing sumbu. Jumlah *error* didominasi oleh sumbu x dengan *error* tertinggi mencapai 38 px.

4.3 Pengujian Pengukuran Jarak

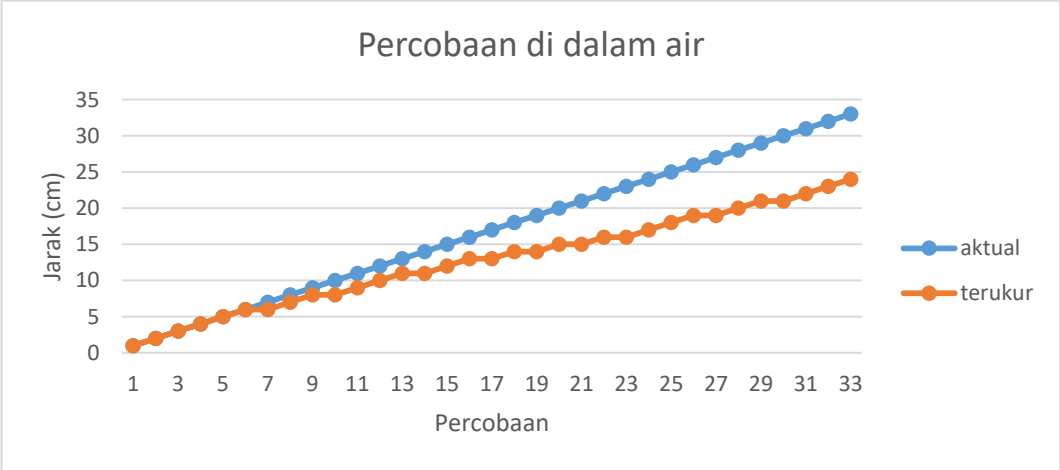
Tujuan dari pengujian ini adalah untuk mengetahui akurasi sensor dalam mengukur jarak aktual antara ROV dan objek. Sensor yang dipasang dibagian dalam tabung ROV akan mengukur jarak aktual antara ROV dan objek. Pengujian dilakukan dengan cara memindahkan secara perlahan objek yang berada didepan sensor hingga objek tidak dapat dideteksi. Kemudian setiap perpindahan jarak akan dicatat jarak aktual dan jarak yang terdeteksi oleh sensor.

Tabel 4.2 Hasil pembacaan sensor jarak

Jarak aktual (cm)	Jarak terdeteksi (cm)	Jarak aktual (cm)	Jarak terdeteksi (cm)
1	1	18	14
2	2	19	14
3	3	20	15
4	4	21	15
5	5	22	16
6	6	23	16
7	6	24	17
8	7	25	18
9	8	26	19
10	8	27	19
11	9	28	20
12	10	29	21
13	11	30	21
14	11	31	22
15	12	32	23
16	13	33	24
17	13		



Gambar 4.11 Grafik pengukuran jarak di permukaan



Gambar 4.12 Grafik pengukuran jarak di dalam air

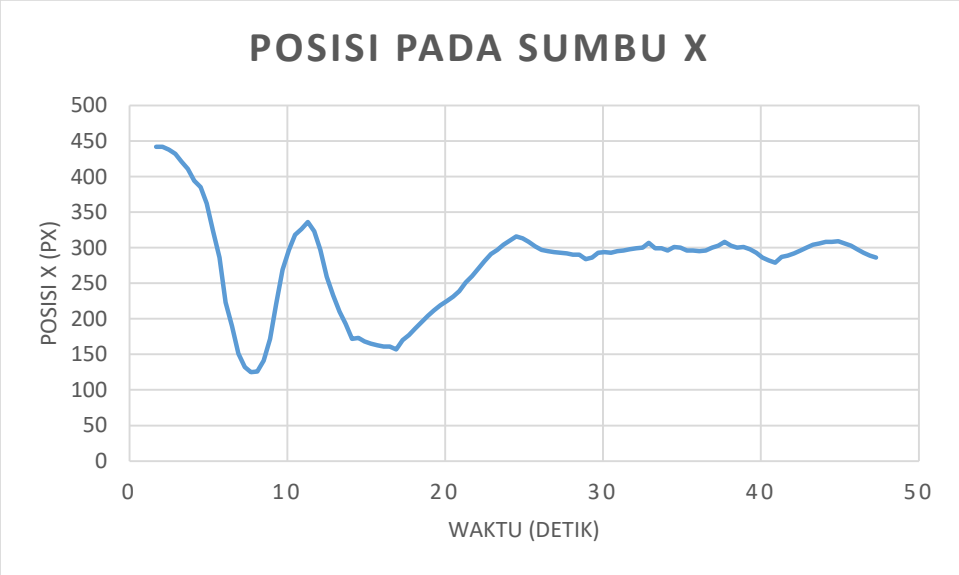
Pengujian ini dilakukan di dua tempat yaitu diatas permukaan air dan dalam air. Pada pengujian diatas permukaan air seperti yang terlihat pada Gambar 4.12, nilai pembacaan sensor dibandingkan dengan nilai aktual. Hasil dari pembacaan terjadi *error* atau selisih pembacaan pada sensor sekitar 3 cm pada jarak 33 cm. Perbedaan pembacaan sensor dimulai pada jarak 12 cm dengan pembacaan sensor 11 cm. Kemudian pada jarak 18 cm pembacaan sensor adalah 16 cm. Maksimal pembacaan sensor adalah 33 cm lebih dari itu data yang dihasilkan tidak akurat dan cenderung berubah-ubah. Pada pengujian pengukuran didalam air seperti yang terlihat pada Gambar 4.12, Selisih pembacaan dimulai pada jarak 6 cm. Semakin jauh jarak antara objek dan kamera maka semakin besar selisih yang dihasilkan.

4.4 Pengujian Respon ROV Saat Pendektesian Objek

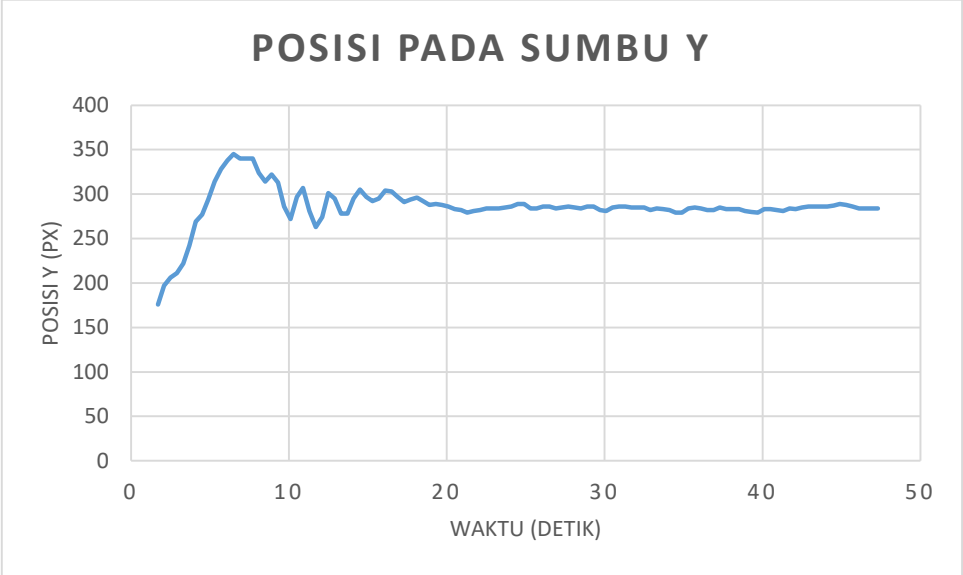
Pengujian ini merupakan pengujian kecepatan respon ROV terhadap posisi objek untuk mencapai posisi stabil atau sejajar dengan objek. Posisi stabil atau sejajar dengan objek adalah posisi saat masing-masing sumbu menghasilkan nilai yang mendekati nilai *setpoint*. Tujuan pengujian ini adalah untuk mengetahui respon sistem dalam penjejakan objek. Pengujian dilakukan dengan cara meletakan objek didepan ROV. Kemudian saat ROV diaktifkan setiap perubahan posisi dicatat dan dianalisa. Perubahan posisi merupakan respon pergerakan ROV dari titik awal menuju titik akhir. Pengujian dilakukan di dua tempat yaitu pada aquarium percobaan dan pada kolam renang.

4.4.1 Pengujian Pada Aquarium Percobaan di Laboratorium

Pengujian pada aquarium percobaan dilakukan pada lingkungan yang terkontrol. Pengujian dilakukan pada siang hari dengan lingkungan percobaan bebas dari gangguan seperti arus, kekeruhan air, dan pencahayaan yang tidak stabil. objek adalah benda berbentuk pipa dengan diameter 6 cm dan panjang 28 cm. Pengujian dilakukan dengan cara meletakan objek didepan ROV kemudian catat respon pergerakan ROV mencapai posisi stabil atau sejajar terhadap objek.



Gambar 4.13 Respon sistem pada sumbu x pengujian di aquarium

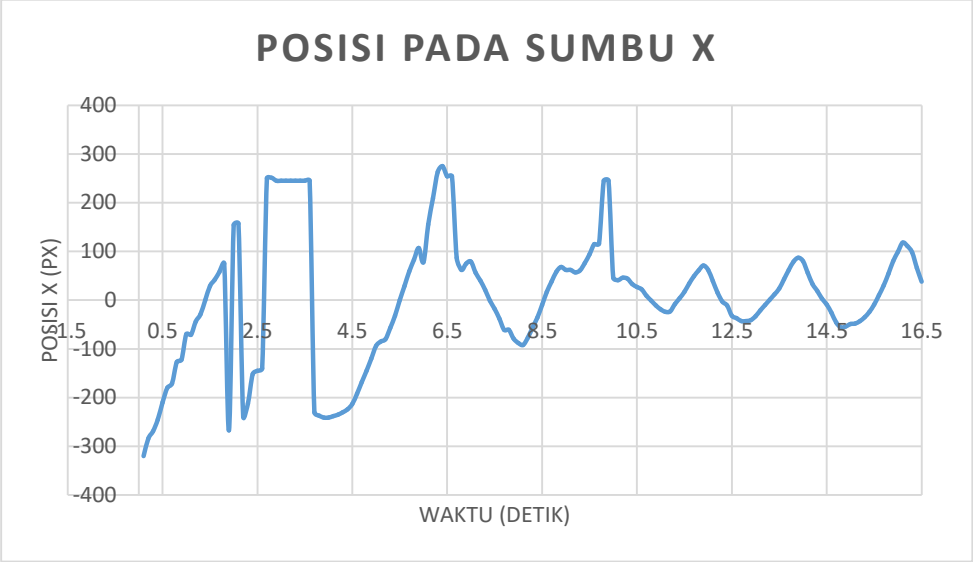


Gambar 4.14 Respon sistem pada sumbu y pengujian di aquarium

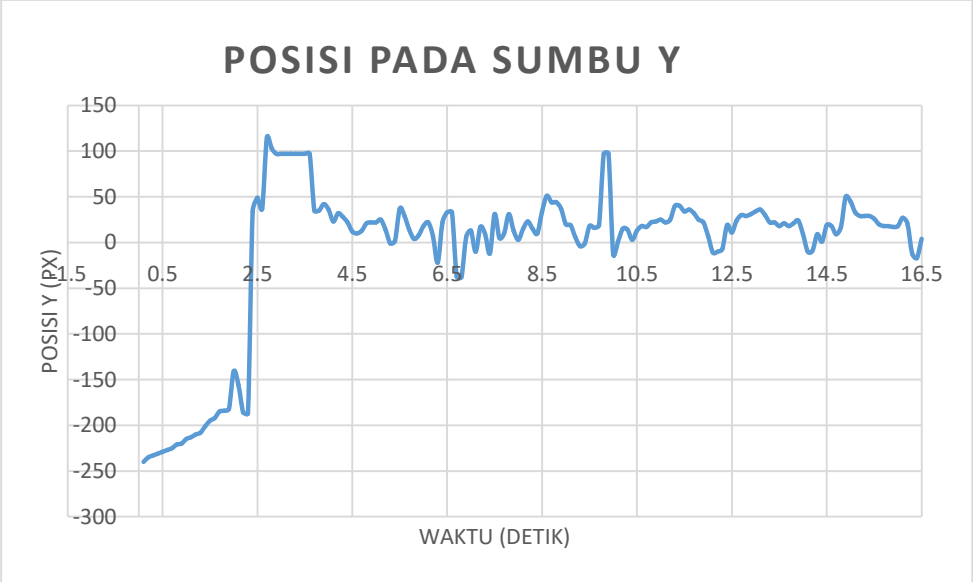
Berdasarakan hasil percobaan, waktu yang dibutuhkan sitem untuk mencapai *steady state* pada sumbu x dan sumbu y memerlukan waktu sekitar 22 detik. Seperti yang terlihat pada Gambar 4.13 dan Gambar 4.14. Kondisi lingkungan percobaan yang bebas dari gelombang membuat ROV dapat mempertahankan posisi terhadap objek dengan *error* yang kecil.

4.4.2 Pengujian Pada Kolam Renang

Percobaan selanjutnya dilakukan di kolam renang. Lingkungan percobaan kali ini berbeda dengan percobaan pada aquarium sebelumnya. Faktor gelombang, kejernihan, dan pencahayaan yang tidak terkontrol akan mempengaruhi hasil pengujian. Pengujian dilakukan dengan cara meletakan objek didepan ROV dengan jarak sekitar 1.5 m kemudian dicatat respon pergerakan ROV mencapai posisi stabil atau sejajar terhadap objek. Pada pengujian ini posisi objek dibuat tetap.



Gambar 4.15 Respon sistem pada sumbu x pengujian kolam renang



Gambar 4.16 Respon sistem pada sumbu y pengujian kolam renang

Hasil yang didapatkan pada pengujian ini berbeda dengan pengujian sebelumnya. Gangguan seperti gelombang membuat ROV mengalami osilasi dan sulit mencapai posisi stabil. Untuk mendapatkan posisi sejajar terhadap objek dibutuhkan waktu sekitar 10 detik namun akibat gangguan gelombang air menyebabkan ROV tidak dapat mempertahankan posisi *steady state*. Perubahan intensitas cahaya yang diakibatkan pengaruh gelombang diatas permukaan air

membuat cahaya yang mengenai permukaan objek menjadi tidak merata. Kondisi seperti ini membuat metode tidak maksimal bahkan cenderung gagal dalam mendeteksi objek. Objek yang menjadi target mengalami pergerakan diakibatkan gelombang dari kolam sehingga titik tengah objek yang menjadi fokus mengalami pergeseran. Setiap pergerakan objek mempengaruhi respon ROV akibatnya ROV tidak dapat stabil karena selalu mengikuti pergerakan objek. Osilasi pada sumbu x dan sumbu y dapat dilihat pada Gambar 4.16 dan Gambar 4.17. Terlihat bahwa saat detik ke 10, ROV telah mampu untuk sejajar dengan objek namun tidak bisa mempertahankan posisi stabil akibat adanya gangguan.

4.5 Evaluasi Kebutuhan Manipulator

Evaluasi kebutuhan manipulator adalah hasil pengujian manipulator yang berdasarkan pada kebutuhan yang telah dirancang sebelumnya. Evaluasi perlu dilakukan untuk memastikan perancangan yang dilakukan sudah sesuai dengan pedoman awal perancangan yang telah dibuat. hasil dari evaluasi ini dapat dilihat pada Tabel 4.3 berikut.

Tabel 4.3 Evaluasi kebutuhan manipulator

Requirement	Tipe	Deskripsi	Keberhasilan
Req 1	Fungsional	Manipulator dapat digunakan dalam air dengan kedalaman minimal 1 m.	Berhasil
Req 2	Fungsional	Manipulator 3 DoF yang dapat bergerak pada sumbu x dan y.	Berhasil
Req 3	Non-Fungsional	Aktuator yang dibuat berbentuk *gripper*.	Berhasil
Req 4	Non-Fungsional	Manipulator harus tahan korosi.	Tidak berhasil
Req 5	Non-Fungsional	Motor servo harus kedap air.	Tidak berhasil

Manipulator ROV telah sepenuhnya direalisasikan dengan berpedoman pada prasyarat yang telah ditentukan. Hasil dari percobaan manipulator sangat dipengaruhi oleh faktor lingkungan. Manipulator telah berhasil digunakan pada kedalaman 1.5 m. Pergerakan manipulator telah sesuai dengan gerakan yang dinginkan. Manipulator yang didesain memiliki *end-effector* berbentuk jepit yang bertujuan untuk mengambil objek yang berada didalam air. Data untuk hasil percobaan manipulator akan dibahas pada sub bab berikutnya.

Pada percobaan yang dilakukan, bagian manipulator yaitu *bearing* mengalami korosi hal ini dikarenakan bearing terbuat dari bahan logam. *Bearing* menjadi salah satu bagian yang penting dalam manipulator sebab setiap sendi pada manipulator menggunakan *bearing* sebagai poros berputarnya *link*. Motor servo yang digunakan masih mengalami kebocoran sehingga air masuk ke dalam motor penggerak servo. Hal ini mengakibatkan motor servo tidak dapat bekerja dengan baik. Air yang masuk ke dalam motor servo menyebabkan karat yang menumpuk dan membuat rangkaian elektronik servo menjadi panas.

4.6 Pengujian Waktu Mencapai Titik Koordinat

Pengujian ini adalah pengujian untuk mengetahui waktu yang dibutuhkan motor servo menggerakan manipulator ke koordinat yang telah ditentukan. Pengujian ini dilakukan dengan cara memasukan nilai koordinat pada persamaan *inverse kinematics*. Nilai sudut yang didapatkan akan menggerakan motor servo. Pengujian dilakukan dengan memasukan 5 koordinat yang berbeda. Perhitungan waktu dilakukan secara manual dengan menggunakan *stopwatch* dan dilakukan beberapa kali kemudian dihitung waktu rata-ratanya. Hasil yang didapatkan seperti yang terlihat pada Tabel 4.4 dibawah ini:

Tabel 4.4 Pengujian waktu mencapai titik koodinat

Gerakan	Koordinat awal	Koordinat akhir	Waktu rata-rata (detik)
1	(9.63,-4.25)	(15.5,-8.35)	0.5
2	(9.63,-4.25)	(19,-8)	0.7
3	(9.63,-4.25)	(21,-8)	0.75
4	(9.63,-4.25)	(23,-4)	0.8
5	(9.63,-4.25)	(25.5,0)	1

Dari pengujian yang dilakukan, waktu yang dibutuhkan manipulator untuk mencapai posisi tegak lurus dari posisi awal dengan sudut 0^o adalah 1 detik. Kecepatan gerakan manipulator menyebabkan ROV mengalami goncangan saat manipulator aktif. Hal ini akan dibahas pada pengujian selanjutnya.

4.7 Pengujian Akurasi Sudut Servo

Pengujian akurasi sudut servo adalah pengujian untuk mengetahui besarnya sudut yang dihasilkan oleh servo dengan nilai yang telah ditentukan. Tujuan dari pengujian ini adalah untuk mengetahui besarnya *error* sudut terhadap sudut yang telah ditentukan. Pengujian dilakukan dengan memasukan nilai sudut pada servo kemudian hasil pergerakan pada manipulator akan diukur besar sudut yang dihasilkan. Nilai sudut tersebut kemudian dibandingkan dengan sudut yang sebenarnya. Dari pengukuran akan didapatkan besarnya nilai *error* pada setiap sendi manipulator.

Tabel 4.5 Ketepatan sudut servo 1

koordinat (cm)		sudut theta (derejat)	sudut theta terukur (derajat)	selisih	error (%)
x	y				
25.5	0	0	0	0	0
25.3	-1.4	10	10	0	0
24.9	-2.9	20	15	5	25
24.3	-4.25	30	25	5	16.66667
22.4	-6.5	50	45	5	10
19.9	-7.9	70	65	5	7.142857
17	-8.5	90	80	10	11.11111
12	-7.9	110	100	10	9.090909
11.5	-6.5	130	120	10	7.692308
9.6	-4.25	150	145	5	3.333333

Tabel 4.6 Ketepatan sudut servo 2

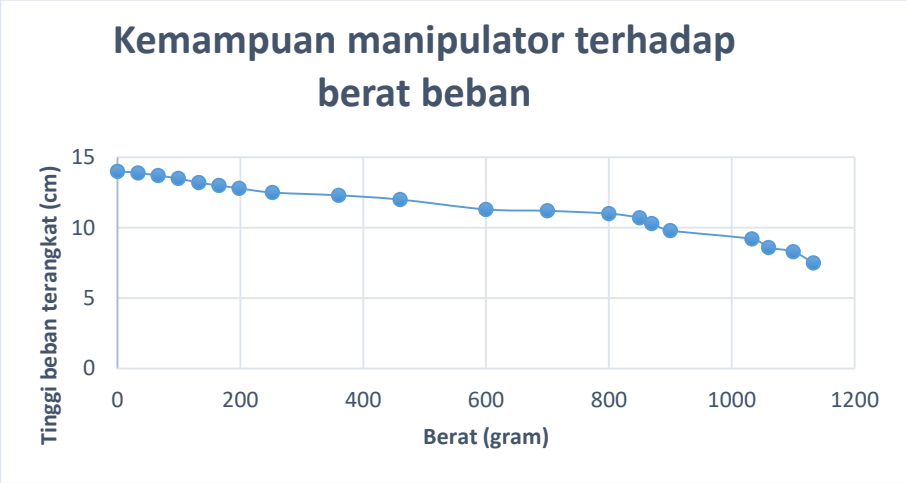
koordinat (cm)		sudut theta (derajat)	sudut theta terukur (derajat)	selisih	error (%)
x	y				
25.5	0	0	0	0	0
25.3	-1.4	10	5	5	50
24.9	-2.9	20	15	5	25
24.3	-4.25	30	25	5	16.66667
22.4	-6.5	50	50	0	0
19.9	-7.9	70	75	5	7.142857
17	-8.5	90	95	5	5.555556
12	-7.9	110	115	5	4.545455
11.5	-6.5	130	135	5	3.846154
9.6	-4.25	150	140	10	6.666667

Dari hasil percobaan yang dilakukan dapat terlihat selisih sudut yang menjadi referensi dengan sudut aktual yang terukur. *Error* yang dihasilkan beragam, dengan *error* maksimal adalah 16%. Hal ini akibat berat motor servo penjepit yang membebani motor servo pada masing-masing sendi. Dengan adanya beban pada masing-masing *link* membuat servo mengalami perubahan sudut semakin besar panjang *link* dan beban penjepit maka semakin besar pula *error* yang dihasilkan.

4.8 Pengujian Kemampuan Manipulator Terhadap Beban

Tujuan dari pengujian ini untuk mengukur kemampuan manipulator untuk mengangkat beban. Kemampuan mengangkat beban diukur dari tinggi manipulator saat diberikan beban. Pengujian dilakukan dengan cara memposisikan manipulator tegak lurus dengan sudut pada masing-masing servo yaitu 0^o. Kemudian diberikan beban berupa baut besi dengan kenaikan 10 gram. Setiap pertambahan beban akan diukur perubahan tinggi penjepit sehingga dapat diketahui kemampuan manipulator dalam mengangkat objek.

Dari hasil pengujian yang dilakukan kemampuan manipulator untuk mengangkat beban mengalami penurunan setiap beban ditambahkan. Tinggi manipulator saat belum mendapatkan beban adalah 14 cm kemudian ditambahkan beban secara perlahan maka tinggi manipulator mengalami penurunan. Pengujian dilakukan hingga berat beban 1100 gr dan tinggi manipulator menurun hingga sekitar 7 cm.



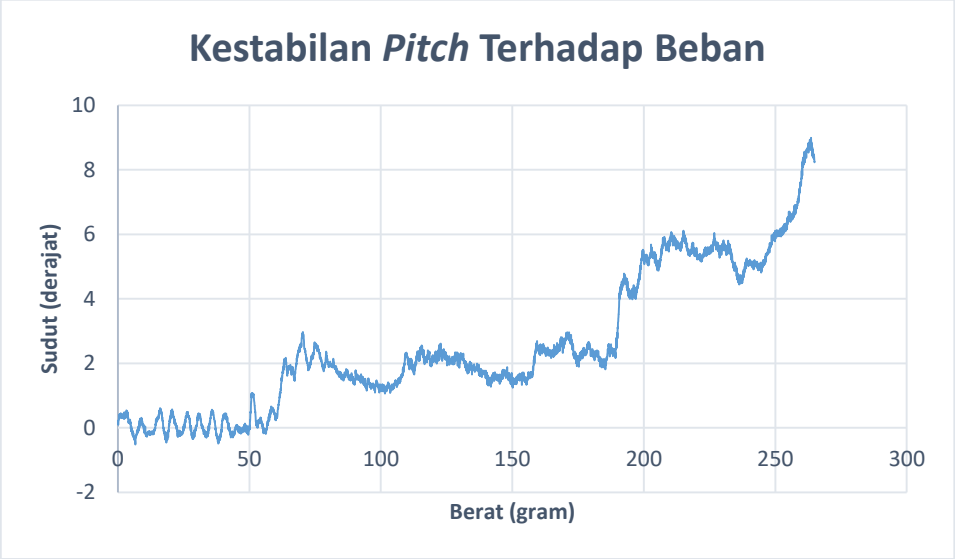
Gambar 4.17 Pengujian beban manipulator

Kemampuan manipulator menurun hingga setengah dari tinggi awal. Hal ini menyebabkan motor servo menjadi panas. Untuk menghindari kerusakan maka batas maksimal beban yang diberikan adalah 1200 gr. Kemampuan manipulator untuk mengangkat beban tidak berhubungan dengan daya angkat ROV. Sebab kemampuan motor ROV untuk mengangkat beban terbatas. Kemudian harus ditambahkan dengan beban manipulator. Sehingga hasil yang didapatkan sangatlah berbeda.

4.9 Pengujian Kestabilan *Pitch* Terhadap Beban

Pengujian ini adalah pengujian untuk melihat kestabilan sudut *pitch* ROV terhadap beban saat mengangkat objek dalam air. Tujuan dari pengujian ini adalah untuk melihat posisi *pitch* ROV saat diberikan beban pada manipulator. Pengujian dilakukan dengan cara memberikan beban pada manipulator dengan kenaikan 10 gram. Setiap penambahan beban akan diukur perubahan sudut pitch sehingga dapat diketahui beban maksimum ROV saat mengambil objek.

Dari pengujian kestabilan *pitch* terhadap beban yang telah dilakukan dapat terlihat bahwa ROV dapat mempertahankan keseimbangan sudut *pitch* dengan beban sekitar 50 gr. Saat beban mulai ditambahkan sudut stabil *pitch* akan mengalami penurunan. Motor ROV tidak dapat menyeimbangkan posisi ROV akibat beratnya beban pada manipulator. Semakin besar beban yang diberikan maka semakin besar pula perubahan sudut yang dihasilkan.



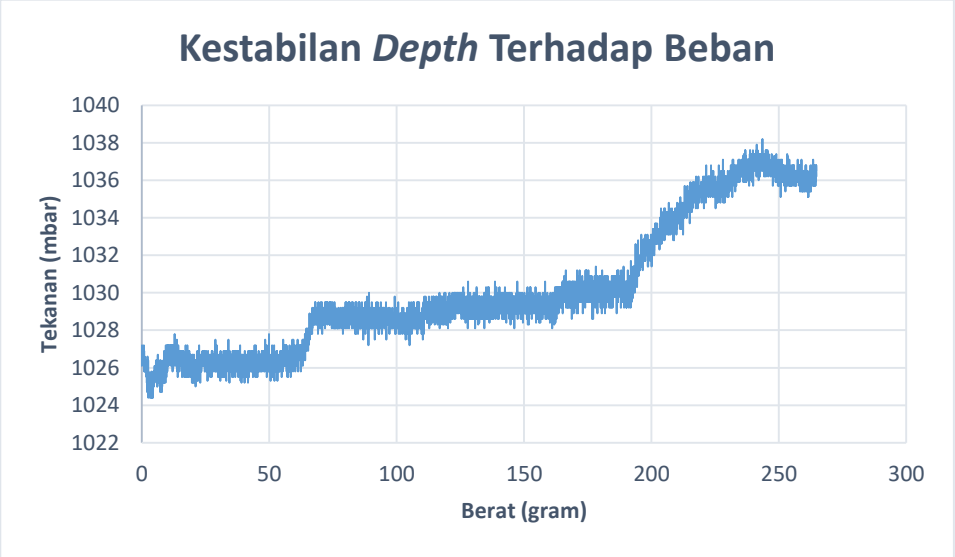
Gambar 4.18 Pengujian kesbilan *pitch* terhadap beban

Dapat terlihat pada Gambar 4.18 saat diberikan beban 200 gr sudut *pitch* berubah hingga 6 derajat hal ini dikeranakan motor 3 dan 4 pada bagian depan tidak mampu untuk mempertahankan posisi seimbang ROV. Hal ini juga mempengaruhi kemampuan ROV untuk bergerak ka atas. Semakin berat beban yang diterima maka semakin besar perubahan sudut *pitch.*

4.10 Pengujian Kestabilan *Depth* Terhadap Beban

Pengujian kestabilan *depth* terhadap beban merupakan pengujian untuk melihat kemampuan ROV mempertahankan kedalaman saat diberikan beban. Pengujian ini dilakukan dengan cara memberikan beban secara perlahan pada ROV. Setiap penambahan beban, nilai perubahan keluaran dari sensor tekanan akan disimpan. Nilai sensor yang disimpan mempresentasikan kedalaman ROV saat dilakukan pengujian.

Dari percobaan yang telah dilakukan, kedalaman ROV mengalami perubahan seiring pertambahan beban yang dilakukan. Saat ROV diberikan beban 50 gr, ROV masih dapat mempertahankan posisi *depth*. Kemudian saat beban ditambahkan secara perlahan kedalaman ROV akan berubah.



Gambar 4.19 Kestabilan *depth* terhadap beban

Dapat terlihat dari Gambar 4.19 saat beban yang diberikan sekitar 70 gr ROV mengalami perubahan *depth* secara perlahan. Saat ROV diberikan beban dengan berat 200 gr motor ROV tidak mampu untuk mempertahankan posisi sehingga ROV turun ke dasar secara perlahan. Dapat disimpulkan bahwa ROV hanya mampu mengangkat benda dengan berat sekitar 50 gr.

4.11 Pengaruh Efek Gerakan Manipulator Pada Kestabilan *Pitch*

Efek gerakan manipulator pada kestabilan *pitch* dapat terjadi akibat adanya gerakan manipulator pada posisi awal menuju posisi untuk mengambil objek. Pengaruh efek gerakan manipulator ini diukur dengan cara memposisikan ROV dalam posisi stabil dalam air kemudian objek yang akan diambil didekatkan secara perlahan ke arah ROV. Manipulator akan aktif ketika objek telah sesuai dengan posisi yang telah ditentukan. Saat manipulator bergerak maka akan dicatat nilai sudut *pitch* ROV hingga kembali pada kondisi normal dan stabil.

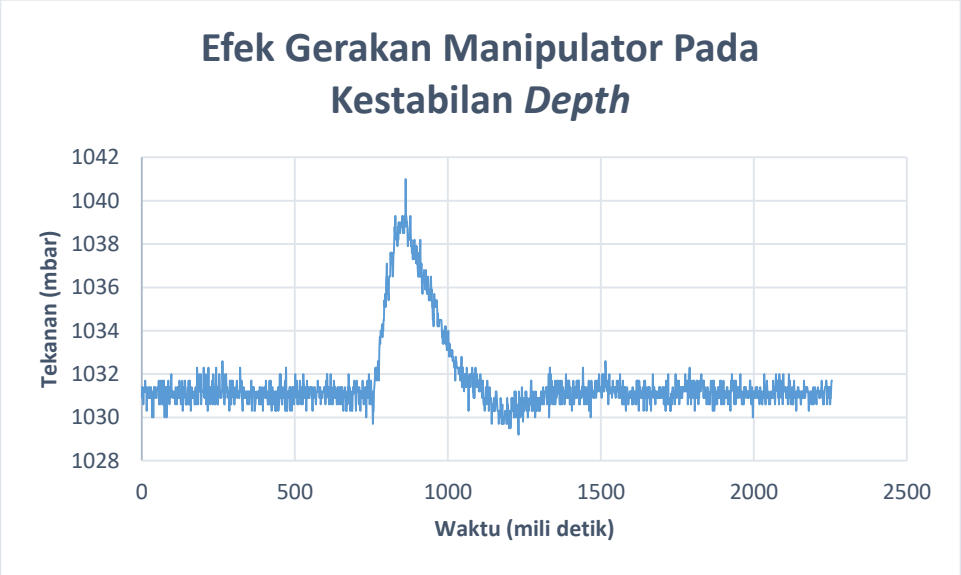
Dari Gambar 4.20 dapat terlihat bahwa sudut *pitch* akan mengalami perubahan sesaat setelah manipulator aktif untuk mengambil objek. Hal ini disebabkan gerakan manipulator dari posisi awal ke posisi akhir membuat ROV mengalami goncangan. Akibat terjadinya goncangan, dibutuhkan watu kurang lebih 0.5 detik untuk kembali ke posisi stabil.

Gambar 4.20 Efek gerakan manipulator pada kestabilan *pitch*

4.12 Pengaruh Efek Gerakan Manipulator Pada Kestabilan *Depth*

Efek gerakan manipulator pada kestabilan *depth* dapat terjadi akibat adanya pergerakan manipulator dari posisi awal ke posisi untuk mengambil objek. Sama seperti pada pengujian kestabilan *pitch*, pengaruh ini dapat terlihat pada *depth* saat ROV mendekati objek. Pengukuran ini dilakukan dengan cara memposisikan ROV dalam posisi stabil dalam air kemudian mendekatkan objek secara perlahan ke arah ROV. Saat posisi dan jarak objek telah sesuai maka secara otomatis manipulator akan aktif dan bergerak mengambil objek. Saat terjadi gerakan, akan dicatat nilai keluaran dari sensor tekanan yang mengindikasikan terjadinya perubahan kedalaman sesaat.

Dari Percobaan yang telah dilakukan dapat terlihat saat manipulator aktif, gerakan manipulator akan mengganggu kestabilan *depth* sesaat. Gerakan sudut pitch juga memperngaruhi kestabilan depth. Saat menipulator aktif dibutuhkan waktu sekitar 0.5 detik untuk kembali ke posisi stabil. Efek akibat pergerakan manipulator ini juga mempengaruhi objek yang terdeteksi. Saat penjejakan berlangsung membutuhkan waktu yang lebih lama untuk dapat mengambil objek.



Gambar 4.21 Efek gerakan manipulator pada kestabilan *depth*

4.13 Pengujian Parameter Kontrol PID

Pengujian parameter kontrol PID merupakan pengujian untuk menentukan nilai-nilai parameter Kp, Ki dan Kd. Pengujian ini bertujuan untuk mendapatkan nilai parameter terbaik yang dilihat dari respon pergerakan ROV. Pengujian ini dilakukan dengan cara mengubah nilai parameter Kp, Ki dan Kd secara perlahan kemudian nilai *overshoot, risetime, steady state error* dan *settling time* akan dicatat. Hasil dari pengujian kemudian dianalisis untuk mendapatkan nilai parameter terbaik.

Tabel 4.7 Respon konstanta PID pada sumbu x

Kp	Ki	Kd	Overshoot (%)	Rise Time (s)	Steady State Error (%)	Settling Time (s)
0.3	0.00E+00	0	9.09	8.9	23.7	22.8
0.3	1.00E-06	0	50.2	8.6	20.2	20.2
0.3	2.00E-06	0	50.6	5.9	22.6	18.9
0.3	3.00E-06	0	50.3	7	-8.3	16.3
0.3	4.00E-06	0	51.2	6.7	-9.7	14.7
0.3	5.00E-06	0	50.9	6.3	-6.3	15.2

Tabel 4.8 Respon konstanta PID pada sumbu y

Kp	Ki	Kd	Overshoot (%)	Rise Time (s)	Steady State Error (%)	Settling Time (s)
0.36	0	0	20.6	6.3	60.2	25.7
0.36	1.00E-06	0	40.4	5.2	40.5	24.4
0.36	2.00E-06	0	38.7	6.3	39.7	25.2
0.36	3.00E-06	0	39.2	5.5	40.2	23.7
0.36	4.00E-06	0	40.2	6	38.3	22.7
0.36	5.00E-06	0	37.6	5.97	38.15	22.33
0.36	6.00E-06	0	38.3	5.94	37.54	21.66
0.36	6.00E-06	0.1	36.3	4.91	36.93	20.99
0.36	6.00E-06	0.2	36.5	4.88	36.32	20.32
0.36	6.00E-06	0.3	36	4.85	35.71	19.65
0.36	6.00E-06	0.4	35.5	4.82	35.1	18.98

Dari hasil pengujian dapat terlihat bahwa nilai Kp mempengaruhi respon awal ROV untuk mencapai nilai *setpoint*. Nilai Kp yang terlalu kecil mengakibatkan motor tidak mampu untuk menggerakan ROV menuju nilai *setpoint*. Nilai Kp yang terlalu besar mengakibatkan respon motor terlalu cepat sehingga *overshoot* yang dihasilkan akan besar. Pada pengujian sumbu x nilai Kp yang didapatkan adalah sebesar 0.3 nilai tersebut merupakan nilai dengan respon *overshoot* terendah kemudian nilai Ki yang didapatkan adalah 0.000005. Karena respon yang dihasilkan sudah cukup baik maka tidak diperlukan nilai Kd. Oleh karena itu nilai Kd yang digunakan adalah 0.

Pada pengujian sumbu y nilai Kp yang didapatkan adalah sebesar 0.36. Nilai tersebut menghasilkan *overshoot* terendah namun dengan *risetime* yang cukup lambat. Setelah ditambahkan dengan Ki tidak ada pengaruh yang terlalu signifikan oleh karena itu ditetapkan nilai Ki sebesar 0.000006. *Overshoot* yang cukup besar membuat sistem membutuhkan waktu yang lama untuk stabil. Oleh karena itu maka ditambahkan dengan nilai Kd sebesar 0.4. Hasil penambahan Kd menghasilkan *overshoot* sebesar 35.5 %.

4.14 Pengujian Tingkat Keberhasilan Sistem Secara Keseluruhan

Pengujian tingkat keberhasilan sistem secara keseluruhan merupakan pengujian untuk melihat kemampuan ROV mendekati objek secara otomatis. Saat posisi ROV telah sesuai maka manipulator akan aktif untuk mengambil objek.

Pengujian dilakukan dengan cara meletakan objek dengan jarak tertentu dari ROV kemudian digunakan mode otomatis untuk mendekati objek. Pengujian dilakukan di aquarium dengan jarak maksimum 60 cm. Setiap percobaan akan dicatat keberhasilannya dalam mendekati dan mengambil objek.

Tabel 4.9 Pengujian tingkat keberhasilan

Percobaan	Jarak	mendekati objek	mengambil objek	Tingkat Keberhasilan mendekati objek (persen)	Tingkat Keberhasilan mengambil objek (persen)
1	60 cm	berhasil	gagal	100	40
2		berhasil	berhasil		
3		berhasil	berhasil		
4		berhasil	gagal		
5		berhasil	gagal		

Dari pengujian yang telah dilakukan, ROV dapat mendekati objek secara otomatis dan menggerakan manipulator untuk mengambil objek. Dari hasil 5 kali percobaan ROV hanya mampu 2 kali berhasil mengambil objek. Hal ini karena gerakan manipulator saat akan mengambil objek membuat ROV mengalami perubahan posisi. Sehingga objek yang berada didepan ROV menjadi bergeser. Saat objek bergeser dan terlalu dekat dengan ROV maka perlu dilakukan percobaan kembali.

Halaman ini sengaja dikosongkan

BAB 5
KESIMPULAN

5.1 Kesimpulan

Berdasarkan hasil pengujian pada sistem yang telah dikembangkan, dapat diketahui bahwa metode dapat mendeteksi objek dengan tingkat keberhasilan 96.8% pada jarak 60 cm. Maksimal jarak objek yang dapat dideteksi oleh metode ini adalah sejauh 2.7 m dengan percobaan yang dilakukan di kolam renang. Sistem membutuhkan waktu rata-rata 18 detik untuk dapat sejajar dengan objek. Pada percobaan yang telah dilakukan di kolam renang faktor lingkungan sangat mempengaruhi hasil yang didapatkan. Perubahan intensitas cahaya, kekeruhan air dan gelombang pada air sangat mempengaruhi tingkat akurasi sistem. ROV berhasil mengangkat beban namun tidak lebih dari 50 gr sebab motor ROV tidak dapat mempertahankan posisi kedalaman. Saat manipulator ROV diberi beban lebih dari 50 gr ROV tidak dapat mempertahankan posisi stabil *pitch* dan berpengaruh pada daya angkat ROV. Saat Manipulator aktif, ROV mengalami guncangan yang mempengaruhi stabilitas *pitch* dan *depth*. Dibutuhkan waktu kurang dari 0.5 detik untuk kembali ke posisi stabil. Percobaan secara keseluruhan ROV mampu mendekati objek dengan jarak 60 cm dengan keberhasil 100 %. Dari 5 kali percobaan yang dilakukan ROV hanya mampu mengambil objek sebanyak 2 kali dengan presentasi keberhasilan 40 %.

Halaman ini sengaja dikosongkan

DAFTAR PUSTAKA

[1] I. Jawhar, N. Mohamed, J. Al-Jaroodi dan S. Zhang, "An efficient framework for autonomous underwater vehicle extended sensor networks for pipeline monitoring," dalam *Robotic and Sensors Environments (ROSE)*, Washington DC, 2013.

[2] J. Franklin, "Subsea Pipeline Inspection, Repair and Maintenance," THEON, 2017. [Online]. Available: http://www.theonltd.com/uncategorized/subsea-pipeline-inspection-repair-and-maintenance/. [Diakses 28 06 2018].

[3] R. L. Wemli, "Operational (efectiveness of unmanned," dalam *ROV Committee of Marine Technology Society*, Wernli, 1998.

[4] B.H.Jun, H.W.Shim, P.M.Lee dan H.back, "Workspace Control System of Underwater Tele-Operated Manipulator on ROVs," dalam *Ministry of Land, Transportation and Marine Affairs (MLTM)*, Daejeon Korea, 2009.

[5] M. Hildebrandt, L. Christensen, J. Kerdels, J. Albiez dan F. Kirchner, "Realtime Motion Compensation for ROV-based," dalam *IEEE*, Bremen, 2009.

[6] J. Rife, "Automated robotic tracking of gelatinous animals," dalam *IEEE*, California, 2003.

[7] J. Zhou dan C. Clark, "Autonomous fish tracking by ROV using Monocular camera," dalam *Canadian Conference*, 2006.

[8] A. Khan, S. S. A. Ali dan A. S. Malik, "Control of autonomous underwater vehicle based on visual feedback for pipeline inspection," dalam *Robotics and Manufacturing Automation (ROMA), 2016 2nd IEEE International Symposium on*, Ipoh, Malaysia, 2016.

[9] D. Ayala dan D. Chávez, "Low cost embedded vision system for location and tracking of a color object," dalam *3DTV Conference: The True Vision - Capture, Transmission and Display of 3D Video (3DTV-CON)*, Copenhagen, Denmark, 2018.

[10] N. Li, D. Zhang, X. Gu, L. Huang, W. Liu dan T. Xu, "An Improved Mean Shift Algorithm for Moving Object Tracking," dalam *Canadian Conference on Electrical and Computer Engineering*, Halifax, Canada, 2015.

[11] S. Yim, C. M. Clark dan T. Peters, "ROV-based Tracking of a Shallow Water Nocturnal," dalam *Oceans - San Diego*, San Diego, CA, USA, 2013.

[12] D. Walther, D. R. Edgington dan C. Koch, “Detection and Tracking of Objects in Underwater Video,” dalam *Computer Vision and Pattern Recognition (CVPR’04)*, California, 2004.

[13] E. group, “ECA group ARM 7E electrical manipulator data sheet,” ECA group, 2016. [Online]. Available: : http://eca-media.ecagroup.com/player/pdf?key=c9faef88b70ef5dade8%. [Diakses 2018 juni 29].

[14] M. Siven, “Vedenalaisen Hydraulisen Puomin Rannemekanismin Suunnittelu,” Tampere University Of Technology, Finland, 2015.

[15] A. MÄENPÄÄ, “DESIGN OF SIX DEGREES OF FREEDOM HYDRAULIC,” Master of Science Thesis, TAMPERE UNIVERSITY OF TECHNOLOGY, Tampere, 2016.

[16] M. J. Poretti, “DESIGN OF A ROBOTIC ARM MANIPULATOR CAMERA UNIT FOR MINI,” Faculty of California Polytechnic State University, San Luis Obispo, 2013.

[17] A. A. Yusof, F. Wasbari dan M. Q. Ibrahim, “Research Development of Energy Efficient Water Hyraulics Manipulator for Underwater Application,” *Trans Tech Publications*, vol. 393, no. Applied Mechanics and Materials, pp. 723-728, 2013.

[18] J. Zhang, W. Li, J. Yu, X. Feng, Q. Zhang dan G. Chen, “Study of manipulator operations maneuvered by a ROV in virtual,” *Elsevier*, vol. 142, no. Ocean Engineering, pp. 292-302, 2017.

[19] T. Salgado-Jimenez, J. L. Gonzalez-Lopez, L. F. Martinez-Soto, E. Olguin-Lopez, P. A. Resendiz-Gonzalez dan M. Bandala-Sanchez, “Deep water ROV design for the Mexican oil industry,” dalam *Center for Engineering and Industrial Development (CIDESI)*, Mexico, 2010.

[20] Z. Md, Zain, M. M. Noh, K. A. A. Rahim dan N. Harun, “Design and Development of an X4-ROV,” dalam *h International Conference on Underwater System Technology: Theory & Applications*, Pahang, Malaysia, 2016.

[21] J. A. Ramırez, R. E. Vasquez, L. B. Gutierrez dan D. A. Florez, “MECHANICAL/NAVAL DESIGN OF AN UNDERWATER REMOTELY OPERATED VEHICLE (ROV) FOR SURVEILLANCE AND INSPECTION OF PORT FACILITIES,” dalam *ASME International Mechanical Engineering Congress and Exposition* , Seattle USA, 2007.

[22] M. e. c. i. t. d. o. d. ROV, “Digital Pendulum: Control in a Matlab Environment,” dalam *Underwater Technology,* Bussan South Korea, 2017.

[23] H. Shim, B.-H. Jun dan P.-M. Lee, “Dynamic workspace Control Methode of Underwater Based on Compensation of an ROV,” dalam *maritim and ocean engineering,* daejeon, 2011.

[24] N. GROUP, “THE ONLINE BOATING AND MARITIME EXHIBITION,” NAUTIC, [Online]. Available: http://www.nauticexpo.com/prod/eca-group/product-25365-119924.html. [Diakses 27 5 2018].

[25] R. L. Wernli, “ROV CATEGORIES,” Marine Technology Society, 21 februari 2017. [Online]. Available: http://www.rov.org/rov_categories.cfm. [Diakses 29 juni 2018].

[26] A. N. Fitriana, K. Mutijarsa dan W. Adiprawita, “Color-Based Segmentation and Feature Detection for Ball and Goal Post on Mobile Soccer Robot Game Field,” dalam *International Conference on Information Technology Systems and Innovation (ICITSI)* , Bali, 2016.

[27] S. Milan, H. Vaclav dan B. Roger, Image Processing, Analisys and machine Vision, USA: Thomson Learning, 2008.

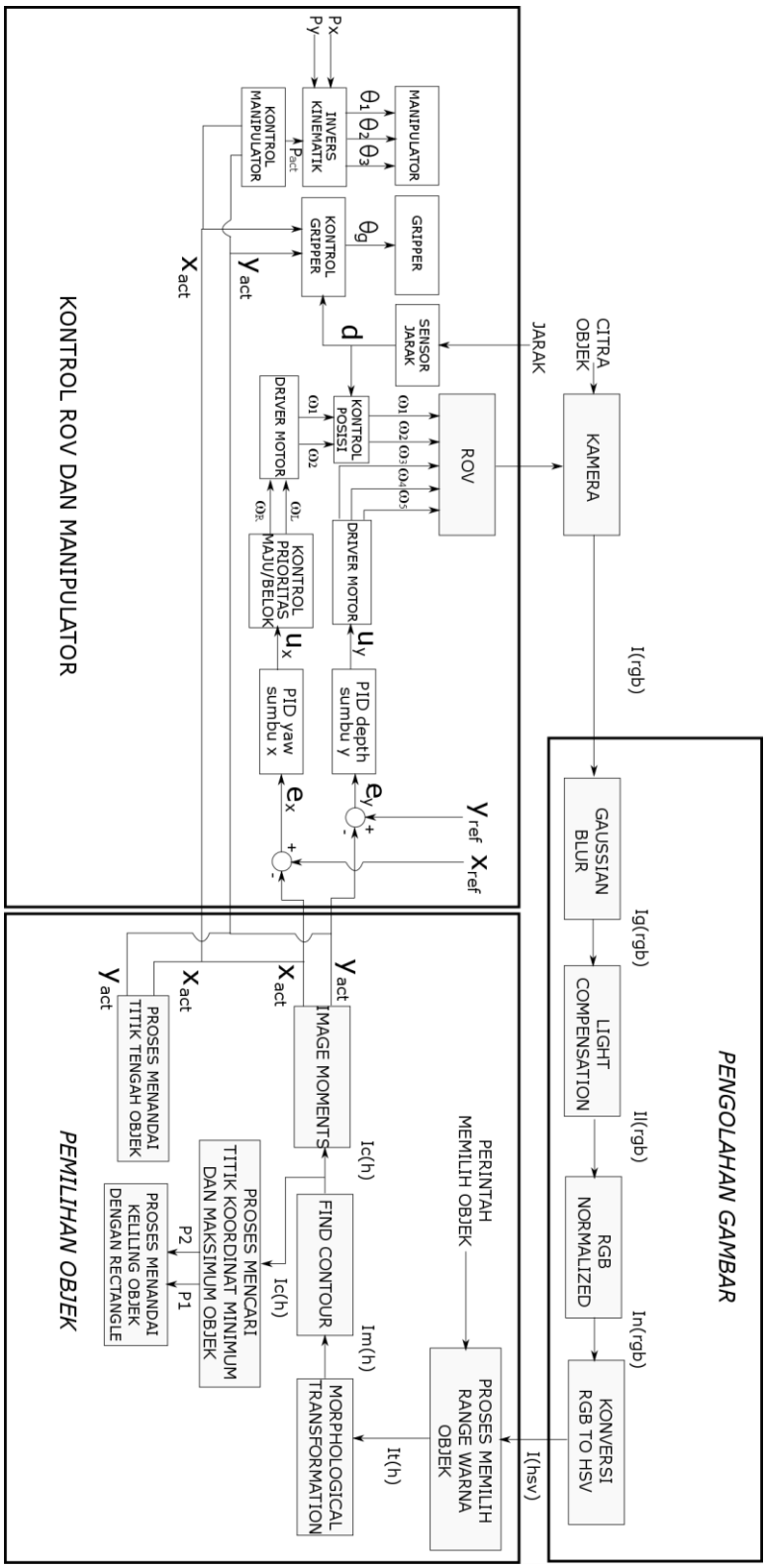
[28] Y. Çelik, M. Altun dan M. Güneş, “Color Based Moving Object Tracking with An Active Camera Using Motion Information,” dalam *International Artificial Intelligence and Data Processing Symposium (IDAP),* Malatya, Turkey, 2017.

[29] N. D. P. A. K. S. Joseph K. P. Tsoi, “Real-time object tracking based on colour feature and perspective projection,” dalam *Ninth International Conference on Sensing Technology,* Auckland, New Zealand, 2015.

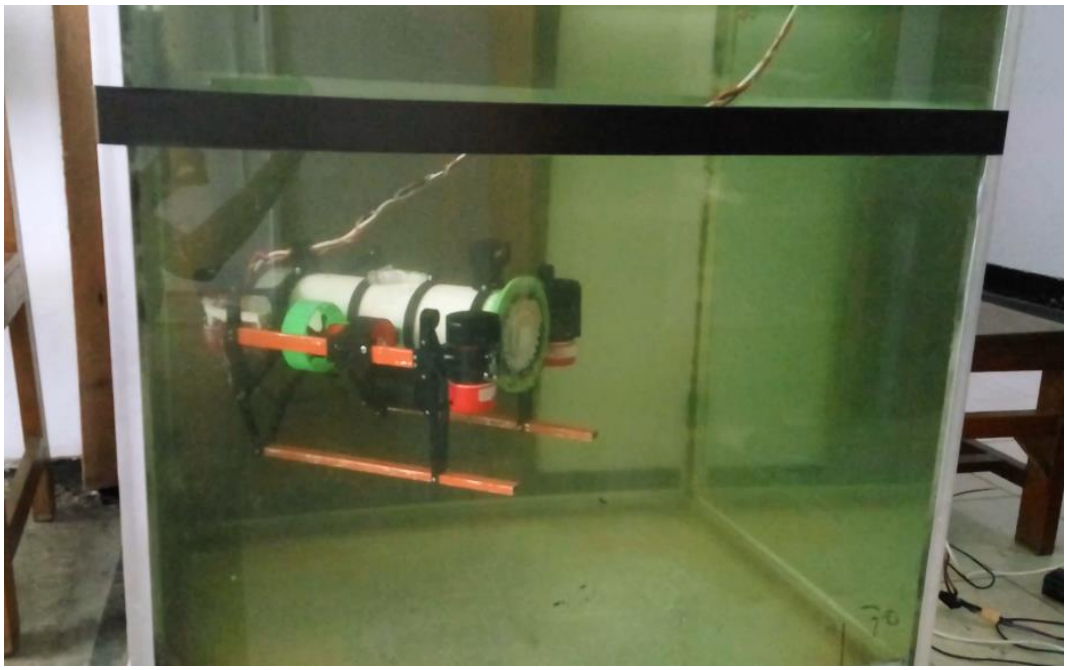
[30] H.-C. L. a. P.-C. C. Guo-Shing Huang, “Robotic Arm Grasping and Placing Using Edge Visual Detection System,” dalam *Asian Control Conference (ASCC),* Taiwan, 2011.

Halaman ini sengaja dikosongkan

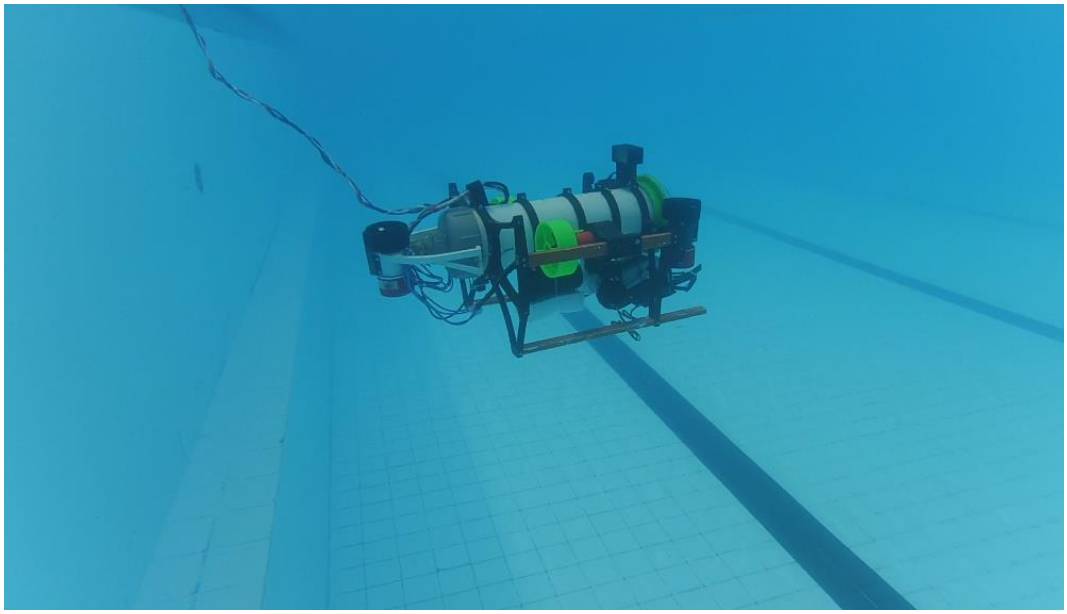
LAMPIRAN


PENGOLAHAN GAMBAR
CITRA OBJEK
KAMERA
I(rgb)
GAUSSIAN BLUR
Ig(rgb)
LIGHT COMPENSATION
Il(rgb)
RGB NORMALIZED
In(rgb)
KONVERSI RGB TO HSV
I(hsv)
JARAK
KONTROL ROV DAN MANIPULATOR
ROV
SENSOR JARAK
GRIPPER
MANIPULATOR
KONTROL GRIPPER
KONTROL MANIPULATOR
INVERS KINEMATIK
KONTROL POSISI
DRIVER MOTOR
DRIVER MOTOR
KONTROL PRIORITAS MAJU/BELOK
PID yaw sumbu x
PID depth sumbu y
PEMILIHAN OBJEK
PERINTAH MEMILIH OBJEK
PROSES MEMILIH RANGE WARNA OBJEK
MORPHOLOGICAL TRANSFORMATION
FIND CONTOUR
IMAGE MOMENTS
PROSES MENCARI TITIK KOORDINAT MINIMUM DAN MAKSIMUM OBJEK
PROSES MENANDAI KELILING OBJEK DENGAN RECTANGLE
PROSES MENANDAI TITIK TENGAH OBJEK

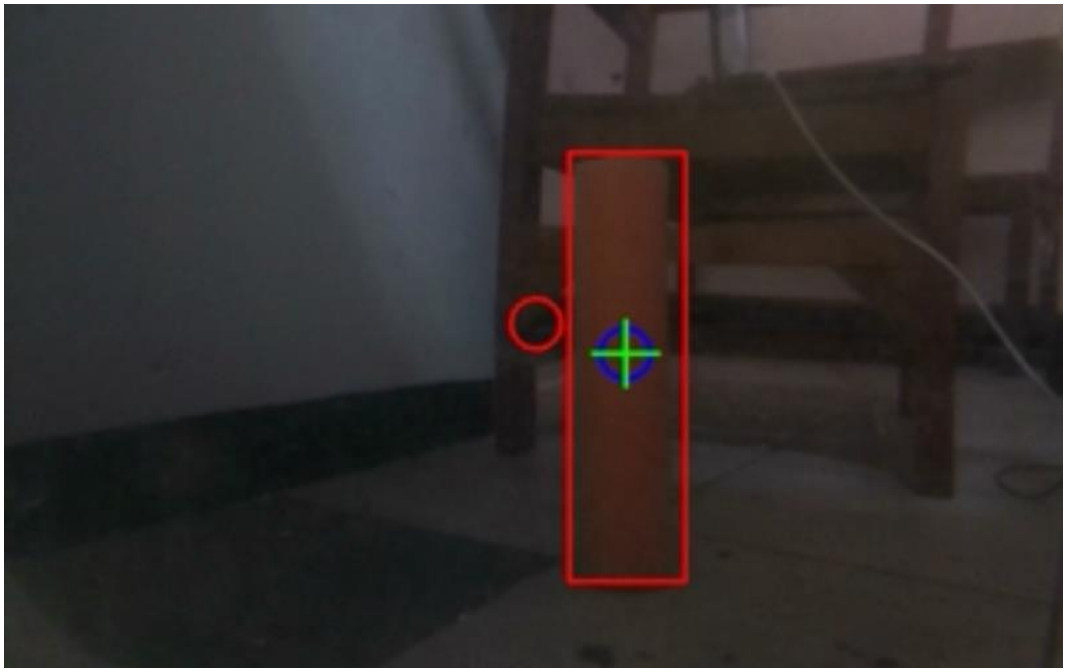
A. Pengujian di Aquarium lab A206



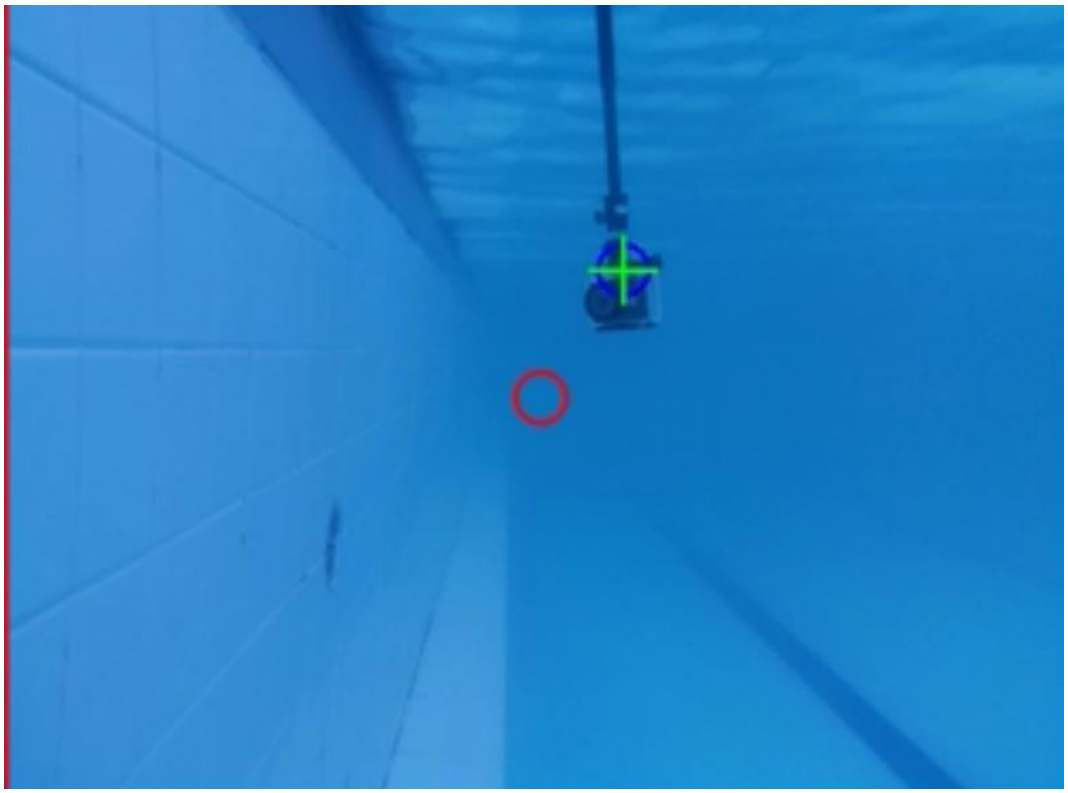
B. Pengujian di Kolam Renang STTAL



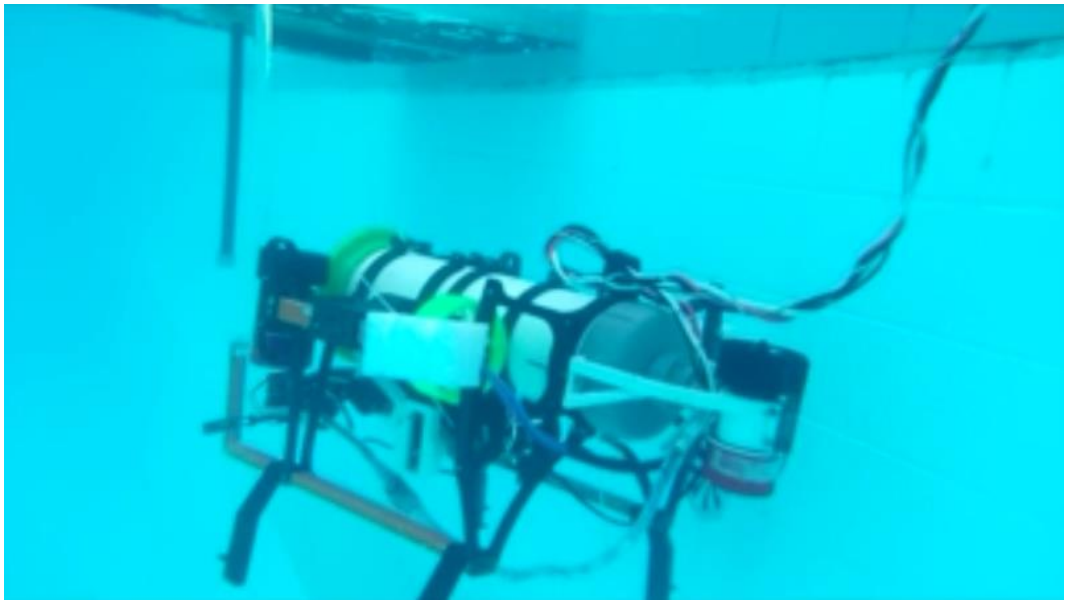
C. Pengujian Deteksi Objek di Aquarium Lab A206



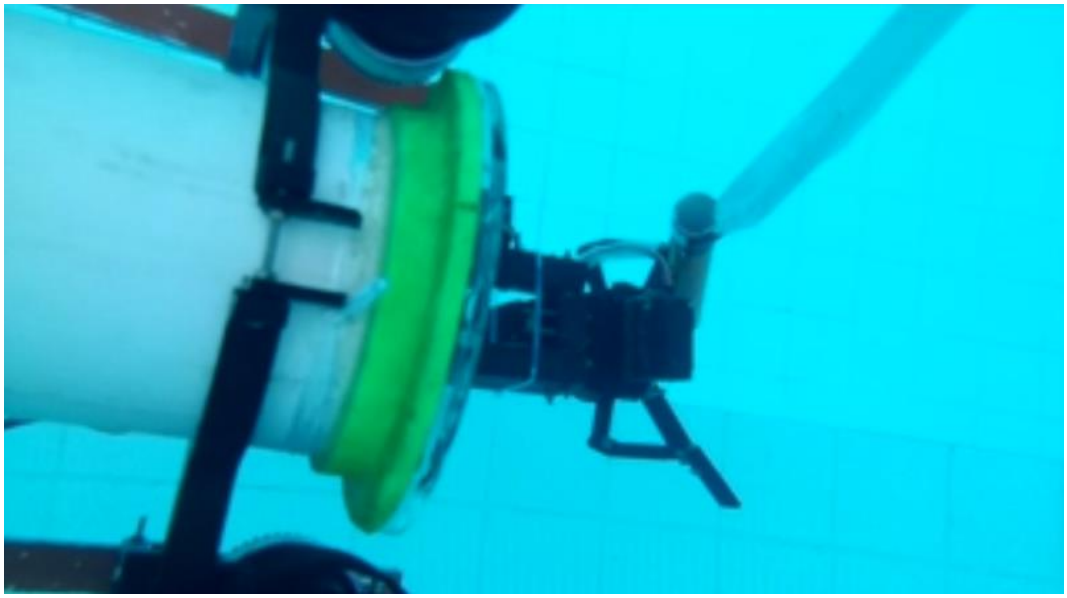
D. PENGUJIAN Deteksi Objek di Kolam Renang STTAL



E. Pengujian Tracking Objek di Kolam Renang STTAL



F. Pengujian Mengambil Objek di Kolam Renang STTAL



Code Program Metode Tracking (Qt C++)

```
#include <iostream>
#include <stdio.h>
#include <string>
#include <sstream>
#include <fstream>
#include <vector>
#include <ctype.h>
#include <opencv2/imgproc/imgproc.hpp>
#include <opencv2/videoio/videoio.hpp>
#include <opencv2/highgui/highgui.hpp>
#include <opencv2/video/tracking.hpp>
#include <math.h>
#include <cv.h>
#include <highgui.h>
using namespace cv;
using namespace std;

 Mat cameraFeed,cameraFeed1;
 VideoCapture capture;

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
Point2f point;
bool addRemovePt = false;
int H_MIN = 0;
int H_MAX = 256;
int S_MIN = 0;
int S_MAX = 256;
int V_MIN = 0;
int V_MAX = 256;
int X_lukas;
int Y_lukas;
int X_color;
int Y_color;
int x = 0, y = 0;
int x_fix, y_fix;
int x_move;
int y_move;
int result;
int counting(int result);
int x1,yy1,x2,y2,x3,y3;
//Mat cameraFeed;
double AVERAGE_H_MIN=0;
double AVERAGE_H_MAX=256;
int AVERAGE_S_MIN=0;
int AVERAGE_S_MAX=256;
```

```
double AVERAGE_V;

int r1 = 70;
int s1 = 0;
int r2 = 140;
int s2 = 255;

const int FRAME_WIDTH = 640;
const int FRAME_HEIGHT = 480;
const int MAX_NUM_OBJECTS = 50;
const int MIN_OBJECT_AREA = 60 * 60;
const int MAX_OBJECT_AREA = FRAME_HEIGHT*FRAME_WIDTH / 1.5;
const string windowName = "Original Image";
const string windowName1 = "HSV Image";
const string windowName2 = "Thresholded Image";
const string windowName3 = "After Morphological Operations";
const string windowName4 = "contrast";
const string trackbarWindowName = "Trackbars";

bool calibrationMode;
double area;
bool mouseIsDragging;
bool mouseMove;
bool rectangleSelected;
cv::Point initialClickPoint, currentMousePoint;
cv::Rect rectangleROI; //this is the ROI that the user has selected
vector<int> H_ROI, S_ROI, V_ROI, R_ROI,G_ROI,B_ROI;
double avg2(std::vector<int> const& v);

void createTrackbars(){
namedWindow(trackbarWindowName, 0);
char TrackbarName[50];

   createTrackbar("H_MIN", trackbarWindowName, &H_MIN, 255,
on_trackbar);
   createTrackbar("H_MAX", trackbarWindowName, &H_MAX, 255,
on_trackbar);
   createTrackbar("S_MIN", trackbarWindowName, &AVERAGE_S_MIN, 255,
on_trackbar);
   createTrackbar("S_MAX", trackbarWindowName, &AVERAGE_S_MAX,
255, on_trackbar);
}

void clickAndDrag_Rectangle(int event, int x, int y, int flags, void* param){
   if (calibrationMode == true){
      Mat* videoFeed = (Mat*)param;
```

```
        if (event == CV_EVENT_LBUTTONDOWN && mouseIsDragging ==
false)
        {
           initialClickPoint = cv::Point(x, y);
           point = Point2f((float)x, (float)y);
           addRemovePt = true;
           mouseIsDragging = true;
        }
        if (event == CV_EVENT_MOUSEMOVE && mouseIsDragging == true)
        {
           currentMousePoint = cv::Point(x, y);
           mouseMove = true;
        }
         if (event == CV_EVENT_LBUTTONUP && mouseIsDragging == true)
        {
           rectangleROI = Rect(initialClickPoint, currentMousePoint);
           mouseIsDragging = false;
           mouseMove = false;
           rectangleSelected = true;
        }

        if (event == CV_EVENT_RBUTTONDOWN){
           H_MIN = 0;
           S_MIN = 0;
           V_MIN = 0;
           H_MAX = 255;
           S_MAX = 255;
           V_MAX = 255;
           AVERAGE_H_MIN=0;
           AVERAGE_H_MAX=256;
           AVERAGE_S_MIN=0;
           AVERAGE_S_MAX=256;
        }
     }
}

void recordHSV_Values(cv::Mat frame, cv::Mat hsv_frame){
    if (mouseMove == false && rectangleSelected == true){

        if (H_ROI.size()>0) H_ROI.clear();
        if (S_ROI.size()>0) S_ROI.clear();
        if (V_ROI.size()>0 )V_ROI.clear();

        if (rectangleROI.width<1 || rectangleROI.height<1) cout << "Please drag a
rectangle, not a line" << endl;
        else{
           for (int i = rectangleROI.x; i<rectangleROI.x + rectangleROI.width; i++){
```

```
            for (int j = rectangleROI.y; j<rectangleROI.y + rectangleROI.height;
j++){
                H_ROI.push_back((int)hsv_frame.at<cv::Vec3b>(j, i)[0]);
                S_ROI.push_back((int)hsv_frame.at<cv::Vec3b>(j, i)[1]);
                V_ROI.push_back((int)hsv_frame.at<cv::Vec3b>(j, i)[2]);
            }
         }
      }
      rectangleSelected = false;

      if (H_ROI.size()>0){
         H_MIN = *std::min_element(H_ROI.begin(), H_ROI.end());
         H_MAX = *std::max_element(H_ROI.begin(), H_ROI.end());

         AVERAGE_H_MAX = avg2(H_ROI)+10;
         AVERAGE_H_MIN = avg2(H_ROI)-10;
         cout << "MINn 'H' VALUE: " << AVERAGE_H_MIN << endl;
         cout << "MAXx 'H' VALUE: " << AVERAGE_H_MAX << endl;
         cout << "MIN 'H' VALUE: " << H_MIN << endl;
         cout << "MAX 'H' VALUE: " << H_MAX << endl;

      }
      if (S_ROI.size()>0){
         S_MIN = *std::min_element(S_ROI.begin(), S_ROI.end());
         S_MAX = *std::max_element(S_ROI.begin(), S_ROI.end());
         AVERAGE_S_MAX = avg2(S_ROI)+10;
         AVERAGE_S_MIN = avg2(S_ROI)-10;
      }
      if (V_ROI.size()>0){
         V_MIN = *std::min_element(V_ROI.begin(), V_ROI.end());
         V_MAX = *std::max_element(V_ROI.begin(), V_ROI.end());
      }
   }
}

double avg2(std::vector<int> const& v) {
   int sum = 0;
     for ( int x : v ) sum += x;
     return v.empty() ? 0 : sum / v.size();
}

string intToString(int number){
   std::stringstream ss;
   ss << number;
   return ss.str();
}
```

```
void drawObject(int x, int y, Mat &frame){

   circle(frame, Point(x, y), 15, Scalar(255, 0, 0), 2);
   if (y - 25>0)
      line(frame, Point(x, y), Point(x, y - 20), Scalar(0, 255, 0), 2);
   else line(frame, Point(x, y), Point(x, 0), Scalar(0, 255, 0), 2);
   if (y + 25<FRAME_HEIGHT)
      line(frame, Point(x, y), Point(x, y + 20), Scalar(0, 255, 0), 2);
   else line(frame, Point(x, y), Point(x, FRAME_HEIGHT), Scalar(0, 255, 0), 2);
   if (x - 25>0)
      line(frame, Point(x, y), Point(x - 20, y), Scalar(0, 255, 0), 2);
   else line(frame, Point(x, y), Point(0, y), Scalar(0, 255, 0), 2);
   if (x + 25<FRAME_WIDTH)
      line(frame, Point(x, y), Point(x + 20, y), Scalar(0, 255, 0), 2);
   else line(frame, Point(x, y), Point(FRAME_WIDTH, y), Scalar(0, 255, 0), 2);
}

void morphOps(Mat &thresh){
   Mat erodeElement = getStructuringElement(MORPH_RECT, Size(1, 1));
   Mat dilateElement = getStructuringElement(MORPH_RECT, Size(5, 5));

   erode(thresh, thresh, erodeElement);
   erode(thresh, thresh, erodeElement);

   dilate(thresh, thresh, dilateElement);
   dilate(thresh, thresh, dilateElement);
}

void trackFilteredObject(int &x, int &y, int X_lukas, int Y_lukas, Mat threshold,
Mat &cameraFeed, Mat HSV){

   Mat temp;
   threshold.copyTo(temp);
   vector< vector<Point> > contours;
   vector<Vec4i> hierarchy;

   findContours(temp, contours, hierarchy, CV_RETR_CCOMP,
CV_CHAIN_APPROX_SIMPLE);
         vector<Rect> boundRect(contours.size());
         vector<vector<Point>> contours_poly(contours.size());

         for (int i=0; i<contours.size(); i++)
         {
            cv::approxPolyDP(Mat(contours[i]),contours_poly[i],3,true);
            boundRect[i]=boundingRect(Mat(contours_poly[i]));
         }
         int max_index=0, max_area=0;
```

```
            for (int i=0; i<boundRect.size(); i++)
            {
               int a = boundRect[i].area();
               if(a>max_area)
               {
                  max_area=a;      // LUAS AREA OBJEK
                  max_index=i;
               }
            }

            if (boundRect.size() > 0 && max_area > 1000)      //jumlah objek
            {
               for (int i=0; i<boundRect.size(); i++){

cv::rectangle(cameraFeed,boundRect[max_index].tl(),boundRect[max_index].br()
,Scalar(0,0,255),2,8,0);
cout << " titik" << boundRect[max_index].tl() << " pxl" <<
boundRect[max_index].br() <<endl;
               }

            }
                  }
   double refArea = 0;
   int largestIndex = 0;
   bool objectFound = false;
   if (hierarchy.size() > 0) {
      int numObjects = hierarchy.size();
      if (numObjects<MAX_NUM_OBJECTS){
         for (int index = 0; index >= 0; index = hierarchy[index][0]) {

            Moments moment = moments((cv::Mat)contours[index]);
            cv::RotatedRect rotatedRect = cv::minAreaRect(contours[index]);
            area = moment.m00;

            if (area>MIN_OBJECT_AREA && area<MAX_OBJECT_AREA &&
area>refArea){
               x = moment.m10 / area;
               y = moment.m01 / area;
               objectFound = true;
               refArea = area;
               largestIndex = index;
            }

             if (area<MIN_OBJECT_AREA ){
               recordHSV_Values(cameraFeed, HSV);
             }
```

```
            }
            if (objectFound == true){
              drawObject(x, y, cameraFeed);
            }
        }
        else {
            putText(cameraFeed, "TERLALU DEKAT COBA FILTER KEMBALI",
Point(0, 50), 1, 2, Scalar(0, 0, 255), 2);
        }
    }
}

int main(int argc, char *argv[]){

        bool trackObjects = true;
        bool useMorphOps = true;
        calibrationMode = true;
        Mat cameraFeed,cameraFeed1;
        Mat HSV;
        Mat threshold,threholdd;
        vector<Mat> channels, channelshsv,HH, SS;

//-----------------------------------COUNTING FRAME---------------------------------//
        int fileKe=0;
        char namaFile[33];
        int hitung=0;
        ofstream myfile;
        sprintf(namaFile,"lalala%d.txt",fileKe);
        myfile.open(namaFile);

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
        VideoCapture capture;
        capture.open(1);
        capture.set(CV_CAP_PROP_FRAME_WIDTH, FRAME_WIDTH);
        capture.set(CV_CAP_PROP_FRAME_HEIGHT, FRAME_HEIGHT);

        cv::namedWindow(windowName);
        cv::setMouseCallback(windowName, clickAndDrag_Rectangle,
&cameraFeed1);
        mouseIsDragging = false;
        mouseMove = false;
        rectangleSelected = false;

//--------------------------------- VIDEO CAPTURE ----------------------------------//
        int frame_width=   capture.get(CV_CAP_PROP_FRAME_WIDTH);
        int frame_height=   capture.get(CV_CAP_PROP_FRAME_HEIGHT);
```

```
	VideoWriter video("out.avi",CV_FOURCC('M','J','P','G'),10,
Size(frame_width,frame_height),true);

//-----------------------------------------------------------------------------------------//
	while (1){
		capture.read(cameraFeed);
		capture >> cameraFeed;
		cameraFeed.copyTo((cameraFeed1));
		GaussianBlur(cameraFeed,cameraFeed,Size(7,7),3.0,3.0);

		split(cameraFeed,channels);

		channels[0]/=channels.size();
		channels[1]/=channels.size();
		channels[2]/=channels.size();

		Mat total = channels[0] + channels[1] + channels[2];

		channels[0] = (channels[0]*255)/total;
		channels[1] = (channels[1]*255)/total;
		channels[2] = (channels[2]*255)/total;

		double min, max;
		minMaxIdx(total, &min, &max);

		merge(channels,cameraFeed);

		imshow("original2", cameraFeed);

		cvtColor(cameraFeed, HSV, COLOR_BGR2HSV_FULL);

		split(HSV,channelshsv);

		Mat totalhsv = channelshsv[0] + channelshsv[1] + channelshsv[2];
		channelshsv[0] = (channelshsv[0]*255)/totalhsv;

		merge(channelshsv,cameraFeed);

		recordHSV_Values(cameraFeed, HSV);

		inRange(HSV, Scalar(H_MIN, AVERAGE_S_MIN, V_MIN),
Scalar(H_MAX, AVERAGE_S_MAX, V_MAX), threshold);

		if (useMorphOps)
		   morphOps(threshold);
		if (trackObjects)
```

```
                trackFilteredObject(x, y, X_lukas, Y_lukas, threshold,
cameraFeed1,HSV);

                int error_x = x - 320;
                int error_y = y - 240;
                circle(cameraFeed1, Point(320, 240), 10, Scalar(255, 0, 0), 2);

                myfile<<hitung;myfile<<" "<<area<<" "<<x<<" "<<y<<"
"<<error_x<<" "<<error_y<<'\n';
                    hitung++;
//////////////////////////////////////////////////////////////////////////////////////////////////////

        if (calibrationMode == true){
        }
        else{
           destroyWindow(windowName2);
           destroyWindow(trackbarWindowName);
        }
createTrackbars();
        imshow("HSV", HSV);
        imshow("threshold", threshold);
        imshow(windowName, cameraFeed1);
        video.write(cameraFeed1);
        if (waitKey(10) == 99) calibrationMode = !calibrationMode;//if user
presses 'c', toggle calibration mode
        if( code == 27 || code == 'q' || code == 'Q' )
           break;
        char c = (char)waitKey(10);
        if( c == 27 )
           break;
        switch( c )
        {
        case 'q':
           imwrite("1original.jpg",cameraFeed);
           imwrite("1HSV.jpg",HSV);
           imwrite("1threshold.jpg",threshold);
           imwrite("1output.jpg",cameraFeed1);
           myfile.close();
           cout<<"selesai"<<endl;
           break;
        case 'r':
           needToInit = true;
           break;
        case 's':
           (char)waitKey(0);
           break;
        case 'c':
```

```
                points[0].clear();
                points[1].clear();
                break;
            }
            std::swap(points[1], points[0]);
            cv::swap(prevGray, gray);
        }
    fileKe++;
     return 0;
}
```

Code Progam Arduino

```
#include <MS5803_14.h>
#include <DistanceGP2Y0A41SK.h>
#include <medianFilter.h>
#include <Wire.h>
#include <EEPROM.h>
#include <Servo.h>

DistanceGP2Y0A41SK Dist;
medianFilter Filter;
////////////////////////////////-- SERVO --/////////////////////////////////////
float ch1ser_scale;
float ch2ser_scale;
float a1 = 10;
float a2 = 8.5;
float x1;
float y1;
float x_invers,y_invers;
float  theta2;
float  theta1;
float  theta3;
float  theta2new;
float  theta1new;
float  theta3new;
int pos = 0;
int catchh;
// Declare the Servo pin
int servoPin1 = 12;
int servoPin2 = 11;
int servoPin3 = 10;
int servoPin4 = 13;//9
int servoPin5 = 9;//13
// Create a servo object
Servo Servo1;
Servo Servo2;
Servo Servo3;
Servo Servo4;
Servo Servo5;

////////////////////////////////-- PPM --/////////////////////////////////////
volatile int channel_0,channel_1, channel_2, channel_3, channel_4, channel_5,
channel_6, channel_7, channel_8, channel_9, channel_10,
channel_1_ser,channel_2_ser;
int pulse = 0;
const byte interruptPin = 2;
unsigned long t[11];
```

```
////////////////////////////////--SENSOR SHARP DISTANCE--////////////////////////////////
const int numreadings = 10;
int readings[numreadings];
int readindex = 0;
int total = 0;
int average = 0;
int sensorpin = 0;
float val;
float realval;
////////////////////////////////////////////////////////////////////////////////////////////////////////////
boolean data;
boolean track;
boolean distance;
String datain;
String dt[10];
int ii;
int x,y;
////////////////////////////////-- MPU6050 && MS5803--////////////////////////////////////
#define gyro_address 0x68
MS_5803 sensor = MS_5803(512);

int cal_int, start, i;
int temperature;
int acc_axis[4], gyro_axis[4];
float depth_input;
float gyro_roll_input, gyro_pitch_input, gyro_yaw_input;
float roll_level_adjust, pitch_level_adjust, yaw_level_adjust;
float angle_roll_acc, angle_pitch_acc, angle_yaw_acc, angle_pitch, angle_roll,
angle_yaw;
float angle_roll_used, angle_pitch_used, angle_yaw_used;

float
pid_error_temp,pid_error_temp_x,pid_error_temp_y,pid_error_temp_distance;
float pid_i_mem_roll, pid_roll_setpoint, pid_output_roll, pid_last_roll_d_error;
float pid_i_mem_pitch, pid_pitch_setpoint, pid_output_pitch,
pid_last_pitch_d_error;
float pid_i_mem_yaw, pid_yaw_setpoint, pid_output_yaw, pid_last_yaw_d_error;
float pid_i_mem_depth, pid_depth_setpoint = 1010, pid_output_depth,
pid_last_depth_d_error;

float pid_i_mem_x, pid_x_setpoint = 320, pid_output_x, pid_last_x_d_error;
float pid_i_mem_y, pid_y_setpoint = 240, pid_output_y, pid_last_y_d_error;

float pid_i_mem_distance, pid_distance_setpoint = 25, pid_output_distance,
pid_last_distance_d_error;

double gyro_pitch, gyro_roll, gyro_yaw;
```

```
double gyro_axis_cal[4];
long acc_x, acc_y, acc_z, acc_total_vector;
unsigned long times;
bool cal_is_done=0;
boolean gyro_angles_set;

void mpu6050_register();
void motor_stop();
void mpu6050_signal();
void inverskinematic(float a1,float a2,float x1,float y1);

float pid_p_gain_roll = 1;              //Gain setting for the roll P-controller
float pid_i_gain_roll = 0.003;          //Gain setting for the roll I-controller
float pid_d_gain_roll = 0.0;            //Gain setting for the roll D-controller
int   pid_max_roll    = 500;            //Maximum output of the PID-controller

float pid_p_gain_pitch = 1.5;  //Gain setting for the pitch P-controller.
float pid_i_gain_pitch = 0.002;  //Gain setting for the pitch I-controller.
float pid_d_gain_pitch = 0;  //Gain setting for the pitch D-controller.
int   pid_max_pitch    = 500;        //Maximum output of the PID-controller (+/-)

float pid_p_gain_yaw = 0;               //Gain setting for the pitch P-controller.
float pid_i_gain_yaw = 0;               //Gain setting for the pitch I-controller.
float pid_d_gain_yaw = 0.0;             //Gain setting for the pitch D-controller.
int   pid_max_yaw    = 0;               //Maximum output of the PID-controller

float pid_p_gain_depth = 11;
float pid_i_gain_depth = 0.05;
float pid_d_gain_depth = 0.3;
int   pid_max_depth    = 800;

float pid_p_gain_x = 0.35;
float pid_i_gain_x = 0.0000000005;
float pid_d_gain_x = 0.7;
int   pid_max_x    = 800;

float pid_p_gain_y = 0.5;
float pid_i_gain_y = 0.0001;
float pid_d_gain_y = 0;
int   pid_max_y    = 800;

float pid_p_gain_distance = 0.7;
float pid_i_gain_distance = 0.00005;
float pid_d_gain_distance = 2;
int   pid_max_distance    = 800;
//////////////////////////////////////////////////////////////////////////////////
```

```
////////////////////////////////-- CONTROL MOTOR --/////////////////////////////////////
int pwm1, pwm2, pwm3, pwm4, pwm5;
int m[5];
int m1a, m2a, m3a, m4a, m5a;
int P1,P2,P3,P4,P5;
int inA1, inA2, inA3, inA4, inA5;
int inB1, inB2, inB3, inB4, inB5;
////////////////////////////////////////////////////////////////////////////////////
int distance1=0;
int filtered=0;

void setup() {
  Serial.begin(57600);
  datain="";
//------------------------------SHARP SENSOR---------------------------------
  Dist.begin(A0);
  Filter.begin();

  pinMode(27,OUTPUT);
  pinMode(38,OUTPUT);
  pinMode(33,OUTPUT);
  pinMode(32,OUTPUT);
  pinMode(39,OUTPUT);

  pinMode(25,OUTPUT);
  pinMode(23,OUTPUT);
  pinMode(31,OUTPUT);
  pinMode(29,OUTPUT);
  pinMode(30,OUTPUT);
  pinMode(28,OUTPUT);
  pinMode(37,OUTPUT);
  pinMode(35,OUTPUT);
  pinMode(34,OUTPUT);
  pinMode(36,OUTPUT);

  pinMode(7,OUTPUT);
  pinMode(3,OUTPUT);
  pinMode(4,OUTPUT);
  pinMode(5,OUTPUT);
  pinMode(6,OUTPUT);

////////////////////////////////-- SERVO--/////////////////////////////////////
   Servo1.attach(servoPin1);
   Servo2.attach(servoPin2);
   Servo3.attach(servoPin3);
   Servo4.attach(servoPin4);
   Servo5.attach(servoPin5);
```

```
  x1 = 9.638784;
  y1 = -4.25;
  catchh = 50;
 ////////////////////////////////-- PPM--////////////////////////////////////
 pinMode(interruptPin, INPUT_PULLUP);
 attachInterrupt(digitalPinToInterrupt(interruptPin), int0, RISING);
 t[0]=micros();

 ////////////////////////////////-- MPU6050 && MS5803--////////////////////////////////////
 Wire.begin();
 sensor.initializeMS_5803(false);
 TWBR = 12;                                                  //set i2c frequency
 mpu6050_register();
                  //Wait 5 seconds before continuing.
 motor_stop();
 delay(2000);

 bool togle = 0;

 for (cal_int = 0; cal_int < 200 ; cal_int ++){           //Take 2000 readings for
calibration.
   mpu6050_signal();                                    //Read the gyro output.
   raw_to_angle();                                     //chane raw to angle
   gyro_axis_cal[1] += angle_roll;                          //Ad roll value to gyro_roll_cal.
   gyro_axis_cal[2] += angle_pitch;                         //Ad pitch value to
gyro_pitch_cal.
   gyro_axis_cal[3] += angle_yaw;                           //Ad yaw value to
gyro_yaw_cal.
  }

 gyro_axis_cal[1] /= 200;
 gyro_axis_cal[2] /= 200;
 gyro_axis_cal[3] /= 200;
 cal_is_done=1;

////////////////////////////////////////////////////////////////////////////////
   sensor.readSensor();
   depth_input = sensor.pressure();
   pid_depth_setpoint = depth_input;
   pid_yaw_setpoint = angle_yaw_used-0.2;//angle_yaw - gyro_axis_cal[3];
}

void loop() {
 // put your main code here, to run repeatedly:
 digitalWrite(13,HIGH);
```

```
//----------------------------------MOTOR STOP-------------------------------------
if(channel_10 < 1300){
  motor_stop();

    pid_i_mem_roll = 0;
    pid_last_roll_d_error = 0;
    pid_i_mem_pitch = 0;
    pid_last_pitch_d_error = 0;
    pid_i_mem_yaw = 0;
    pid_last_yaw_d_error = 0;
    pid_i_mem_depth = 0;
    pid_last_depth_d_error = 0;
    pid_i_mem_x = 0;
    pid_last_x_d_error = 0;
    pid_i_mem_y = 0;
    pid_last_y_d_error = 0;
    pid_i_mem_distance = 0;
    pid_last_distance_d_error = 0;
}
else if(channel_10 > 1300){
  motor_on();
}
    ////////////////////////////////-- MPU6050 && MS5803 --////////////////////////////////////
    sensor.readSensor();
    depth_input = sensor.pressure();
    mpu6050_signal();
    raw_to_angle();

    angle_pitch = angle_pitch_acc;
    angle_roll = angle_roll_acc;
    gyro_angles_set = true;

//---------------------------------------ROLL-----------------------------------------------
  pid_roll_setpoint = 0.0;
  pid_roll_setpoint -= roll_level_adjust;
  pid_roll_setpoint /= 3;
//---------------------------------------PITCH----------------------------------------------
  pid_pitch_setpoint = 0.0;
  pid_pitch_setpoint -= pitch_level_adjust;
  pid_pitch_setpoint /= 3;
//---------------------------------------YAW------------------------------------------------
  float ch1_scale;
  channel_1 = constrain (channel_1, 1000, 2000);
  if(channel_1 < 1200){
    ch1_scale = -1;
  }
  else if(channel_1 > 1900){
```

```
   ch1_scale= 1;
  }
  else if(channel_1 > 1200 && channel_1 < 1900){
    pid_yaw_setpoint = 0;
    ch1_scale= 0;
  }
  pid_yaw_setpoint = pid_yaw_setpoint + ch1_scale;

//--------------------------------------FORWARD-----------------------------------------
float ch3_scale;
if(channel_3>1200){
  ch3_scale = 150;
}
if(channel_3>1900){
  ch3_scale = -150;
}
if(channel_3 > 1200 && channel_3 < 1900){
    ch3_scale = 0;
  }

//-----------------------------------------DEPTH-------------------------------------------
float ch2_scale;

  channel_2 = constrain (channel_2, 1000, 2000);
  if(channel_2 < 1200){
    ch2_scale= -0.1;
  }
  else if(channel_2 > 1900){
    ch2_scale= 0.1;
  }
  else if(channel_2 > 1200 && channel_2 < 1900){
      ch2_scale= 0;
    }
  pid_depth_setpoint = pid_depth_setpoint + ch2_scale;


  if (channel_6 < 1300){
  x1=9.638784,y1=-5.5;
}
  if (channel_6 > 1300){
  x1=25.5, y1=0;
}
  if (channel_7 < 1200){
  //   Servo5.write(20);
  catchh = 20;
}
  if (channel_7 > 1500){
```

```
  catchh = 50;
}
 ////////////////////////////////--SENSOR SHARP DISTANCE--////////////////////////////////
if(channel_9 > 1300){
    distance1 = Dist.getDistanceCentimeter();
    average = Filter.run(distance1);

  if(Serial.available()>0){
    char inchar = (char)Serial.read();
    datain += inchar;
    if(inchar=='\n'){
    data = true;
    }
  }
  if (data == true){
    parsingdata();
    data = false;
    datain = "";
  }
track = true;
distance = true;
}

calculate_pid();
if (track == true){
    pid_output_depth=0;
  }

if (pid_output_x <= 20 && pid_output_x >= -20 && pid_output_y <= 20 &&
pid_output_y >= -20){
  pid_output_distance=pid_output_distance;
}
else{
pid_output_distance=0;
}

    if (x >= 200 && x <= 460 && y > 100 && y <= 390) {
      if (pid_output_x <= 30 && pid_output_x >= -30) {

        ambil4();
        inverskinematic(a1, a2, x1, y1);
        serv();
        //delay(2000);
        //jepit();
      }
    }
    else  {
```

```
      lepas();

      ambil1();
      inverskinematic(a1, a2, x1, y1);
      serv();
    }
    //}
    if (x >= 270 && x <= 400 && y > 100 && y <= 390 && average <= 17 &&
!batasAtas) {
      jepit();
      batasAtas=1;
    }
    else if (x >= 270 && x <= 400 && y > 100 && y <= 390 && average > 22
&& batasAtas){
      lepas();
      batasAtas=0;
    }
}

if(channel_9 < 1300){
  pid_output_x = 0;
  pid_output_y = 0;
  pid_output_distance = 0;
  average = 0;
  pid_i_mem_x = 0;
  pid_last_x_d_error = 0;
  pid_i_mem_y = 0;
  pid_last_y_d_error = 0;

  pid_i_mem_depth = 0;
  pid_last_depth_d_error = 0;

  pid_i_mem_distance = 0;
  pid_last_distance_d_error = 0;
  track = false;
}

    m[1] = ch3_scale - pid_yaw_setpoint + pid_output_x - pid_output_distance;
//hor-right
    m[2] = ch3_scale + pid_yaw_setpoint - pid_output_x - pid_output_distance;
//hor-left
    m[3] = pid_output_depth + pid_output_pitch + pid_output_roll - pid_output_y;
//front-right
    m[4] = pid_output_depth + pid_output_pitch - pid_output_roll - pid_output_y;
//front-left
    m[5] = pid_output_depth - pid_output_pitch - pid_output_y;                  //rear-
middle
```

```
    m1a = m[1];
    m2a = m[2];
    m3a = m[3];
    m4a = m[4];
    m5a = m[5];

    m1a = constrain (m[1], -250, 250);
    m2a = constrain (m[2], -250, 250);
    m3a = constrain (m[3], -250, 250);
    m4a = constrain (m[4], -250, 250);
    m5a = constrain (m[5], -250, 250);


if(m[1]<0){
    inA1=LOW;
    inB1=HIGH;
  }
  else{
    inA1=HIGH;
    inB1=LOW;
  }

  if(m[2]<0){
    inA2=HIGH;
    inB2=LOW;
  }
  else{
    inA2=LOW;
    inB2=HIGH;
  }

  if(m3a>0){
    inA3=LOW;
    inB3=HIGH;
  }
  else{
    inA3=HIGH;
    inB3=LOW;
  }

  if(m4a>0){
    inA4=HIGH;
    inB4=LOW;
  }
  else{
    inA4=LOW;
```

```
  inB4=HIGH;
 }

 if(m5a<0){
  inA5=LOW;
  inB5=HIGH;
 }
 else{
  inA5=HIGH;
  inB5=LOW;
 }

  pwm1 = abs(m1a);
  pwm2 = abs(m2a);
  pwm3 = abs(m3a);
  pwm4 = abs(m4a);
  pwm5 = abs(m5a);

  ////////////////////////////////-- CONTROL MOTOR --////////////////////////////////////
  analogWrite(7,pwm1);       //motor tengah kiri
  digitalWrite(23,inA1);
  digitalWrite(25,inB1);

  analogWrite(3,pwm2);       //motor tengah kanan
  digitalWrite(34,inA2);
  digitalWrite(36,inB2);


  analogWrite(8,pwm3);        //motor depan kiri
  digitalWrite(31,inA3);
  digitalWrite(29,inB3);

  analogWrite(5,pwm4);        //motor depan kanan
  digitalWrite(30,inA4);
  digitalWrite(28,inB4);

  analogWrite(6,pwm5);       //motor belakang
  digitalWrite(35,inA5);
  digitalWrite(37,inB5);

//////////////////////////////////////-MANIPULATOR-//////////////////////////////////////
inverskinematic(a1,a2,x1,y1);
 if (isnan(theta1) || isnan(theta1) || isnan(theta1)) {
  theta1 = theta1new;
  theta2 = theta2new;
  theta3 = theta3new;
  return;
```

```
  }
  theta1new = theta1;
  theta2new = theta2;
  theta3new = theta3;

  int theta1deg = abs(((theta1 * 4068) / 71));
  int theta2deg = abs(((theta2 * 4068) / 71)+150);
  int theta3deg = abs(((theta3 * 4068) / 71)-150);
    Servo1.write(theta1deg);
    Servo2.write(theta2deg);
    Servo3.write(theta3deg);
    Servo5.write(catchh);
}

 ////////////////////////////////-- PPM--/////////////////////////////////////
void int0() {
    t[pulse] = micros();
    switch (pulse) {
      case 0:
      pulse++;
      break;

      case 1:
      channel_1 = t[1] - t[0];
      pulse++;
      if (channel_1 > 3000) {
        t[0] = t[1];
        pulse = 1;
        }
      break;

      case 2:
      channel_2 = t[2] - t[1];
      pulse++;
      if (channel_2 > 3000) {
        t[0] = t[2];
        pulse = 1;
        }
      break;

      case 3:
      channel_3 = t[3] - t[2];
      pulse++;
      if (channel_3 > 3000) {
        t[0] = t[3];
        pulse = 1;
        }
```

```
break;

case 4:
channel_4 = t[4] - t[3];
pulse++;
if (channel_4 > 3000) {
  t[0] = t[4];
  pulse = 1;
  }
break;

case 5:
channel_5 = t[5] - t[4];
pulse++;
if (channel_5 > 3000) {
  t[0] = t[5];
  pulse = 1;
  }
break;

case 6:
channel_6 = t[6] - t[5];
pulse++;
if (channel_6 > 3000) {
  t[0] = t[6];
  pulse = 1;
  }
break;

case 7:
channel_7 = t[7] - t[6];
pulse++;
if (channel_7 > 3000) {
  t[0] = t[7];
  pulse = 1;
  }
break;

case 8:
channel_8 = t[8] - t[7];
pulse++;
if (channel_8 > 3000) {
  t[0] = t[8];
  pulse = 1;
  }
break;
```

```
    case 9:
    channel_9 = t[9] - t[8];
    pulse++;
    if (channel_9 > 3000) {
      t[0] = t[9];
      pulse = 1;
      }
    break;
    case 10:
    channel_10 = t[10] - t[9];
    pulse++;
    if (channel_10 > 3000) {
      t[0] = t[10];
      pulse = 1;
      }
    break;

    case 11:
      t[0] = t[11];
      pulse = 1;
    break;

    default:
    pulse++;
    break;
    }

}

////////////////////////////////-- MPU6050 && MS803 --////////////////////////////////////
void mpu6050_signal(){
Wire.beginTransmission(gyro_address);
Wire.write(0x3B);
Wire.endTransmission();
Wire.requestFrom(gyro_address,14); //request 14 bit

while(Wire.available()<14);
acc_axis[1] = Wire.read()<<8|Wire.read();
acc_axis[2] = Wire.read()<<8|Wire.read();
acc_axis[3] = Wire.read()<<8|Wire.read();
temperature = Wire.read()<<8|Wire.read();
gyro_axis[1] = Wire.read()<<8|Wire.read();
gyro_axis[2] = Wire.read()<<8|Wire.read();
gyro_axis[3] = Wire.read()<<8|Wire.read();

gyro_roll = gyro_axis[1];
gyro_pitch = gyro_axis[2];
```

```
  gyro_yaw = gyro_axis[3];

  acc_x = acc_axis[1];
  acc_y = acc_axis[2];
  acc_z = acc_axis[3];
}

void mpu6050_register(){
  Wire.beginTransmission(gyro_address);
  Wire.write(0x6B);
  Wire.write(0x00);
  Wire.endTransmission();

  Wire.beginTransmission(gyro_address);
  Wire.write(0x1B);
  Wire.write(0x08);
  Wire.endTransmission();

  Wire.beginTransmission(gyro_address);
  Wire.write(0x1C);
  Wire.write(0x10);
  Wire.endTransmission();

  Wire.beginTransmission(gyro_address);
  Wire.write(0x1B);
  Wire.endTransmission();
  Wire.requestFrom(gyro_address, 1);

  while(Wire.available() < 1);

  if(Wire.read() != 0x08){
    digitalWrite(13,HIGH);
    while(1)delay(10);
    }

  Wire.beginTransmission(gyro_address);
  Wire.write(0x1A);
  Wire.write(0x06);
  Wire.endTransmission();
}

void raw_to_angle()
{
  //based on datasheet mpu6050----------------------------------------------------------
  gyro_roll_input = (gyro_roll_input * 0.7) + ((gyro_roll / 65.5) * 0.3);
  gyro_pitch_input = (gyro_pitch_input * 0.7) + ((gyro_pitch / 65.5) * 0.3);
  gyro_yaw_input = (gyro_yaw_input * 0.7) + ((gyro_yaw / 65.5) * 0.3);
```

```
  //Gyro angle calculations
  //0.0000611 = 1 / (250Hz / 65.5)
  angle_pitch += gyro_pitch * 0.0000611;
  angle_roll += gyro_roll * 0.0000611;
  angle_yaw += gyro_yaw * 0.0000611*100-0.1;

  //0.000001066 = 0.0000611 * (3.142(PI) / 180degr) The Arduino sin function is
in radians
  angle_pitch -= angle_roll * sin(gyro_yaw * 0.000001066);
  angle_roll += angle_pitch * sin(gyro_yaw * 0.000001066);
  //Accelerometer angle calculations
  acc_total_vector = sqrt((acc_x*acc_x)+(acc_y*acc_y)+(acc_z*acc_z));

  if(abs(acc_y) < acc_total_vector){
    angle_pitch_acc = asin((float)acc_y/acc_total_vector)* 57.296;
    }
  if(abs(acc_x) < acc_total_vector){
    angle_roll_acc = asin((float)acc_x/acc_total_vector)* -57.296;
    }

  angle_pitch_acc -= 0.0;
  angle_roll_acc -= 0.0;
  angle_yaw_acc -=0.0;

  angle_pitch = angle_pitch * 0.9 + angle_pitch_acc * 0.1;
  angle_roll = angle_roll * 0.9 + angle_roll_acc * 0.1;
  angle_yaw = (angle_yaw *0.01 + angle_yaw_acc * 0.1);

  angle_pitch_used = angle_roll - gyro_axis_cal[1];
  angle_roll_used = angle_pitch - gyro_axis_cal[2];
  angle_yaw_used += angle_yaw;


  pitch_level_adjust = angle_pitch_used * 15;
  roll_level_adjust = angle_roll_used * 15;
  yaw_level_adjust = angle_yaw_used * 15;
  }
 ////////////////////////////////-- CONTROL MOTOR --////////////////////////////////////
  void motor_stop(){
    digitalWrite(27,LOW);
    digitalWrite(38,LOW);
    digitalWrite(33,LOW);
    digitalWrite(32,LOW);
    digitalWrite(39,LOW);
  }
```

```
void motor_on(){
   digitalWrite(27,HIGH);    //motor tengah kiri
   digitalWrite(38,HIGH);    //motor tengah kanan
   digitalWrite(33,HIGH);    //motor depan kiri
   digitalWrite(32,HIGH);    //motor depan kanan
   digitalWrite(39,HIGH);    //motor belakang
  }

  void calculate_pid(){
  //------------------------------Roll calculations-----------------------------------------
  pid_error_temp = angle_roll_used - pid_roll_setpoint;
  pid_i_mem_roll += pid_i_gain_roll * pid_error_temp;
  if(pid_i_mem_roll > pid_max_roll)pid_i_mem_roll = pid_max_roll;
  else if(pid_i_mem_roll < pid_max_roll * -1)pid_i_mem_roll = pid_max_roll * -
1;

  pid_output_roll = pid_p_gain_roll * pid_error_temp + pid_p_gain_roll
*pid_i_mem_roll + pid_p_gain_roll *pid_d_gain_roll * (pid_error_temp -
pid_last_roll_d_error);
  if(pid_output_roll > pid_max_roll)pid_output_roll = pid_max_roll;
  else if(pid_output_roll < pid_max_roll * -1)pid_output_roll = pid_max_roll * -1;

  pid_last_roll_d_error = pid_error_temp;

  //------------------------------Pitch calculations-----------------------------------------
  pid_error_temp = angle_pitch_used - pid_pitch_setpoint;
  pid_i_mem_pitch += pid_i_gain_pitch * pid_error_temp;
  if(pid_i_mem_pitch > pid_max_pitch)pid_i_mem_pitch = pid_max_pitch;
  else if(pid_i_mem_pitch < pid_max_pitch * -1)pid_i_mem_pitch =
pid_max_pitch * -1;

  pid_output_pitch = pid_p_gain_pitch * pid_error_temp + pid_p_gain_pitch
*pid_i_mem_pitch + pid_p_gain_pitch *pid_d_gain_pitch * (pid_error_temp -
pid_last_pitch_d_error);
  if(pid_output_pitch > pid_max_pitch)pid_output_pitch = pid_max_pitch;
  else if(pid_output_pitch < pid_max_pitch * -1)pid_output_pitch =
pid_max_pitch * -1;

  pid_last_pitch_d_error = pid_error_temp;

  //------------------------------Yaw calculations-----------------------------------------
  pid_error_temp = angle_yaw_used - pid_yaw_setpoint;
  pid_i_mem_yaw += pid_i_gain_yaw * pid_error_temp;
  if(pid_i_mem_yaw > pid_max_yaw)pid_i_mem_yaw = pid_max_yaw;
  else if(pid_i_mem_yaw < pid_max_yaw * -1)pid_i_mem_yaw = pid_max_yaw *
-1;
```

```
  pid_output_yaw = pid_p_gain_yaw * pid_error_temp + pid_p_gain_yaw
*pid_i_mem_yaw + pid_p_gain_yaw *pid_d_gain_yaw * (pid_error_temp -
pid_last_yaw_d_error);
  if(pid_output_yaw > pid_max_yaw)pid_output_yaw = pid_max_yaw;
  else if(pid_output_yaw < pid_max_yaw * -1)pid_output_yaw = pid_max_yaw *
-1;

  pid_last_yaw_d_error = pid_error_temp;

  //------------------------------depth calculations-----------------------------------------
  pid_error_temp = depth_input - pid_depth_setpoint;
  pid_i_mem_depth = pid_i_mem_depth +  pid_i_gain_depth* pid_error_temp;
  if(pid_i_mem_depth > pid_max_depth)pid_i_mem_depth = pid_max_depth;
  else if(pid_i_mem_depth < pid_max_depth * -1)pid_i_mem_depth =
pid_max_depth * -1;

  pid_output_depth = pid_p_gain_depth * pid_error_temp + pid_p_gain_depth
*pid_i_mem_depth + pid_p_gain_depth *pid_d_gain_depth * (pid_error_temp -
pid_last_depth_d_error);
  if(pid_output_depth > pid_max_depth)pid_output_depth = pid_max_depth;
  else if(pid_output_depth < pid_max_depth * -1)pid_output_depth =
pid_max_depth * -1;

  pid_last_depth_d_error = pid_error_temp;

  //---------------------------------X calculations-----------------------------------------
  pid_error_temp_x = x - pid_x_setpoint;
  pid_i_mem_x = pid_i_mem_x +  pid_i_gain_x* pid_error_temp_x;
  if(pid_i_mem_x > pid_max_x)pid_i_mem_x = pid_max_x;
  else if(pid_i_mem_x < pid_max_x * -1)pid_i_mem_x = pid_max_x * -1;

  pid_output_x = pid_p_gain_x * pid_error_temp_x + pid_p_gain_x
*pid_i_mem_x + pid_p_gain_x *pid_d_gain_x * (pid_error_temp_x -
pid_last_x_d_error);
  if(pid_output_x > pid_max_x)pid_output_x = pid_max_x;
  else if(pid_output_x < pid_max_x * -1)pid_output_x = pid_max_x * -1;

  pid_last_x_d_error = pid_error_temp_x;

  //---------------------------------Y calculations-----------------------------------------
  pid_error_temp_y = y - pid_y_setpoint;
  pid_i_mem_y = pid_i_mem_y +  pid_i_gain_y* pid_error_temp_y;
  if(pid_i_mem_y > pid_max_y)pid_i_mem_y = pid_max_y;
  else if(pid_i_mem_y < pid_max_y * -1)pid_i_mem_y = pid_max_y * -1;
```

```
  pid_output_y = pid_p_gain_y * pid_error_temp_y + pid_p_gain_y
*pid_i_mem_y + pid_p_gain_y * pid_d_gain_y * (pid_error_temp_y -
pid_last_y_d_error);
  if(pid_output_y > pid_max_y)pid_output_y = pid_max_y;
  else if(pid_output_y < pid_max_y * -1)pid_output_y = pid_max_y * -1;

  pid_last_y_d_error = pid_error_temp_y;

  //--------------------------DISTANCE calculations----------------------------------
  pid_error_temp_distance = average - pid_distance_setpoint;
  pid_i_mem_distance = pid_i_mem_distance +  pid_i_gain_distance*
pid_error_temp_distance;
  if(pid_i_mem_distance > pid_max_distance)pid_i_mem_distance =
pid_max_distance;
  else if(pid_i_mem_distance < pid_max_distance * -1)pid_i_mem_distance =
pid_max_distance * -1;

  pid_output_distance = pid_p_gain_distance * pid_error_temp_distance +
pid_p_gain_distance *pid_i_mem_distance + pid_p_gain_distance
*pid_d_gain_distance * (pid_error_temp_distance - pid_last_distance_d_error);
  if(pid_output_distance > pid_max_distance)pid_output_distance =
pid_max_distance;
  else if(pid_output_distance < pid_max_distance * -1)pid_output_distance =
pid_max_distance * -1;

  pid_last_distance_d_error = pid_error_temp_distance;
  }

////////////////////////////////////////////////////////////////////////////////////////////////
void parsingdata(){
  int j = 0;

  dt[j]="";
  for (ii=1; ii<datain.length();ii++){
    if ((datain[ii]=='#')||(datain[ii]==',')){
      j++;
      dt[j]="";
    }
    else
    {
      dt[j]+=datain[ii];
    }
  }

  x = dt[0].toInt();
  y = dt[1].toInt();
```

```
  if( x==0 && y==0){
    x = 320;
    y = 240;
  }
}

//////////////////////////////////////--INVERS KINEMATICS--////////////////////////////////////
void inverskinematic(float a1,float a2,float x1,float y1){
x_invers = x1 - 7 *cos (0);
y_invers = y1 - 7 *sin (0);
theta2 = -acos((sq(x_invers) + sq(y_invers) - sq(a1) - sq(a2))/(2*a1*a2));
theta1 = atan(y_invers/x_invers) - atan((a2*sin(theta2))/(a1+a2*cos(theta2)));
theta3 = 0 - theta1 - theta2;
}

void serv() {
  Servo1.write(theta3deg);
  //    delay(kecepatan);
  Servo2.write(theta2deg);
  //    delay(kecepatan);
  Servo3.write(theta1deg);
}
void ambil1() {
  x1 = 9.638784;
  y1 = -4.25;
}
void ambil2() {
  x1 = 12, y1 = -3;
}
void ambil3() {
  x1 = 22, y1 = -3;
}
void ambil4() {
  x1 = 25.5, y1 = 0;
}

void jepit() {
  Servo5.write(70);
}

void lepas() {
  Servo5.write(20);
}

int expfilter( int input, int smoothbefore, float alpha) {
  return smoothbefore + alpha * (input - smoothbefore);
}
```

RIWAYAT HIDUP PENULIS

Erwin Ardias Saputra, lahir pada tanggal 13 maret 1992 di Palu Provinsi Sulawesi Tengah. Penulis merupakan anak ke 2 dari 3 bersaudara dari pasangan Dra. Nurul Qomariah dan Drs. Asmuji.

Penulis memulai menempuh pendidikan dari SD Negeri 1 Birobuli pada tahun 1998. Kemudian penulis melanjutkan pendidikan tingkat menengah di SMP Negeri 1 Palu pada tahun 2004. Kemudian penulis melanjutkan pendidikan di SMA Negeri 2 Palu pada tahun 2007. Pada tahun 2010 penulis memulai pendidikan Strata 1 di jurusan Teknik Elektro Universitas Tadulako Palu. Selama kuliah Penulis aktif diberbagai kegiatan akadamik seperti robotika, himpunan mahasiswa dan radio kampus. Diluar kegiatan akademik penulis juga mengikuti berbagai kegiatan non akademik seperti kelompok wirausaha dan komunitas fotografi.

Setelah lulus pada jenjang strata 1 (S1) pada tahun 2015 penulis bekerja disalah satu perusahaan televisi swasta di Palu yaitu PT. Net Mediatama Televisi (Net TV) selama kurang lebih satu tahun sebagai operation staff pada stasiun transmisi Palu.

Pada tahun 2016 penulis melanjutkan pendidikan strata 2 di Program Pascasarjana Departemen Teknik Elektro, Fakultas Teknologi Elektro Institut Teknologi Sepluh Nopember (ITS).