TESIS – EE185401

RANCANG BANGUN METODE *TRACKING OBJECT* PADA *UNDERWATER REMOTELY OPERATED VEHICLE*

TRI SUSANTO
NRP 07111450040011

DOSEN PEMBIMBING
Ronny Mardiyanto, S.T. M.T. Ph.D.
Ir. Djoko Purwanto, M.Eng. Ph.D.

PROGRAM MAGISTER
BIDANG KEAHLIAN TEKNIK ELEKTRONIKA
DEPARTEMEN TEKNIK ELEKTRO
FAKULTAS TEKNOLOGI ELEKTRO
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA
2019

TESIS – EE185401

RANCANG BANGUN METODE *TRACKING OBJECT* PADA *UNDERWATER REMOTELY OPERATED VEHICLE*

TRI SUSANTO
NRP 07111450040011

DOSEN PEMBIMBING
RONNY MARDIYANTO, S.T. M.T. PH.D.
IR. DJOKO PURWANTO, M.ENG. PH.D.

PROGRAM MAGISTER
BIDANG KEAHLIAN TEKNIK ELEKTRONIKA
DEPARTEMEN TEKNIK ELEKTRO
FAKULTAS TEKNOLOGI ELEKTRO
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA
2019

LEMBAR PENGESAHAN

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

Tri Susanto
NRP. 07111450040011

Tanggal Ujian : 20 Desember 2018
Periode Wisuda : Maret 2019

Disetujui oleh:

1. Ronny Mardiyanto, S.T., M.T., Ph.D (Pembimbing I)
   NIP: 19810118 200312 1 003

2. Ir. Djoko Purwanto, M.Eng., Ph.D (Pembimbing II)
   NIP: 19651211 199002 1 002

3. Dr. Ir. Hendra Kusuma, M.Eng.Sc (Penguji)
   NIP: 19640902 198903 1 003

4. Astria Nur Irfansyah, S.T., M.Eng., Ph.D. (Penguji)
   NIP: 19810325 201012 1 002

5. Muhammad Attamimi, B.Eng., M.Eng., Ph.D. (Penguji)
   NPP: 1985201711039

Dekan Fakultas Teknologi Elektro

Dr. Tri Arief Sardjono, S.T., M.T.
NIP. 19700212 199512 1 001

Halaman Ini Sengaja Dikosongkan

PERNYATAAN KEASLIAN TESIS

Dengan ini saya menyatakan bahwa isi keseluruhan Tesis saya dengan judul "**RANCANG BANGUN METODE *TRACKING OBJECT* PADA *UNDERWATER REMOTELY OPERATED VEHICLE* "** 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, 20 Desember 2018

Tri Susanto

NRP. 07111450040011

Halaman Ini Sengaja Dikosongkan

RANCANG BANGUN METODE *TRACKING OBJECT* PADA *UNDERWATER REMOTELY OPERATED VEHICLE*

Nama mahasiswa	:	Tri Susanto
NRP	:	07111450040011
Pembimbing	:	1. Ronny Mardiyanto, ST. MT. Ph.D
		2. Ir. Djoko Purwanto, M.Eng. Ph.D

ABSTRAK

*Underwater Remotely Operated Vehicle (*ROV) adalah robot yang memiliki kemampuan beroperasi di dalam air. Penggunaan *underwater* ROV dapat membantu pekerjaan manusia antaralain dalam bidang inspeksi lambung kapal, pengelasan dalam air. Dalam bidang perikanan, *underwater* ROV biasanya digunakan untuk pelestarian terumbu karang. Dalam bidang *rescue, underwater* ROV digunakan untuk mencari korban yang tenggelam diperairan. *underwater* ROV ini sering digunakan untuk menggantikan penyelam dalam melakukan pencarian korban. Hal ini dilakukan karna penyelam memiliki keterbatasan penglihatan ketika berada di dalam air. Instansi pemerintah dan swasta menggunakan teknologi *underwater* ROV ini dengan tujuan yang berbeda.

Tujuan dari penelitian ini adalah membuat rancang bangun metode *tracking object* pada *underwater* ROV. Penelitian ini dilakukan pengolahan citra yang dimediakan pada *underwater* ROV menggunakan *raspberry* sebagai *processor*, Arduino Mega sebagai kontrol pada *underwater* ROV dan Open CV sebagai pengolahan citra. *Underwater* ROV dapat melakukan pencarian korban berdasarkan warna yang telah ditetapkan. Ketika warna telah ditetapkan, *underwater* ROV tersebut mengikuti pergerakan objek tersebut.

Rata-rata dalam waktu sekitar 1 detik *underwater* ROV sudah dapat mengikuti pergerakan objek pada kondisi air jernih. Pengujian yang dilakukan dengan deteksi warna HSV mendapatkan kesalahan deteksi sebesar 44.83% dan jika dibandingkan dengan deteksi RGB memiliki nilai kesalahan sebesar 62.41%.

Kata Kunci: kamera, objek, *Underwater* ROV, *tracking object*.

Halaman Ini Sengaja Dikosongkan

DESIGN METHOD TRACKING OBJECT IN REMOTELY OPERATED UNDERWATER VEHICLE (ROV)

Student Name	: Tri Susanto
Student Identity Number	: 07111450040011
Supervisors	: 1. Ronny Mardiyanto, ST. MT. Ph.D
	2. Ir. Djoko Purwanto, M.Eng. Ph.D

ABSTRACT

Underwater Remotely Operated Vehicle (ROV) is a robot which has the ability to operate in the water. The operation of underwater ROV can help human's work, such as, in the inspection of ship hulls sector , it can be used for welding in water. In the fishery's sector, underwater ROV is usually used for the preservation of coral reefs. In the rescue's sector , the ROV underwater is used to look for victims who drown in the waters. underwater ROV is often used to replace divers in searching for victims. It's because divers have limited vision when they are in the water. Government and private agencies use this underwater ROV technology with different objectives.

The purpose of this research is to make the design of the object tracking method in the underwater ROV. This research carried out image processing provided in the underwater ROV using raspberry as a processor, Arduino Mega as a controller on underwater ROV and Open CV as image processing. Underwater ROV can search victims based on the colors has been set. It follows the movement of the object.

On the average for about 1 second the underwater ROV is able to follow the movement of objects in clear water conditions. In the experiment was performed with HSV color detection get a detection error of 44.83% and when compared with RGB detection it has an error value of 62.41%.

Keywords: camera, object, Underwater ROV, tracking

Halaman Ini Sengaja Dikosongkan

KATA PENGANTAR

Alhamdulillah, puji dan syukur penulis panjatkan kepada Allah SWT, karena atas segala nikmat, dan taufik Nya lah tesis ini dapat diselesaikan. Tesis berjudul "**RANCANG BANGUN METODE TRACKING OBJECT PADA REMOTELY OPERATED UNDERWATER VEHICLE (ROV)**" ini disusun untuk memenuhi sebagian persyaratan memperoleh gelar Magister Teknik (MT) pada Departemen Teknik Elektro, Fakultas Teknologi Elektro, Institut Teknologi Sepuluh November.

Penulis menyadari bahwa dalam penyusunan tesis ini tidak terlepas dari bantuan berbagai pihak. Oleh karena itu, dengan ketulusan dan kerendahan hati penulis menyampaikan terima kasih kepada:

1. Bapak Ronny Mardiyanto S.T., M.T., Ph.D. selaku Dosen Pembimbing, yang telah banyak memberikan saran, bantuan, serta sabar dalam membimbing penulis.
2. Bapak Ir. Djoko Purwanto, M.Eng., Ph.D. selaku Dosen Pembimbing, yang telah banyak memberikan saran, bantuan, serta sabar dalam membimbing penulis.
3. Bapak Dr. Tri Arief Sardjono, S.T., M.T. selaku Pembimbing Akademik.
4. Bapak Achmad Arifin, S.T., M.Eng., Ph.D. dan Bapak Dr. Muhammad Rivai, S.T., M.T. selaku Dosen Pengajar Studi S2.
5. Pimpinan dan civitas akademika Jurusan Teknik Elektro FTI – ITS.
6. Ibu, Bapak, dan saudara, atas segala dukungan dan doanya hingga terselesaikannya tesis ini.
7. Ali Zainal, Dhadhang S.B.W, Ali Uroidi, semua rekan S2 Pasca ITS angkatan 2014 dan semua rekan rekan Cometronica , Sekolah Robot Indonesia.

Pada akhirnya, penulis menyadari bahwa tesis ini masih belum sempurna. Oleh karena itu, penulis sangat mengharapkan kritik dan saran yang membangun. Penulis berharap semoga tesis ini dapat bermanfaat bagi pengembangan ilmu pengetahuan dan teknologi serta bagi masyarakat.

Surabaya, 5 Desember 2018

Penulis

Halaman Ini Sengaja Dikosongkan

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

Halaman Ini Sengaja Dikosongkan

BAB 1
PENDAHULUAN

1.1 Latar Belakang

Perkembangan teknologi menjadi peran penting dalam kehidupan manusia. Teknologi robotik merupakan salah satu contoh perkembangan yang besar manfaatnya, antara lain dalam dunia kedokteran, pertanian, dan militer. Indonesia merupakan Negara yang terdiri dari pulau-pulau yang dikelilingi lautan. Rekapitulasi kejadian laut ditunjukan pada Tabel 1.1 yang dikutip dari Badan Keamanan Laut Republik Indonesia terhitung tanggal 28 Maret - 3 April 2015. Sebelumnya kecelakaan yang sudah terjadi pada Pesawat Air Asia QZ8501 yang terjatuh di perairan Indonesia yang di dalamnya menampung 155 penumpang dan 7 orang kru di dalam pesawat.

Kecelakaan yang terjadi di dalam air sering melibatkan manusia sebagai metode penyelamatan. Peralatan selam, kacamata dan tabung oksigen adalah alat yang wajib digunakan seorang penyelam dalam melakukan penyelamatan. Sementara manusia memiliki fisik dan stamina yang terbatas. Waktu dan pandangan adalah faktor utama kelemahan dari manusia dalam melakukan pencarian korban. Tabung oksigen yang biasa digunaka hanya bertahan beberapa waktu saja. Dan keterbatasan manusia dalam melihat obyek dipengaruhi oleh keruhnya air.

Elfram Yuliawan seorang tim SAR Indonesia yang bertugas di area Jakarta, tergabung dalam tim pencarian korban bencana Air Asia kemarin memberikan data-data survey tentang penyelamatan. Narasumber mengatakan, untuk melakukan penyelaman secara manual atau dilakukan oleh seorang penyelam memiliki kendala pada cuaca, kedalaman, waktu dan jarak pandang. Jarak pandang dipengaruhi oleh cuaca buruk seperti mendung yang dapat melemahkan pandangan penyelam. Dasar penglihatan penyelam di sekitar area saat cuaca baik yaitu 5m – 7m. Waktu dari penyelaman di pengaruhi oleh kedalaman, pada kecelakaan Air Asia dengan kedalamannya 35 meter maka penyelam melakukan 1 kali penyelaman dalam waktu 15 menit. Gambar 1.1 BASARNAS

menggunakan ROV untuk pencarian Air Asia QZ8501. ROV ini dapat menyelam di kedalaman hingga 100 meter. Pada ROV tersebut dilengkapi dengan sensor sonar dan kamera [1].

Tabel 1.1 Rekapitulasi Kejadian Laut

No	Kejadian	Lokasi	Waktu Kejadian
1	Kapal pembawa BBM illegal terbakar di Pasie Nan Tigo	Muaro Anai, Pasir Nan Tigo Kecamatan Koto Tangah, Padang	Sabtu, 28 Maret 2015
2	Nelayan terjatu da tenggelam ketika kapalnya dihantam gelombang	Perairan Tanjung Berikat	Minggu, 29 Maret 2015
3	Perahu mengalami mati mesin, 3POB berhasil di selamatkan	Perairan Maumere, Kupang	Senin, 30 Maret 2015
4	Pol air Jatim tangkap 3 kapal nelayan, tanpa dokumen	Perairan Madura	Selasa, 31 Maret 2015
5	Korban tenggelam ditemukan meninggal dunia	Perairan Nusa Barong	Rabu, 1 April 2015
6	Nelayan yang tenggelam itemukan tersangkut jarring	Perairan Braatan Pesisir, Kecamatan Pademawu, Pamekasan	Kamis, 2 April 2015
7	Terseret ombak di pantai Mata Nurung, Bryan di evakuasi dengan selamat	Pantai Mata Nurung kecamatan Tepah Selatan Kabupaten Simueulue	Jumat, 3 April 2015

Gambar 1.1 ROV BASARNAS [1]

Pada penelitian ini, penulis akan merancang metode *tracking object* pada robot ROV. Robot ini akan digunakan untuk membantu pekerjaan manusia dalam mencari korban yang tenggelam. ROV ini dikendalikan secara manual oleh operator, dan setelah proses pengolahan citra ROV akan bekerja secara otomatis. ROV bekerja otomatis setelah dapat menentukan objek yang ditentukan melalui warna pilihan.

1.2 Rumusan Masalah

Berdasarkan tinjauan dari latar belakang yang telah dijelaskan, maka dalam penelitian ini dapat dirumuskan masalah sebagai berikut:

1. Bagaimana merancang bagun *underwater* ROV
2. Bagaimana merancang bangun metode untuk mengidentifikasi objek di dalam air
3. Bagaimana merancang sistem kendali pergerakan *underwater* ROV.
4. Bagaimana *underwater* ROV dapat mengikuti objek yang teridentifikasi.

1.3 Tujuan dan Manfaat

Secara umum tujuan dari penelitian ini adalah membatu meringankan pekerjan manusia. Adapun tujuan dari penelitian ini sebagai berikut:

1. *Underwater* ROV dapat mengindetifikasi objek.
2. *Underwater* ROV dapat mempertahankan posisi terhadap pergerakan objek.
3. *Underwater* ROV dapat mempermudah proses evakuasi di dalam air.

Adapun manfaat yang di dapat dari penelitian ini adalah *underwater ROV* dapat mempermudah dalam pencarian korban yang disimulasikan dengan objek.

1.4 Keluaran yang Diharapkan

Keluaran yang diharapkan setelah dilakukan penelitian ini, *underwater* ROV dapat mengidentifikasi benda yang ditentukan sebagai simulasi korban dan dapat mengikuti pergerakan dari objek tersebut.

1.5 Kontribusi

Dari hasil penelitian yang telah dilakukan, diharapkan dapat memberikan kontribusi terhadap perkembangan teknologi *underwater* ROV, khususnya pada sistem *tracking* objek serta memberikan kontribusi pada masyarakat dan pemerintahan dalam proses evakuasi.

BAB 2
KAJIAN PUSTAKA DAN DASAR TEORI

ROV dikenalkan pada awal tahun 1953 oleh Dimitri Rebikoff, teknologi ROV dikembangkan melalui PUV (*Programmed Underwater Vehicle*) yang dibuat oleh *Luppis-Whitehead Automobile* di Australia pada tahun 1864. Menurut *Marine Technology Society ROV Committee's* dalam *Operational Guidelines for ROVs* (1984) dan *The National Research Council Committee's* dalam *Undersea Vehicles and National Needs* (1996) ROV merupakan robot bawah laut yang dioperasionalkan oleh operator. *Underwater* ROV harus dapat menjaga posisi yang diinginkan. Pada dasarnya *underwater* ROV ini disebut juga kapal selam mini tanpa awak, dimana posisi awak kapal digantikan oleh operator yang berada di *ground station*.

Perkembangan teknologi *underwater* ROV ini sangat pesat dimulai dari negara Amerika Serikat yang memanfaatkan ROV dibidang militer angkatan laut sebagai robot penyelamatan dan pengambilan objek yang ada di dasar laut. Perkembangan ROV selanjutnya dilakukan untuk daerah lepas pantai lainnya seperti kegiatan ekplorasi, eksploitasi, dan distribusi migas. Perusahaan Hydro Products menggunakan ROV RCV-225 dan RCV-150 sebagai perusahaan pertama yang menggunakan *underwater* ROV dengan tujuan pengeboran minyak di lepas pantai.

2.1 Referensi Penelitian

Pengembangan *underwater* ROV pada perusahaan Slingsby Engineering Limited (SEL) yang sebelumnya memiliki berbagai macam jenis ROV seperti MMIM, SOLO, ORVIL, CIRRUS, PIC, dan OBSERVER. Pengembangan teknologi baru yang diterapkan pada ROV Trojan. ROV Trojan memiliki sistem perangkat lunak yang lebih baik dan nilai jual lebih rendah debandingkan generasi sebelumnya [2].

Underwater ROV dalam bidang pemeliharaan berkala atau inspeksi pada lambung kapal dan dermaga. Efek dari pembiasan air dapat diredam dengan

menggunakan sistem *stereo vision* atau sistem penglihatan. Sehingga dengan menggunakan kamera sistem *stereo vision* efek yang awalnya terekam miring dapat diperbaiki sampai menjadi tampilan citra yang sebenarnya [3].

Perancangan sistem pelacakan objek dengan menggunakan sensor laser dan kamera CCD. Dengan menggunakan metode *active conturs* pada *computer vision* dan deteksi tepi pada sistem pelacakan tersebut. Penggunaan sistem informasi yang diberikan oleh kamera kendaraan dan proyeksi dua laser *pointer* di bidang optik kamera. Sistem menghitung koordinat x, y, z dan sudut *yaw* kendaraan. Prosedur eksperimen membuktikan keakuratan pengukuran sistem, karena mereka dibandingkan menggunakan sistem pelacakan posisi komersial. Selain itu, sistem berhasil digunakan dalam gerakan kontrol loop tertutup skema, memberikan umpan balik halus dan akurat untuk kontrol [4].

Sistem *tracking* dengan warna dasar yang dikombinasikan dengan kamera PTZ. Kamera dapat menangkap objek, kemudian dilakukan proses pemetaan untuk menentukan titik tengah dari objek tersebut. Selanjutnya digunakan kamera PTZ sebagai pengikut gerakan objek tersebut.

Optical flow dan CNN digunakan untuk pelacakan sebuah objek. Dengan menggunakan metode ingrate adaptif untuk memperkenalkan mekanisme persaingan antara aliran optic CNN dan KLP mengkompensasi *over-fitting*. Dihasilkan kemampuan melacak target dalam aliran video dalam aspek *motion blur* dalam gerakan yang cepat.

Desain sistem struktur posisi pada ROV dengan menggunakan metode *Variable Structure Model-Reference Adaptive Control* (VS-MRAC). Dengan hasil keluaran VS-MARC lebih bagus dibandingkan P-PI *Control* [5]. Penggunaan efisien manipulator pada ROV di perairan eropa. Pada jurnal ini diterapkan dengan metode *Barycenter*. Di mana dalam menggunakan metode ini energi lebih renda saat manipulator melakukan *payload* atau beban diberikan [6].

Penelitian ini melakukan eksperimen terhadap kontroler sebuah ROV. Dilakukan pada ROV ODIN III yang sebelumnya menggunakan kontroler berbasis PID. Pada eksperimen ini menerapkan sistem (ADOB) yang di dalamnya terdapat *regressor-free adaptive control* dan *disturbance observer* (DOB). Karena pada awalnya dengan menggunakan *tuning* PID yang terlalu lama maka dengan

mengadopsi sistem ADOB pada robot bawah air dengan parameter kondisi lingkungan yang berbeda-beda, bisa dialihkan dengan (DOB) [7].

Sistem *autonomus* menggunakan metode *tracking color algorithms.* Dengan mengklasifikasikan warna yang sudah dijadikan inputan atau parameter sebagai pelacakan warna. Kondisi yang bisa menghambat dalam pelacakan warna adalah sinar yang berasal dari permukaan [8].

Sistem *tracking object* pada sensor sonar dengan menggunakan metode *optical flow.* Pada pendekatan *optical flow* dengan menggunakan filter FFT dapat mengamati benda yang statis dan benda yang bergerak. Pada metode Lucas, perhitungan aliran optik telah terbukti untuk bekerja secara memadai dengan gambar sonar pada sampel antara 1-4 *frame*/*s* dan asumsi gerakan halus telah divalidasi dalam kondisi ini, dengan bantuan dari beberapa penyaringan *Gaussian*. Kesalahan prediksi posisi antara 10 cm dan 50 cm diamati pada pengaturan rentang 10-m (1% -5% dari kisaran *scan*) dengan menggunakan estimasi gerak aliran optik [2]. *Tracking* pada ubur-ubur yang menggunakan kamera konvensional. Dengan menerapkan metoda *segmentation* pada ROV dilakukan dengan otomatis selama 89 menit [9].

Pengikut objek pada permukaan air, penelitian ini disebutkan bahwa sama halnya dengan pertandingan robot *soccer*. Dengan meletakkan kamera tepat di atas kolam sebagai pengindraan untuk pengolahan citra. Menggunakan *template matching algorithm* di mana bahwa hasil akhir hampir semua gangguan, terutama refleksi di permukaan air, akan dihapus atau dikurangi [10].

Penggabungan antara tracking warna dan pergerakan objek, di mana penelitian ini menggunakan *optical flow* sebagai *tracking* pergerakan objek. Pelacakan objek dilakukan atas dasar wilayah sifat seperti centroid. Pada latar belakang gambar dilakukan penghitaman warna untuk mempermudah pemilihan warna [11].

Pada penelitian ini dilakukan *tracking objeck* dengan menggunakan *active camera. Active camera* dimaksud adalah sister visual servoing, dengan menggunakan pelacakan warna dan sistem *maping* untuk menentukan koordinat dari benda tersebut [12].

Pada beberapa penelitian *optical flow* dapat digunakan dalam objek segmentasi, pada gambar inframerah dapat lebih akurat [13]. Ada juga yang beranggapan dengan metoda piramida *lucas canade, optical flow* dapat melacak target dengan kecepatan yang lebih tinggi. [14].

Pendeteksian tepi pada *optical flow* dapat bekerja pada kendaraan yang bergerak dengan menggunakan bantuan analisis *cluster* [15]. Beberapa penambahan yang didapat berupa informasi dari kendaraan tersebut dapat menjadikan acuan tracking sistem seperti informasi *contur* [16] [17].

Penggunaan metode *template maching* konvensional terdapat kelemahan dalam pencocokan gambar akibat cahaya yang berubah, sehingga ditingkatkan pada metode *weighted template matching* [18]. Pengembangan dari *template maching* dengan *adaptyive template algorithm* maka objek yang terdeteksi dapat selalu diperbaruhi. [19]. Penggunaan metode *template matching* ini dapat dilakukan pada kendaraan seperti pada drone [20].

Pendeteksian juga bisa menggunakan warna yang terdapat pada objek tersebut. Metode *fourier* digunakan sebagai deteksi warna yang digunakan beberapa peneliti sebelumnya dengan menggabungkan deteksi bentuk benda tersebut [21]. Pada deteksi warna tersebut dapat berlaku pada pendeteksian wajah dengan menggunakan warna pada kulit wajah. Ada penelitian dengan menggunakan deteksi wajah dengan menggunakan warna kulit dan digabung dengan jarak kamera terhadap wajah tersebut. Objek awal dapat disegmentasikan dari latar belakang menggunakan *histogram siparitas stereo* yang merepresentasikan informasi jangkauan objek [22].

Framework *Tracking Learning Detection* (TLD) dapat membantu pendeteksian objek pada video yang beresolusi tinggi. TLD sendiri bisa dikatakan mirip dengan *template maching*. Tetapi dalam kekurangannya ada beberapa penelitian menambahkan metode *down-sample* [23]. Ada juga dengan menggunakan merode pemotongan *region of interest* (ROI) [24]. Dan kemudian dikembangkan lagi dengan penambahan *P-N Learning*. Dimana *P-Learning* mengamati objek yang lolos seleksi atau terdeteksi. Dan *N-Learning* kebalikannya [25].

2.2 Citra

Citra adalah gambar pada bidang dua dimensi, citra merupakan fungsi menerus atau *continue* dari intensitas cahaya pada suatu bidang. Sumber cahaya yang masuk ke benda akan dipantulkan kembali ke berkas cahaya tersebut sehingga tertangkap oleh alat optik seperti mata pada manusia, kamera, *scanner*, dan lainnya sehingga terbentuklah suatu citra atau gambar yang biasa kita lihat.

Citra terbagi menjadi dua yaitu citra analog dan citra digital [26]. Citra analog adalah suatu citra kontinu seperti gambar pada monitor televisi, foto sinar X, hasil CT Scan dan lainnya, sedangkan pada citra digital adalah citra yang bisa langsung diolah oleh suatu media seperti komputer.

Citra digital diwakilkan oleh sebuah matrik yang terdiri dari M kolom dan N baris. Perpotongan antara kolom dan baris bisa disebut dengan *pixsel* (*picture element*) yaitu yang merupakan elemen terkecil dari suatu citra. Piksel terdiri dari dua buah parameter, yaitu koordinat dan intensitas atau sering disebut warna koordinat *(x,y)* yang memiliki nilai berupa *f(x,y)* bisa dilihat pada Persamaan 2.1.

$$f(x,y) = \begin{bmatrix} f(0,0) & f(0,1) & \cdots & f(0,M-1) \\ f(1,0) & \cdots & \cdots & f(1,M-1) \\ \cdots & \cdots & \cdots & \cdots \\ f(N-1,0) & f(N-1,1) & \cdots & f(N-1,M-1) \end{bmatrix} \quad (2.1)$$

dengan,

f(x,y) : fungsi intensitas

M : jumlah baris, $0 \leq y \leq M-1$

N : kolom, $0 \leq y \leq N-1$.

Hasil dari proses sampling dan kuantitas terbentuklah matrik *M x N*.

2.2.1 Pengolahan Citra

Secara visual melalui penglihatan mata, suatu gambar atau citra terlihat baik atau sempurna. Pada dasarnya jarang atau hanya 10% bisa dijumpai citra yang didapat melalui kamera atau perangkat lainnya menunjukkan hasil yang berkualitas. Pada suatu citra yang terlihat cacat tersebut terdapat kekurangan seperti warna terlalu kontras, kurang tajam, kabur, dan lainnya. Dengan kecacatan suatu citra tersebut menjadi lebih sulit diinterpretasikan.

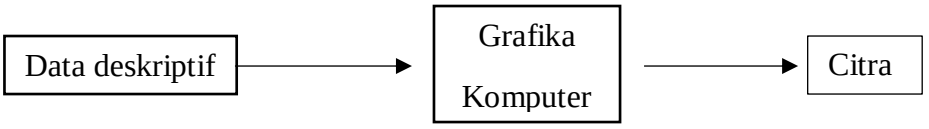
Untuk membuat suatu citra menjadi mudah untuk diinterprestasikan, dilakukan manipulasi agar citra tersebut bisa disempurnakan kualitasnya. Biasa dinamakan dengan pengolahan citra (*image processing*). Pengolahan citra ini biasanya berkaitan dengan komputer yang memiliki citra yang buruk akan diubah kualitasnya menjadi lebih baik.

Pada dasarnya pengolahan citra akan dilakukan apabila citra tersebut:

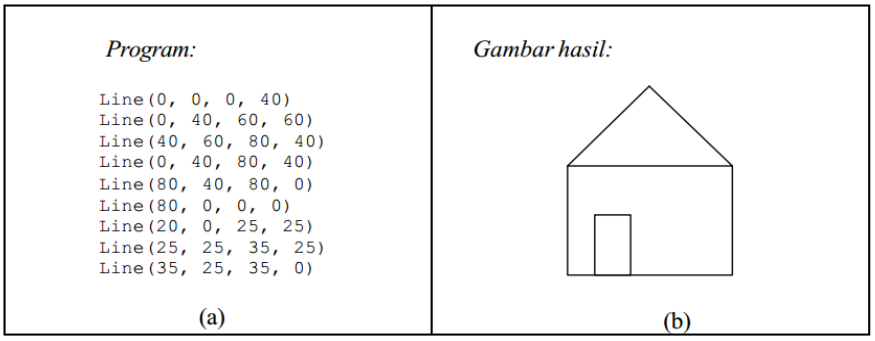
1. Meningkatkan suatu kualitas dari sebuah citra. Dari sebuah citra yang rusak pada pencahayaan dan lainnya sehingga dilakukan perbaikan pada citra tersebut.
2. Pengelompokan atau pengukuran suatu citra.
3. Penggabungan citra.

Pada bidang komputer, terdapat tiga bidang studi yang berkaitan dengan data citra dengan tujuan berbeda antara lain [26]:

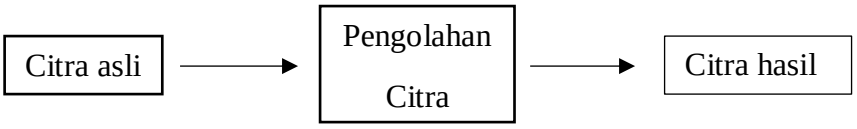
1. Grafika komputer yang bertujuan menghasilkan sebuah citra. Dalam membuat sebuah citra grafika memerlukan data deskriptif seperti, koordinat titik, panjang garis, jari-jari lingkaran, warna, dan lainnya. Gambar 2.2 merupakan sebuah contoh dari bentuk gambar rumah yang dibuat oleh garis-garis lurus, dengan masukan berupa koordinat awal dan koordinat ujung garis.
2. Pengolahan citra yang bertujuan memperbaiki kualitas citra gambar agar gambar citra tersebut bisa lebih mudah diinterpretasikan oleh manusia atau mesin. Dapat dilihat pada Gambar 2.3 dimana dari input yang berupa citra dapat diolah atau diperbaiki lagi ke dalam bentuk citra, dengan hasil yang lebih baik. Pada contoh Gambar 2.4 diketahui bahwa dari gambar yang semula merupakan gambar penuh dengan *noise* atau derau ini dilakukan pengolahan dengan operasi penapisan atau *filtering* sehingga gambar tersebut lebih halus atau disempurnakan kembali [26].
3. Pengenalan pola data yang ada dalam citra tersebut dikelompokkan menjadi data numerik dan simbolik. Diagram blok pengenalan pola dapat dilihat pada Gambar 2.5. Tujuan dari pengelompokan ini adalah untuk mengenali objek di dalam citra tersebut. Adapun contoh pengenalan pola dapat dilihat pada Gambar 2.6, terdapat suatu tulisan tangan dengan huruf A. Dengan menggunakan suatu pola pada algoritma diharapkan komputer tersebut bisa mengenali karakter huruf A [26].



Gambar 2.1 Diagram Blok Grafika Komputer [26]



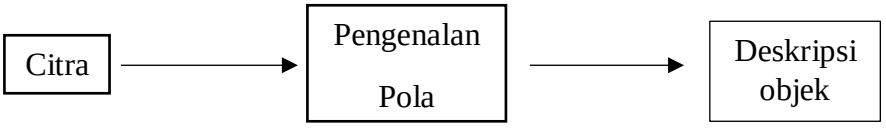
Gambar 2.2 (a) Program Grafika Komputer (b) Gambar Hasil "Rumah" [26]



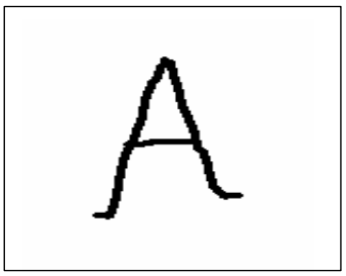
Gambar 2.3 Diagram Blok Pengolahan Citra [26]

(a) (b)

Gambar 2.4 (a) Gambar Dengan *Noise* (b) Gambar Yang Sudah Diolah [26]



Gambar 2.5 Diagram Blok Pengolahan Pola



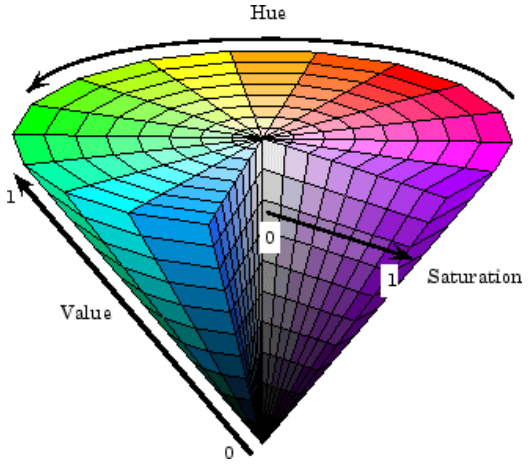
Gambar 2.6 Citra Karakter ‘A’ Sebagai Masukan Pengenalan Huruf [26]

2.3 Pengolahan Warna

Terdapat berbagai macam model warna pada pengolahan citra. Di antaranya yang dapat dipergunakan adalah RGB dan HSV. Dimana RGB adalah warna dasar pada citra itu sendiri. Tiga komponen dasar RGB antara lain *red, green* dan *blue.* Sedangkan pada model HSV memiliki komponen warna *hue, saturation,* dan *value. Hue* adalah satuan panjang gelombang dari warna dominan. *Saturation* banyaknya cahaya putih dan *value* menentukan ketajaman dari warna tersebut. Pada Gambar 2.7 menunjukan ruang warna pada HSV.

Untuk konversi RGB menjadi HSV menggunakan Persamaan 2.2-2.4 [27]. Nilai *hue* didapat dari Persamaan 2.2, nilai *saturation* didapat dengan menggunakan Persamaan 2.3 dan *value* didapat dengan Persamaan 2.4.

$$H = tan\left(\frac{3(G-B)}{(R-G)+(R-B)}\right) \tag{2.2}$$



Gambar 2.7 Ruang Warna HSV [27]

$$S = 1 - \frac{min(R,G,B)}{V} \quad (2.3)$$

$$V = \frac{R+G+B}{3} \quad (2.4)$$

dengan,

H = *Hue*

S = *Saturation*

V = *Value*

R = nilai dari *Red*

G = nilai dari *Green*

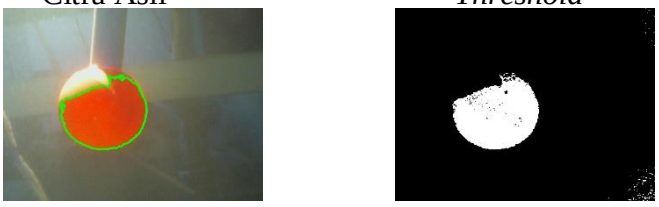
B = nilai dari *Blue.*

2.4 Metode *Threshold*

Threshold adalah metode yang sederhada dalam pengolahan citra. Metode ini digunakan untuk mengubah citra warna menjadi biner. *Threshold* memisahkan antara objek dengan background dalam suatu citra berdasarkan perbedaan tingkan kecerahan dan gelap dari citra tersebut. Warna gelap dipresentasikan dengan nilai 0 dan cerah dengan nilai 1. *Threshold* ditunjukan pada Persamaan 2.5.

$$g(x,y) = \begin{cases} 1 \; if \; f(x,y) > T \\ 0 \; if \; f(x,y) \leq T \end{cases} \quad (2.5)$$

Citra Asli *Threshold*



Gambar 2.8 Gambar *Threshold*

Nilai $g(x, y)$ merupakan hasil dari *threshold* untuk setiap pixel warna pada *f(x,y)*. Jika nilai *T* konstan makan disebut global *threshold*. Warna hitam dapat diberikan jika nilai pixel dari citra teresebut di bawah *T* dan jika nilai pixel citra tersebut lebih dari *T* maka dapat diberi warna putih. Contoh dari *threshold* ditunjukan pada Gambar 2.8.

2.5 Computer Vision

Computer Vision adalah ilmu dalam bidang teknologi pencitraan komputer. Ilmu ini dapat mengekstrak informasi dari gambar yang diperlukan sesuai kebutuhannya. Secara langsung semuanya diolah oleh komputer untuk mendapatkan data visual.

Computer vision merupakan sebuah proses otomatis yang mengintegrasikan sejumlah besar proses persepsi visual, seperti pengolahan citra, klasifikasi citra, pengenalan citra, dan akusisi citra. *Computer vision* didefinisikan sebagai salah satu cabang ilmu pengetahuan yang mempelajari bagaimana komputer dapat mengenali objek yang diamati atau diobservasi. Proses-proses dalam computer vision dapat dibagi menjadi tiga aktivitas: .

1. Memperoleh atau mengakuisisi citra digital.
2. Melakukan teknik memecahkan masalah dalam memperoses atau memodifikasi data citra.
3. Menganalisis citra dan menggunakan hasil pemrosesan untuk tujuan tertentu, misalnya perintah dan mengontrol robot, memantau proses manufaktur, dan lain-lain.

Beberapa kombinasi dari *Computer Vision*:

1. Pengolahan citra (*Image Processing*), bidang yang berhubungan dengan proses transformasi citra dengan tujuan menghasilkan suatu citra yang lebih berkualitas.
2. Pengenalan pola (*Pattern Recognition*), bidang ini berhubungan dengan proses identifikasi objek pada citra atau interpretasi citra. Proses ini bertujuan untuk mendapatkan informasi/pesan yang disampaikan oleh gambar atau citra.

2.6 Visual Servoing

Visual servoing adalah suatu cara robot mengontrol dirinya sendiri pada posisi yang telah ditentukan berdasarkan referensi yang didapat melalui informasi sebuah gambar dari kamera [28]. Sebuah kamera yang dianggap sebagai sensor *vision* adalah bagian yang terpenting dalam proses *visual servoing*, dimana kamera tersebut akan memberikan informasi agar robot dapat bergerak secara dinamis untuk mengikuti objek yang diinginkan. Penempatan kamera diletakkan di badan robot itu sendiri. Untuk mendapatkan hasil pencitraan yang sempurna, maka teknik pengambilan citra dan pencahayaan perlu diperhatikan. Tujuan dari penggunaan *visual servoing* ini adalah bagaimana caranya untuk meminimalisir nilai *error* $e(t)$ yang ada. Persamaan umum *visual servoing* ditunjukan pada Persamaan 2.6,

$$e(t) = s(m(t), a) - s^* \tag{2.6}$$

dengan,

vektor $m(t)$	: hasil koordinat dari titik tengah gambar
a	: parameter informasi tambahan dari suatu gambar tersebut
s	: fitur vektor
s^*	: nilai konstan.

Visual servoing terdiri dari 2 macam yaitu *image-based visual servo kontrol* (IBVS) yang bekerja pada keadaan citra tersebut yang akan dicari nilai *error* antara nilai koordinat yang lama dengan nilai koordinat yang baru. Kedua adalah *position-based visual kontrol* (PBVS) yang bekerja berdasarkan sudut pandang tiga dimensi yang diperoleh dari peragaan gambar berdasarkan posisi yang diinginkan. *Visual servoing* ini biasanya dipakai pada robot manipulator dan juga pada robot *mobile*.

2.7 OpenCV

Open Computer Vision (OpenCV) adalah suatu *software program interface library* yang sering digunakan dalam proses pengolahan citra. C*omputer vision* adalah salah satu cabang dari bidang ilmu pengolahan citra yang

memungkinkan komputer dapat melihat seperti manusia. Dengan menggunakan *OpenCV* tersebut komputer membentuk suatu perintah seperti pengambilan keputusan, melakukan suatu perintah, dan mengenali suatu objek. Beberapa implementasi dari *OpenCV* adalah *face recognition, face detection, face/object tracking, road tracking,* dan lainnya. *OpenCV* ini menyediakan *library open source* untuk *computer vision* seperti C/C++, *OpenCV* didesain untuk aplikasi real-time, memiliki fungsi-fungsi akuisisi yang baik untuk image/video. Fitur yang dimiliki OpenCV antara lain :

1. Manipulasi data citra.
2. Citra dan video I/O.
3. Manipulasi matriks dan vektor.
4. Data struktur dinamis (*lists, queues, sets, trees, and graphs*). Pemroses citra fundamental (*filtering, edge detection, corner detection, sampling, and interpolation, color conversion, morphological operations, histograms, image pyramids).*
5. Analisis struktur.
6. Dapat mengkalibrasi kamera (*calibration patterns*, estimasi fundamental *matrix*).
7. Estimasi *homography*, stereo correspondence.
8. Analisis gerakan (*optical flow*, segmentation, tracking).
9. Pengenalan objek.
10. *Graphical User Interface* (*display image*/*video*, penanganan *keyboard* dan *mouse handling, scroll-bars*).

2.8 Tekanan Pada Zat Cair (Hukum Archimedes)

Berkaitan dengan ROV yang merupakan robot *under water*, pada robot ini harus diperhatikan kondisi robot tersebut. Jika sebua benda dicelupkan ke dalam zat cair, maka benda tersebut akan mendapatkan gaya yang disebut gaya apung (gaya ke atas) sebesar berat zat cair yang dipindahkannya. Hubungan antara berat benda dapat dinyatakan dengan Persamaan 2.7,

$$W_s = W - F_a \tag{2.7}$$

dengan,

Ws : berat benda dalam zat cair $(Kg.\frac{m}{s^2})$

W : berat benda sebenarnya $(Kg.\frac{m}{s^2})$

Fa : gaya apung (N).

Besarnya gaya apung (Fa) dirumuskan seperti pada Persamaan 2.8,

$$F_a = \rho cair V_b g \tag{2.8}$$

dengan,

ρcair : massa jenis zat cair (kg/m^3)

Vb : volume benda yang tercelup (m^3)

g : percepatan gravitasi (m/s^2).

2.8.1 Benda Dalam Hukum Archimedes

Dalam Hukum Archimedes ada tiga kemungkinan ketika benda dicelupkan ke dalam air, yaitu tenggelam, melayang, dan terapung.

1. Benda Tenggelam

Benda dikatakan tenggelam ketika posisi benda berada di dasar air, dimana benda itu dimasukkan. Posisi benda tenggelam dapat dilihat pada Gambar 2.9. Pada benda tenggelam terdapat tiga gaya yaitu, W adalah gaya berat benda, Fa adalah gaya *archimedes,* dan N adalah gaya normal bidang. Dalam keadaan seimbang maka $W = N + F_a$ sehingga dapat diturunkan rumus seperti pada Persamaan 2.9,

$$\begin{aligned} W &> F_a \\ m.g &> \rho ZC.V_b.g \\ \rho b.Vb.g &> \rho ZC.V_b.g \\ \rho b &> \rho ZC \end{aligned} \tag{2.9}$$

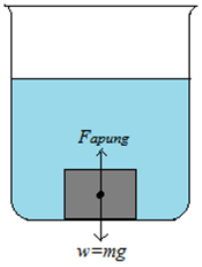
dengan,

ρb : massa jenis benda

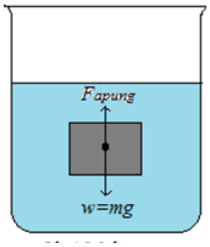
ρZC : massa jenis zat cair.

2. Benda Melayang

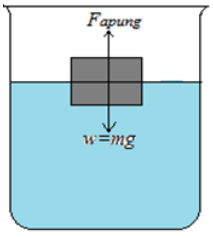
Benda dikatakan melayang apabila benda tersebut berada di antara posisi di bawah permukaan air dan di atas dasar. Posisi benda melayang dapat dilihat pada Gambar 2.10. Pada benda melayang terdapat dua gaya yaitu, Fa dan W. Dalam keadaan seimbang maka dapat diturunkan rumus pada Persamaan 2.12.



Gambar 2.9 Benda Tenggelam



Gambar 2.10 Benda Melayang



Gambar 2.11 Benda Terapung

$$W > F_a$$
$$\rho b.Vb.g > \rho ZC.V_b.g$$
$$\rho b = \rho ZC \quad (2.11)$$

3. Benda Terapung

Benda terapung dalam zat cair apabila posisi benda sebagian muncul di permukaan zat cair dan sebagian terbenam dalam zat cair. Posisi benda terapung dapat dilihat pada Gambar 2.11. Pada benda terapung terdapat dua gaya yaitu, Fa dan W. Dalam keadaan seimbang maka dapat dilihat pada Persamaan 2.12.

$$W = F_a$$
$$\rho b.Vb.g = \rho ZC.V_b.g$$
$$\rho b.Vb = \rho ZC.V2$$
karena $Vb > V2$ maka : $\rho b < \rho ZC$ (2.12)

Penerapan Hukum Archimedes untuk menentukan massa jenis benda dapat dilihat pada Persamaan 2.15,

$$\rho_{benda} = \frac{m}{V_{benda}} = \frac{m}{V_{air}}$$
$$\rho_{benda} = \frac{m}{m-m_s} x \rho_{air} \text{ karena } \rho_{air} = \frac{m-m_s}{\rho_{air}} \quad (2.13)$$

dengan,

V_{air} : volume air yang dipindahkan
m : massa benda di udara
ms : massa semu benda (di air)
ρ_{benda} : massa jenis benda
ρair : massa jenis air.

2.9 Kamera

Kamera pada penelitian berfungsi sebagai pengindraan atau sensor *vision* pada ROV. Terdapat banyak jenis kamera yang bisa dipakai dalam penelitian ROV antara lain, kamera jenis *actioncam, webcam,* DSRL, dan lainnya.

Gambar 2.12 Kamera Webcam

Penelitian ini menggunakan kamera berjenis webcam Logitech dengan seri C525 pada ROV. Pemilihan kamera webcam ini dikarenakan bentuk ukuran yang kecil dan ringan. Dengan ukuran yang kecil kamera ini mempunyai ukuran pixel yang besar hingga 8 Mp, maximum rate 30 fps, dan resolusi 720 pixel. Gambar 2.12 menunjukan kamera yang digunakan pada penelitian ini.

2.10 Motor DC

Motor DC adalah suatu alat atau mesin yang bisa mengubah energi listrik menjadi gerak. Berbagai macam jenis motor yang ada seperti motor AC, motor DC, Motor *Stepper* dan lainnya. Pada teknologi ROV ini banyak digunakan jenis motor DC karena dalam sistem daya, motor DC lebih praktis dibandingkan dengan motor AC.

Perancang ROV yang dapat diteliti menggunakan motor DC yang biasanya digunakan sebagai pompa air pada kapal-kapal yang bocor. Motor DC buatan *bilge pump,* ditunjukkan pada Gambar 2.13 merupakan motor DC yang dinilai memenuhi standar dalam pembuatan ROV. Motor DC memiliki daya dorong atau *thrust* yang cukup besar dengan sumber daya 12 VDC. Degan bantuan *propeller* atau baling-baling jenis *Clockwise* (CW) dan *Control Clockwise* (CCW) motor dapat mendorong air kearah luar dan dalam.

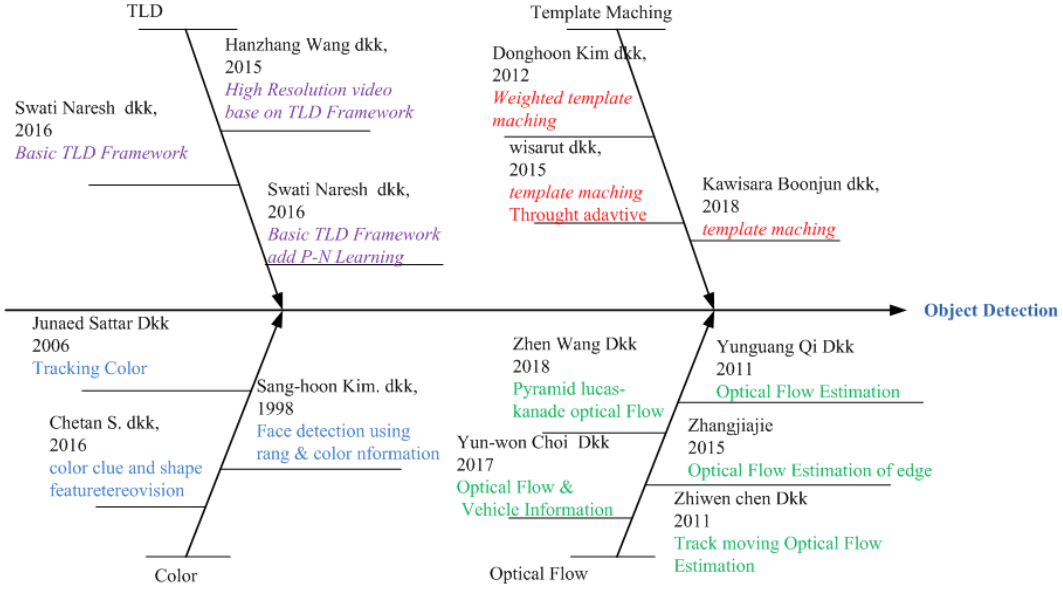
2.11 Diagram Fishbone

Fisbone adalah suatu bentuk diagram tulang ikan yang bertujuan untuk mengetahui sejarah atau penelitian yang telah dilakukan sebelumnya. Pada bagian depan atau kepala merupakan topik dari suatu penelitian yang akan kita lakukan.

Selanjutnya pada bagian tulangnya terdiri kategori atau bagian bagian dari penelitian yang telah dilakukan berdasarkan pembahasannya.

Pada Gambar 2.14 Diagram *Fishbone* ditunjukkan beberapa penelitian yang telah dilaksanakan berkaitan dengan *Object Detectio*n. Yang terdapat dalam beberapa kategori yaitu, *Template Maching*, *Optical Flow*, *TLD*, dan *Color fiture.*

Gambar 2.13 DC Motor *Bilge Pump*

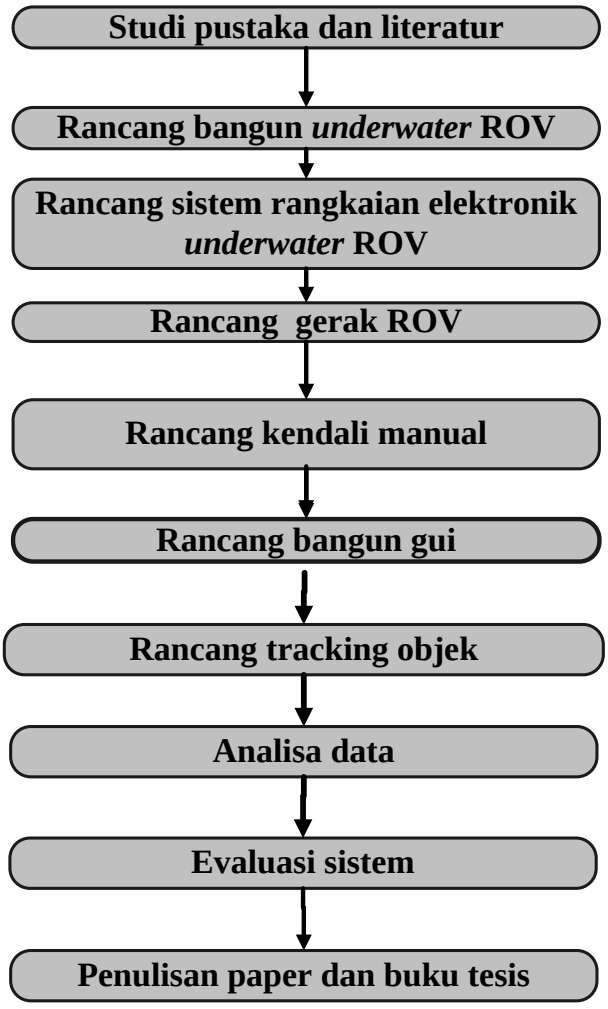


Gambar 2.14 Diagram *Fishbone*

Halaman Ini Sengaja Dikosongkan

BAB 3
METODOLOGI PENELITIAN

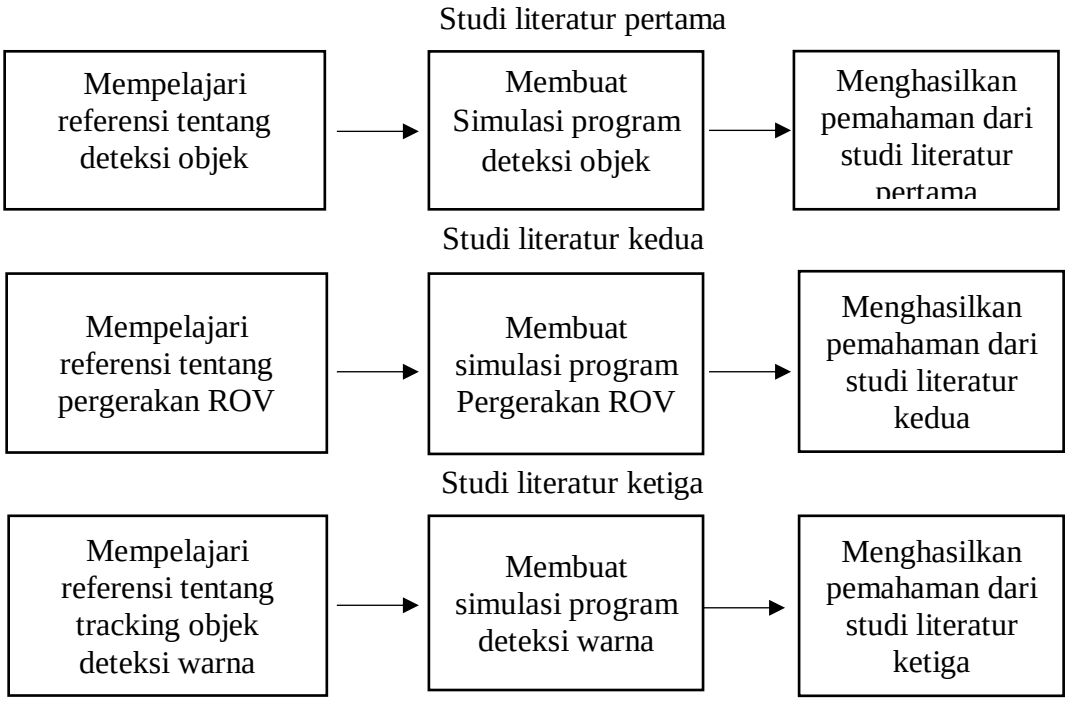
Pada penelitian ini dirancang metode *tracking object* pada *underwater* ROV dengan menggunakan metode deteksi warna. Penelitian ini memiliki beberapa tahapan yang dapat dilihat pada Gambar 3.1.



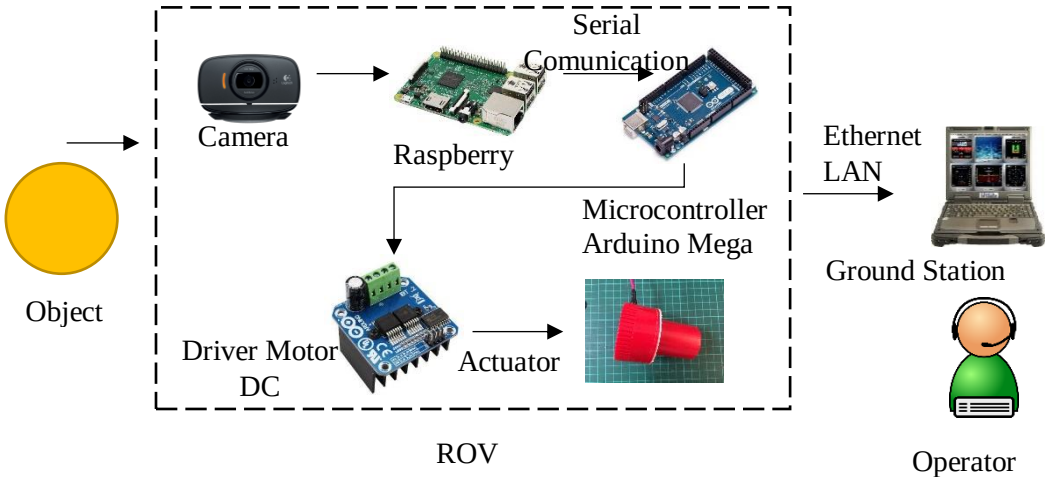
Gambar 3.1 Tahapan Penelitian

3.1 Studi Pustaka dan Literatur

Pada tahapan ini, penulis mencari dan membaca serta mempelajari dari beberapa pustaka dan literatur yang relevan dengan penelitian yang akan dikerjakan. Mempelajari dan membahas beberapa referensi dari *tracking oject*, dan pergerakan robot agar dapat dijadikan landasan berfikir dan permudah dalam mengerjakan penelitian dan pembuatan laporan tesis. Tujuan dari studi pustaka ini adalah sebagai dasar pengetahuan dalam melakukan penelitian dan pembuatan laporan tesis. Studi literatur ini juga dapat memberikan gambaran rancangan dari apa yang akan dilakukan nanti. Pada Gambar 3.2 menunjukan ilustrasi dari studi pustaka dan literatur.



Gambar 3.2 Ilustrasi Studi Pustaka dan Literatur



Gambar 3.3 Perancangan Sistem

3.2 Rancang Bangun Underwater ROV

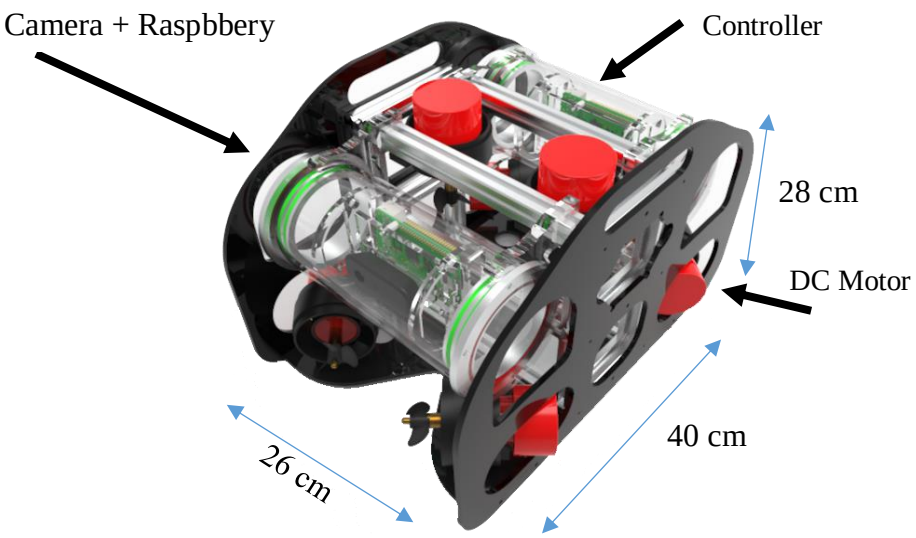
Pada tahap ini dilakukan rancang bangun *underwater* ROV yang ditunjukan pada Gambar 3.3. Operator adalah pengendali pusat yang dapat mengoperasikan semua sistem dari ROV tersebut. GUI pada komputer dapat memberikan informasi pada operator. Citra yang didapat melalui kamera akan diproses menggunakan *raspbery*. Sistem kontrol dari pergerakan *underwater* ROV ini terdiri dari mikrokontroler yang dapat meruskan pada kamera sebagai *first vision view* dan juga bagian dari perangkat pengolahan citra yang dapat diteruskan ke komputer untuk diolah.

3.2.1 Rancang Bangun Mekanik ROV

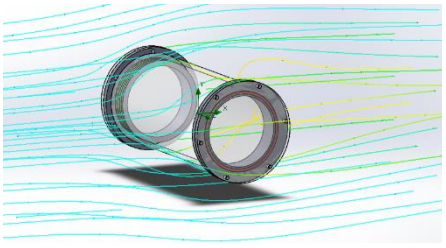
Dalam tahap ini dilakukan perancangan mekanik dari ROV dengan beberapa tahapan proses pembuatan. Perancangan tersebut dibagi beberapa tahapan-tahapan.

3.2.1.1 Desain Model *Underwater* ROV

Desain model *underwater* ROV ini menggunakan software 3D yang menghasilkan model ROV seperti yang ditunjukkan pada Gambar 3.4. *Underwater* ROV memiliki dimensi panjang ROV 40 cm, lebar 26 cm, dan tinggi



Gambar 3.4 Desain Model *Underwater* ROV



Gambar 3.5 *Flow Simulation* ROV Pada Tabung Pelampung

28 cm. Motor penggerak diletakkan pada posisi vertikal dan horizontal. Pelampung berada di bagian atas depan dan belakang. Pada desain *modeling* ini juga dapat melakukan *flow simulation* yang ditunjukkan pada Gambar 3.5. *Flow simulation* ini berfungsi sebagai simulasi bagaimana ROV mendapatkan gaya dorong ketika berjalan maju dan berjalan mundur.

3.2.1.2 *Frame* ROV

Frame dari ROV ini memiliki dua bagian yaitu, sisi atas dan bawah. Pada sisi atas difokuskan pada *buoyancy* dan sisi bagian bawah khusus pada peletakan motor penggerak. Desain dua sisi dapat mempermudah pemeliharaan dan perbaikan.

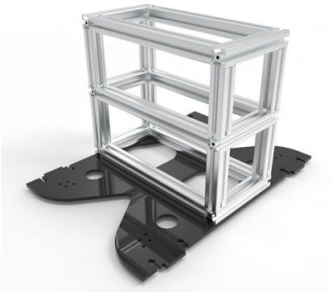
Gambar 3.6 ROV Frame

Pada bagian tengah ROV menggunakan alumunium profil 2 cm. pada bagian *base* samping dan bawah ROV digunakan bahan *acrylic* dengan tebal 5 mm. Bahan *acrylic* digunakan karena perhitungan biaya yang murah dan cukup kuat digunakan sebagai *frame*. Dengan bantuan CNC sebagai alat potong *acrylic* maka terbentuk dasar *frame* seperti Gambar 3.6.

Langkah awal pada pembuatan *frame* ROV ini adalah menggabungkan potongan alumunium yang telah ditentukan ukurannya sehingga terbentuklah *frame* alumunium persegi seperti yang ditunjukkan pada Gambar 3.7. Selanjutnya untuk peletakan motor penggerak dibutuhkan *base* dasar berupa *acrilic* seperti ditunjukan Gambar 3.8.

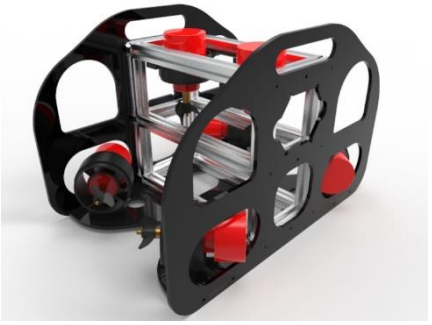
Peletakan motor penggerak yang diletakan pada posisi *base* bawah dengan menggunakan *mounting* yang terbuat dari bahan 3D printer maka terlihat seperti Gambar 3.9. *Finising* dari *frame* keselurhan dari perakitan bodi dengan menambahkan bodi samping yang terbuat dari bahan *acrilic*. Sehingga bentuk awal *frame* menyerupai bentuk yang ditunjukkan Gambar 3.10. Dari pengujian *frame underwater* ROV ini pernah diujikan sampai kedalaman kurang lebih 10 m.

Gambar 3.7 Frame Alumunium



Gambar 3.8 Frame dan Base Bawah

Gambar 3.9 Pemasangan Motor pada Frame ROV



Gambar 3.10 Frame Keseluruhan

3.2.1.3 Pelampung (*Buoyancy*)

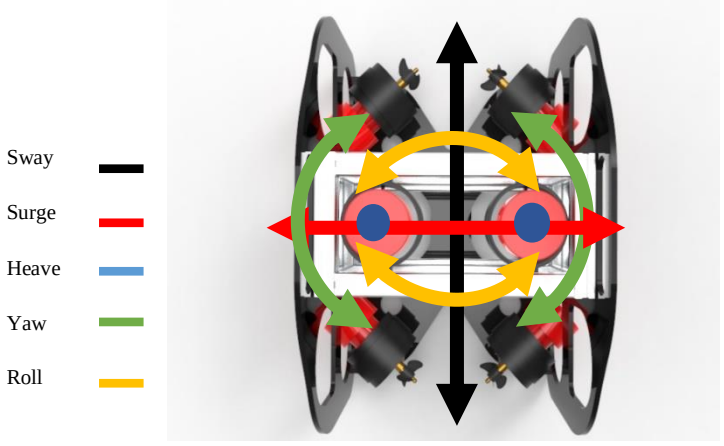
Untuk mengimbangi berat dari *frame* dan komponen, maka diperlukannya pelampung agar ROV tersebut memiliki daya apung netral. Pada *underwater* ROV dipasang 2 buah pelampung yang diletakkan pada sisi depan dan belakang. Ukuran masing-masing pelampung adalah 20 cm. Peletakan tabung pelampung dilihat pada Gambar 3.11.

Gambar 3.11 Penempatan Tabung pada Frame

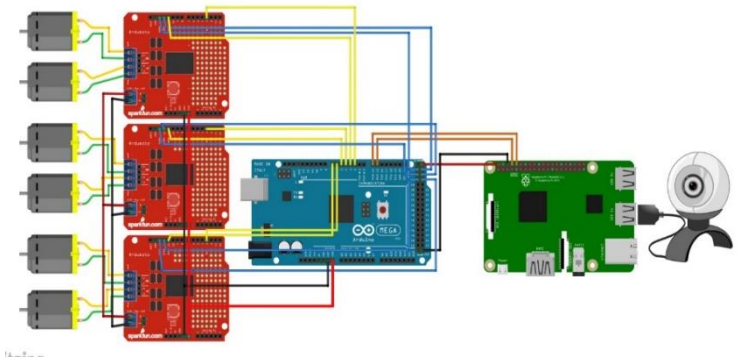
Pelampung ini terbuat dari bahan *acrylic* yang berbentuk tabung dengan diameter 100 mm. Fungsi dari tabung pelampung ini juga sebagai tempat komponen utama dari ROV tersebut. Bagian belakang terisi *microcontroller* yang merupakan inti pada ROV, dan bagian depan terdiri dari *microprocessor* dan kamera.

3.2.1.4 *Propulsion*

Propulsion merupakan gaya dorong yang dihasilkan oleh putaran baling-baling yang digerakan oleh motor. *Underwater* ROV ini memiliki enam buah motor DC yang terdiri dari *propeller clock wise* (CW) dan *counter clock wise* (CCW). *Propeller* yang digunakan terbuat dai bahan PLA yang dicetak dengan menggunakan 3D *Printing*. Empat buah motor di sisi bawah dipasang horizontal dan diatur deng sudut 40°. Dorongan dari keempat motor menghasilkan gerakan ROV kesegala arah. Selanjutnya pada dua motor berikutnya disusun pada sumbu vertikal dan menghasilkan gerak *heave* atau *roll*. Total pola gerakan yang dihasilkan motor tersebut berupa 5 derajat kebebasan yang ditunjukan pada Gambar 3.12.



Gambar 3.12 ROV dengan 5 Derajat Kebebasan



Gambar 3.13 Sistem Rangkaian Elektronik *Underwater* ROV

3.3 Rancang Sistem Rangkaian Elektronik *Underwater* ROV

Perancangan elektronik *underwater* ROV ini menggunakan 1 buah mikrokontroler Arduino Mega sebagai pengolah gerakan ROV. Sedangkan dalam pengolahan *image* menggunakan Raspberry Pi 3 sebagai mikroprosesornya. Untuk mengatur putar balik motor menggunkan *driver* motor tipe Monster moto. Sistem rangkaaian keseluruhan terlihat pada Gambar 3.13.

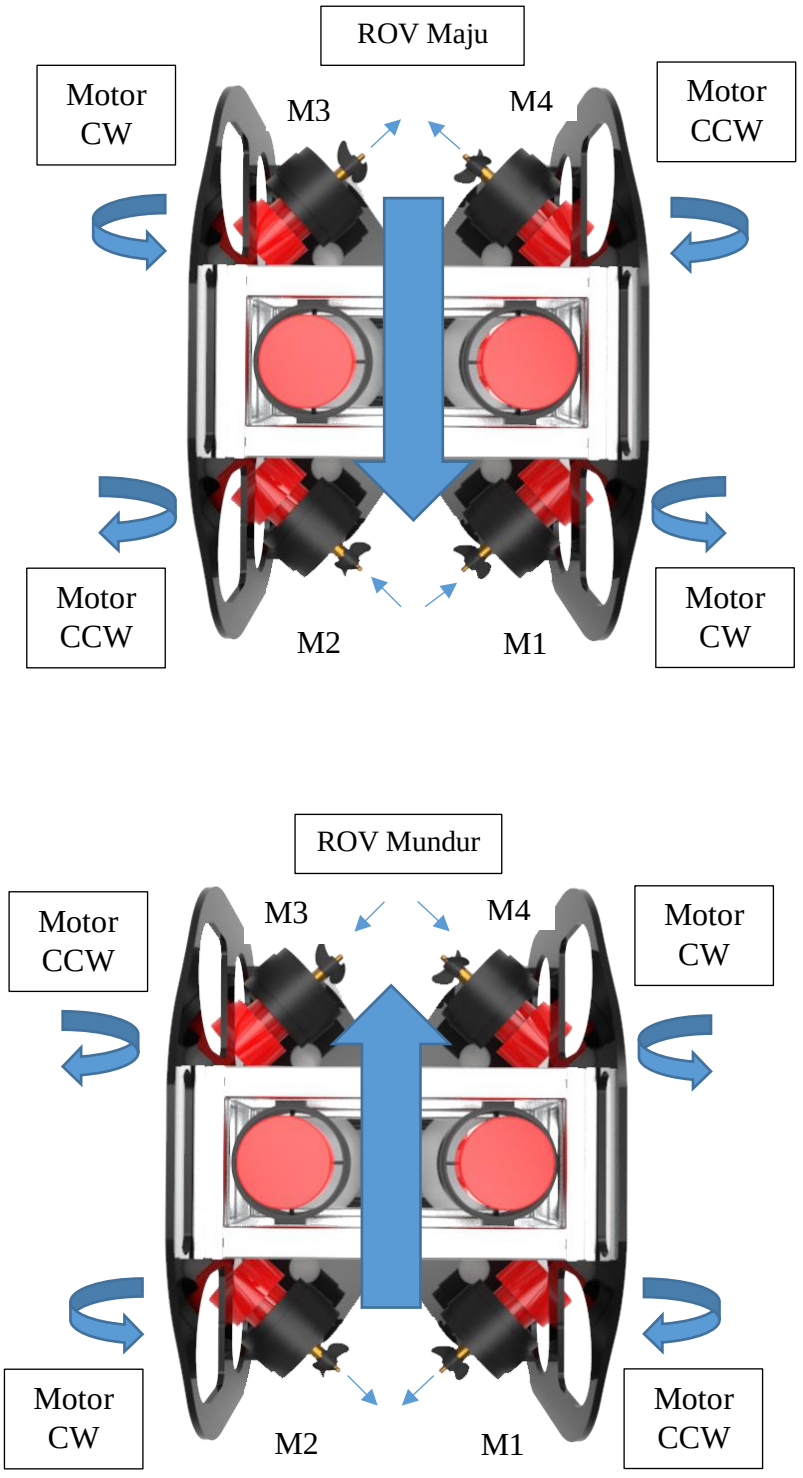
3.4 Rancang Gerak ROV

Pada perancangan sistem gerak ROV ini, dilakukan dengan pengendalian motor DC yang digunakan. *Underwater* ROV yang dibuat menggunakan 6 buah motor yang terdiri dari 2 buah motor yang dipasang vertikal dan 4 buah motor lagi dipasang horizontal. Untuk mendapatkan gerakan yang dinamis, maka pada motor DC kanan dan kiri tersebut dipasang *propeller* CW dan CCW. Berikut tahapan gerakan ROV dapat disimulasikan dari beberapa gambar berikut:

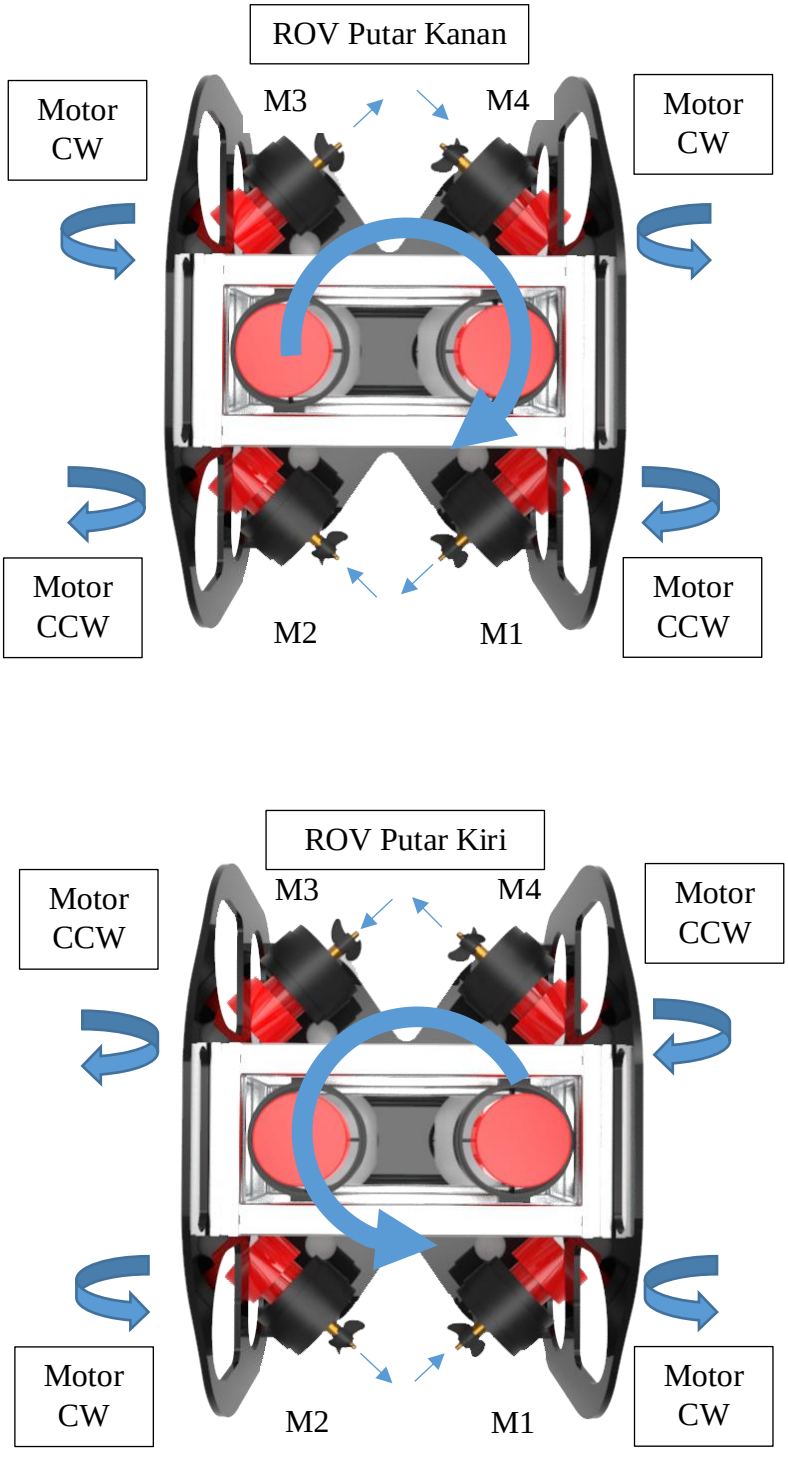
1. Gerakan maju dan mundur

   Pada gerakan maju dan mundur, ini motor yang dipasang *horizontal* akan saling bergerak berlawanan antara motor depan dan belakang . Jika gerak maju motor 1 dan 3 berputar CW untuk motor 2 dan 4 berputar CCW. Jika gerak mundur akan sebaliknya, diilustrasikan pada Gambar 3.14 motor 1 dan 3 berputar CCW, untuk motor 2 dan 4 berputar CW.

.



Gambar 3.14 Gerak Maju dan Mundur



Gambar 3.15 Gerak Kanan dan Kiri

2. Gerakan putar kanan dan putar kiri

Pada pola gerakan putar kanan dan kiri, motor 1 dan motor 2 akan berputar CCW bersamaan dengan motor 3 dan 4 akan berputar CW maka akan dihasilkan

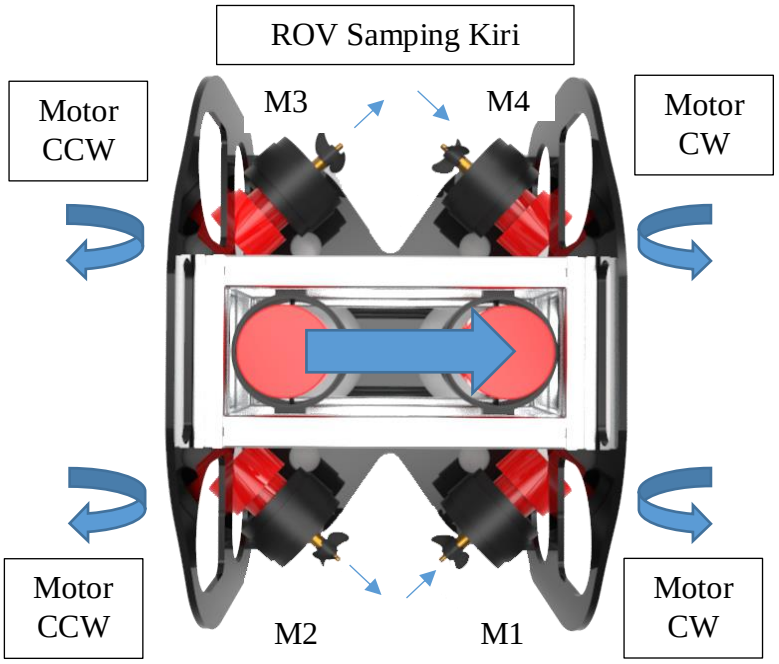
gerakan putar kanan dan pola selanjutnya motor 1 dan 2 berputar CW, motor 3 dan 4 berputar CCW akan menghasilkan gerakan sebaliknya yaitu putaran kekiri seperti Gambar 3.15.

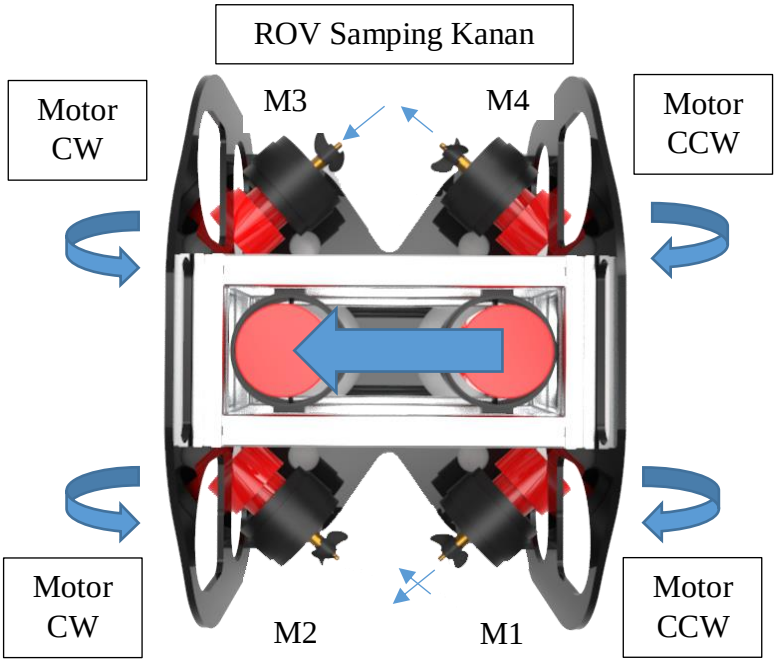
3. Gerakan samping kanan dan samping kiri

Pola gerak samping kiri ditentukan dengan pola gerak motor 1 dan motor 4 berputar CW sedangkan motor 2 dan 3 bergerak CCW. Untuk sebaliknya gerakan kesamping kanan maka motor 1 dan 4 berputar CCW sedangkan motor 2 dan 3 bergerah CW seperti Gambar 3.16.

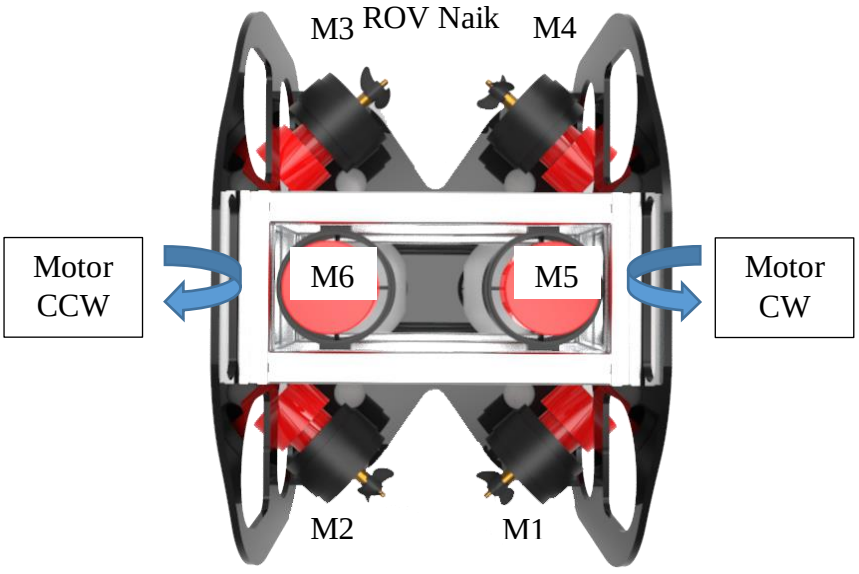
4. Gerakan naik dan turun

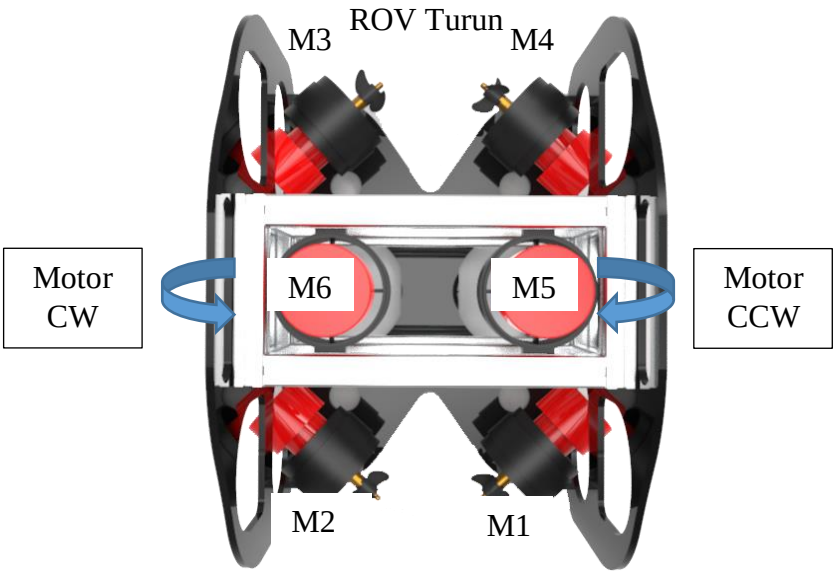
Untuk gerakan ke atas dan ke bawah, motor yang terpasang vertikal akan bergerak berlawanan sama seperti pada gerakan maju dan mundur. Pergerakan motor dapat dilihat pada . Motor 5 CW dan motor 6 CCW maka ROV naik, motor 5 CCW dan motor 6 CW maka motor turun.



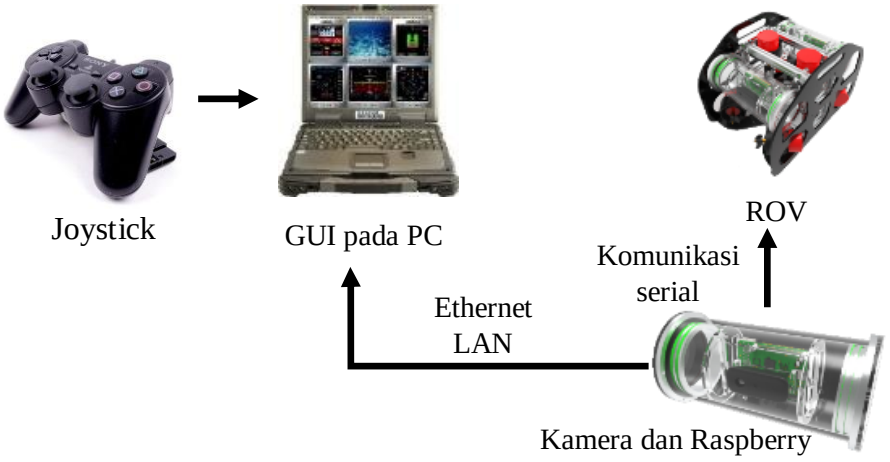


Gambar 3.16 Geser Kanan dan Kiri





Gambar 3.17 Gerak Naik dan Turun



Gambar 3.18 Rancang Sistem Kendali Manual

3.5 Rancang Kendali Manual

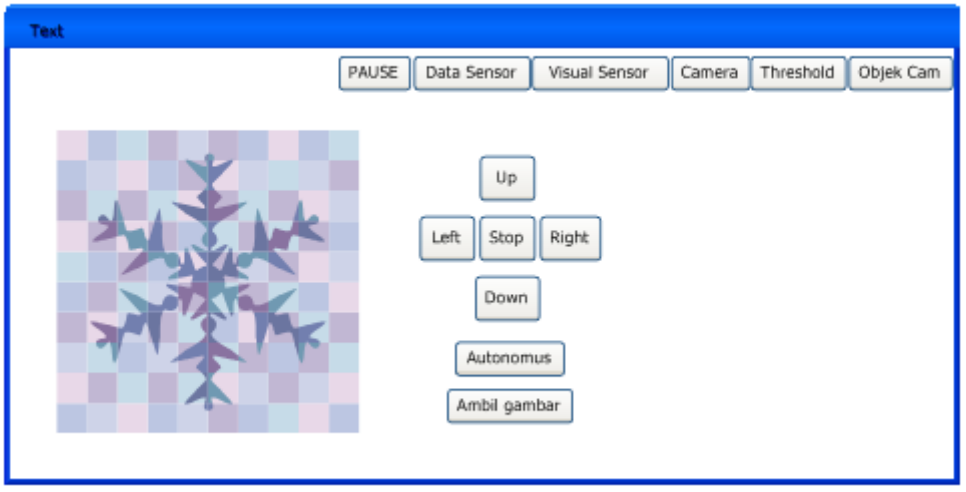
Sistem kendali manual ini diharapkan ketika mengetahui ada korban yang tenggelam, ROV segera diterjunkan kedalam air. Pembuatan sistem kendali manual ini adalah langkah awal atau tindakan pencarian sebelum memasuki proses pendeteksian dan *tracking object* yang dimaksudkan. Rancang sistem manual ini melibatkan beberapa operator, *joystik* sebagai kendali dan PC sebagai fungsi

ground station. Operator akan melihat objek di sekeliling ROV melalui tampilan GUI pada PC dan mengendalikannya berdasarkan penglihatannya. Gambar 3.18 menunjukkan diagram dari rancang sistem kendali manual. *Joystick* yang terhubung ke PC akan diteruskan ke ROV dengan menggunakan komunikasi serial. Sehingga perintah gerakan ROV akan langsung diterjemahkan ke dalam sinyal digital yang akan memerintahkan ROV tersebut bergerak.

3.6 Rancang Bangun GUI

Perancangan perangkat lunak ini meliputi proses yang membentuk sebuah visualisasi atau tampilan visual berupa GUI (*Graphical User Interface*) tampilan sistem kerja pada ROV seperti Gambar 3.19. GUI ini digunakan pusat informasi, pengolahan data dan sebagai *First Person View* (FPV). Dalam perancangan ini hasil dari pengolahan data yang menggunakan kamera akan tampil kedalam GUI menggunakan *web*. Visual C++ sebagai program utama dan penggunaan Open CV sebagai *library* pengambilan, pengelola dan penampilan data gambar.

Desain interface dilengkapi dengan beberapa komponen penting, 6 buah menu (*Pause*, data sensor, visual sensor, *camera*, *threshold*, objek cam). Dan 6 buah tombol (*up, down, left, right, stop, autonomus*, dan ambil gambar). Adapun fungsi masing masing komponen antara lain:

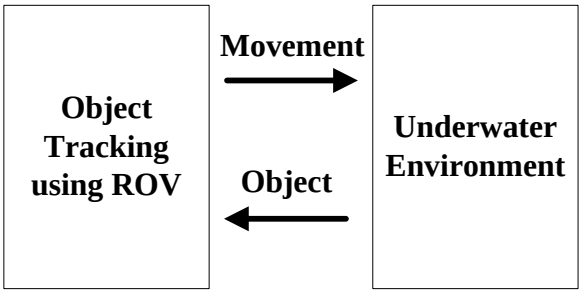


Gambar 3.19 Tampilan GUI

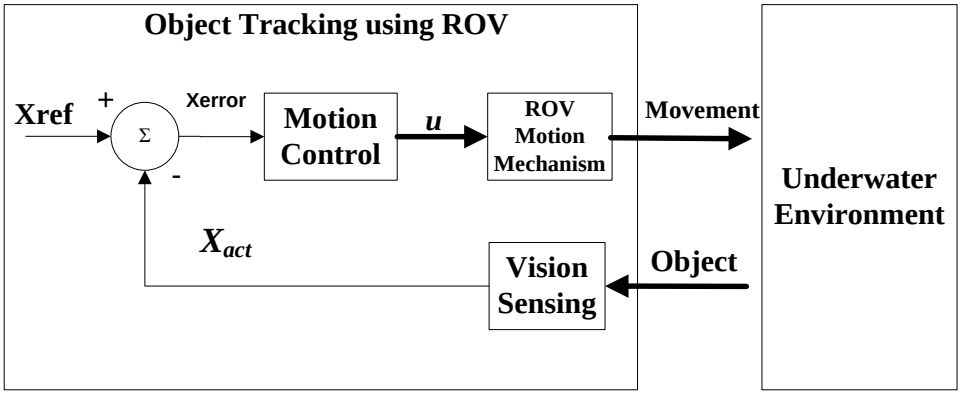
1. **Menu *PAUSE*,** digunakan untuk peintah berhenti keseluruhan.
2. **Menu Data Sensor,** digunakan untuk menampilkan data sensor.
3. **Menu Visual Sensor,** digunakan untuk menampilkan data visual.
4. **Menu *Camera*,** digunakan untuk menampilkan gambar dari camera.
5. **Menu *Threshold*,** digunakan untuk menampilkan gambar threshold.
6. **Menu Objek Cam,** digunakan untuk menampilkan gambar yang sudah terdeteksi objek.
7. **Tombol *Up*,** digunakan sebagai perintah ROV bergerak keatas.
8. **Tombol *Down*,** digunakan sebagai perintah ROV bergerak kebawah.
9. **Tombol *Left*,** digunakan sebagai perintah ROV bergerak kekiri.
10. **Tombol *Right*,** digunakan sebagai perintah ROV bergerak kekanan.
11. **Tombol *Autonomus*,** digunakan sebagai perintah ROV gerak automatis.
12. **Tombol Ambil Gambar,** digunakan sebagai perintah capture gambar.

3.7 Rancang *Tracking* Objek Menggunakan ROV

Sistem otomatis pada *underwater* ROV ini menggunkan kamera sebagai sensor penglihatan. Di dalam lingkungan bawah air, precobaan ini diketahui objek yang akan diambil itu seperti objek yang berwarna merah dan cream (warna kulit). Dalam perancangan *tracking* objek, *underwater* ROV menggunakan 1 buah *micropocessor* yaitu raspbery pi. Raspbery pi ini digunakan untuk mengolah data citra. Dan 1 buah mikrokontroler yaitu arduino mega, yang digunakan sebagai kembali kontrol *movement* ROV tersebut. Gambar 3.20 menunjukkan diagram blok keselurah dari sistem ROV yang dikerjakan. *Underwater* ROV ini akan bergerak otomatis dengan cara mengikuti warna yang telah dideteksi. Gambar 3.21 menunjukkan diagram proses tracking. Dalam proses ini terdapat subproses yaitu proses *vision sensing*, dimana proses ini akan dilakukan pengolahan citra. Selanjutnya subproses *motion control* yang merupakan sistem kontrol gerakan ROV. Subproses terakhir adalah *ROV motion Mechanism*, proses pergerakan atau perputaran motor yang menggerakan ROV kearah sesuai kontrol yang terbentuk.



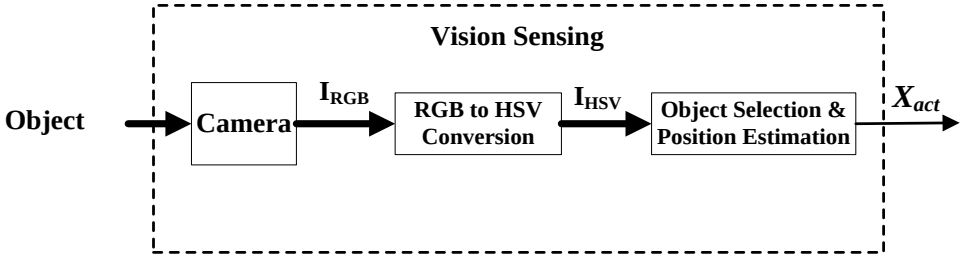
Gambar 3.20 Diagram Blok Keseluruhan



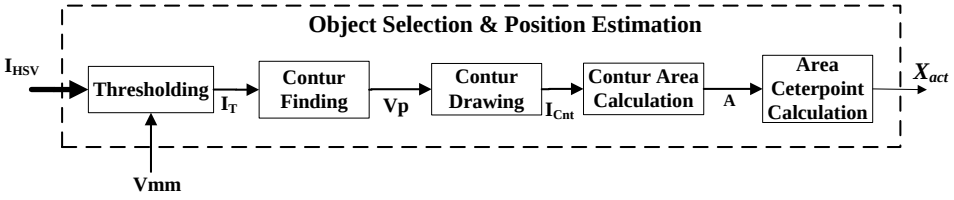
Gambar 3.21 Diagram Blok Object Tracking using ROV

3.7.1 Vision Sensing

Seluruh proses pengolahan citra akan dilakukan pada proses *vision sensing* ini. Pada proses ini dilakukan dengan menggunakan *raspbery pi* sebagai mikroprosessor pengolah citra. Terdapat beberapa tahapan anatara lain, objek yang tertangkap oleh kamera akan dikonversikan kedalam HSV, dimana sebelumnya gambar berbentuk RGB. Selanjutnya dilakukan proses seleksi objek dan menentukan posisi tengah objek tersebut.



Gambar 3.22 Blok Diagram *Vision Sensing*



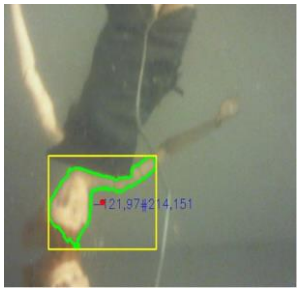
Gambar 3.23 Blok Diagram *Object Selection dan Position Estimation*

3.7.2 Seleksi dan Perkiraan Posisi Objek

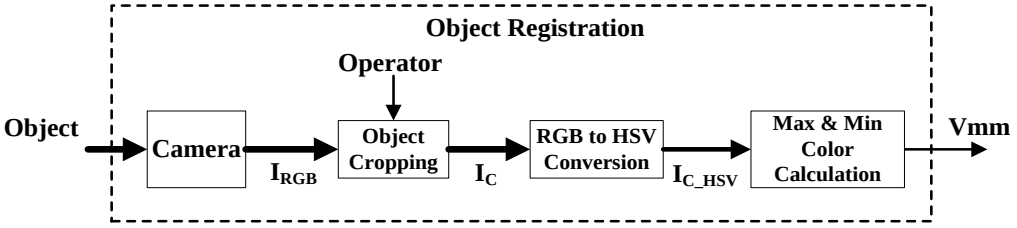
Proses seleksi objek ini dilakukan untuk menbedakan benda yang dideteksi dengan benda yang lainnya. Warna dari benda yang dideteksi akan dipotong untuk mendapatkan nilai minimum dan maksimum dari HSV (Vmm) yang di jadikan parameter dalam melakukan *threshold*. Ketika nilai *threshold* dari benda yang menjadi target diperoleh, selanjutnya mencari bentuk kontur dari benda tersebut.

Untuk menentukan kontur dari benda target (*find countour*), dilakukan dengan menggunakan *metode contour approximation simple* dan mode *contour retieval list*. Setelah melalui tahapan ini, benda yang menjadi target akan digambarkan berdasarkan countour yang di dapatkan.

Dari bentuk *contour* yang diketahui maka akan didapatkan ukuran tinggi dan lebar dari benda yang dideteksi. Serta didaptkan juga posisi x_1 dan x_1 yang semuanya disebutkan dalam vektor A, dimana vektor A terdiri dari A = [x y W H]. Selanjutnya akan ditentukan *center of points* sebagain nilai x_{actual} yang akan jadi referensi pergerakan ROV tersebut. Blok diagram pemilihan objek ditunjukan pada Gambar 3.23. Gambar 3.24 merupakan simulasi hasil akhir yang akan melihatkan info dari pencarian nilai x, y, W dan H. Dari gambar ini akan terbentuklah suatu *contour* dari objek yang akan terdeteksi dan menentukan tinggi serta lebar *contour*. Sehingga didapat pula nilai x dan y sebagai *center point* pendeteksian objek.



Gambar 3.24 Simulasi Pencarian nilai A[x y W H]



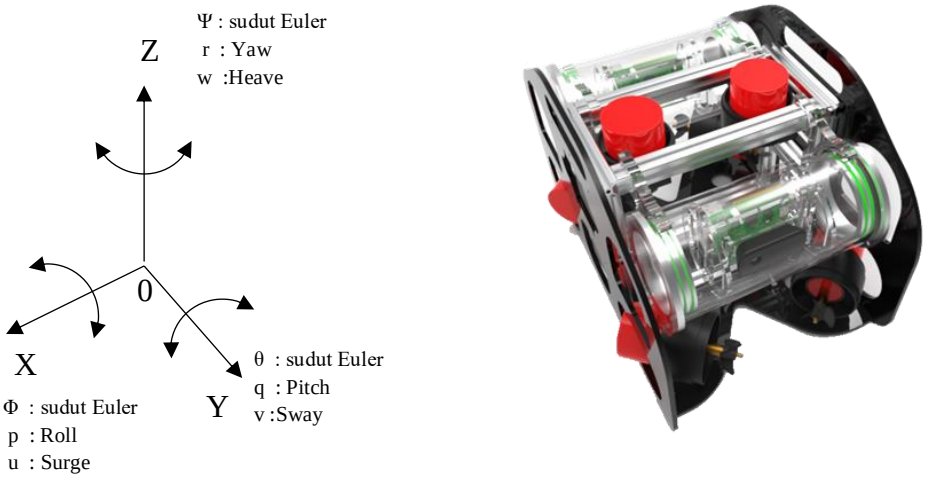
Gambar 3.25 Blok Diagram *Object Registration*

3.7.3 Object *Registration*

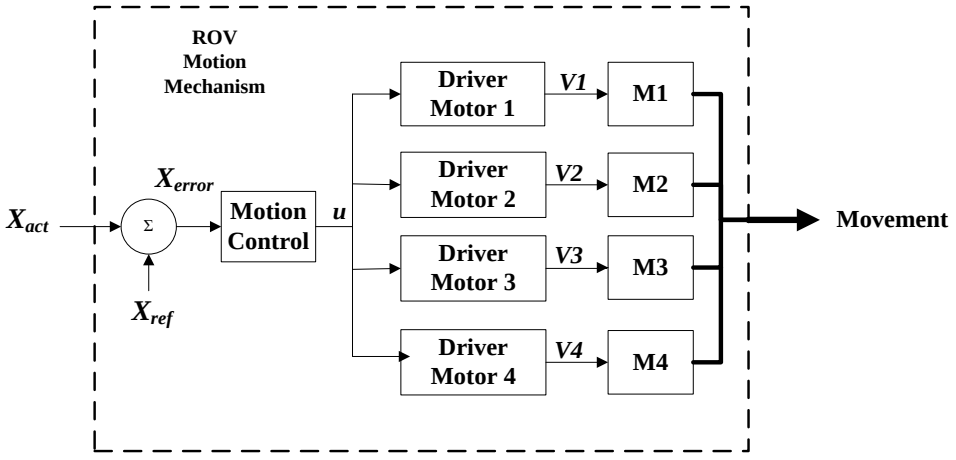
Object registration merupakan langkah pertama untuk menentukan objek mana yang akan dipilih sebagai objek terdeteksi. Objek awal berupa gambar RGB yang dipotong sehingga hanya mendapatkan objek dari gambar tersebut. Dari hasil potongan tersebut maka dapat dikonversikan dari RGB ke HSV. Dari hasil konversi ini dapat didapatkan nilai minimum dan masimum warna dari objek tersebut yang dapat dijadikan reverensi pada *tracking* object. Gambar 3.25 menunjukan proses *registration.*

3.7.4 *Motion* ROV

Perancang posisi dari ROV menggunakan prinsip kendali x, y, dan z. Sumbu x dapat menentukan posisi atau gerak ROV ke arah kanan dan kiri searah sumbu x yang disebut gerak *surge*. *Roll* yang merupakan rotasi terhadap sumbu x. Sumbu y terdapat pergerakan arah maju dan mundur yang disebut *sway.* Terdapat pula posisi yang merupakan pergerakan rotasi terhadap sumbu y yang disebut gerakan *pitch*. Untuk posisi selanjutnya, pergerakan ROV terhadap sumbu z. Pada sumbu z, posisi ROV akan membentuk gerak *heave* atau posisi saat ROV berada di atas dan di bawah searah sumbu z. Rotasi dari sumbu z juga dapat dikondisikan untuk menentukan posisi *yaw.* Posisi ROV dilihatkan pada Gambar 3.26.



Gambar 3.26 Posisi *Underwater* ROV



Gambar 3.27 Blog Diagram Kendali Posisi

Blok diagram pada Gambar 3.27 menunjukkan alur kendali posisi pada ROV. Untuk menentukan *motion control* dari ROV, rumus yang digunakan dalam penelitian ini ditunjukkan pada persamaan 3.4,

$$u = \frac{(X_{act} - Pixel_{min})x(PWM_{maks} - PWM_{min})}{(Pixel_{max} - Pixel_{min}) + PWM_{min}} \quad (3.4)$$

Pada bidang kerja terhadap sumbu X, memiliki besar nilai target (*setpoint*) di posisi 240 px (Xref), dimana nilai tersebut merupakan hasil bagi dari besar *frame*

gambar pada sumbu X yang berukuran 480 px. Nilai target digunakan sebagai acuan posisi objek yang diinginkan.

Untuk menentukan *tracking* objek dibutuhkan kendali arah putar dan kecepatan motor agar dapat selalu berada pada posisi target. Nilai dari hasil posisi objek yang terdeteksi pada sumbu X merupakan nilai aktual dari objek (Xactual), maka digunakan Persamaan 3.4, yang nantinya digunakan sebagai nilai penentu kecepatan dan arah gerak motor.

Jika nilai Xactual > Xref, maka arah gerak ROV berputar searah jarum jam, sedangkan jika nilai Xactual < Xref maka arah gerak ROV berputar berlawanan jarum jam terhadap sumbu Z pada bidang kerja ROV.

Batasan untuk kendali motor CW ialah dengan *range* piksel minimum 160 dan piksel maximum 320 dengan nilai minimum piksel itu adalah 0 dan piksel maximum itu adalah 255. Jadi untuk menentukan motor berpuitar CW apabila Xact lebih besar dari pada Xref dapat dilihat pada Persamaan 3.5.

ROV Rotate CW

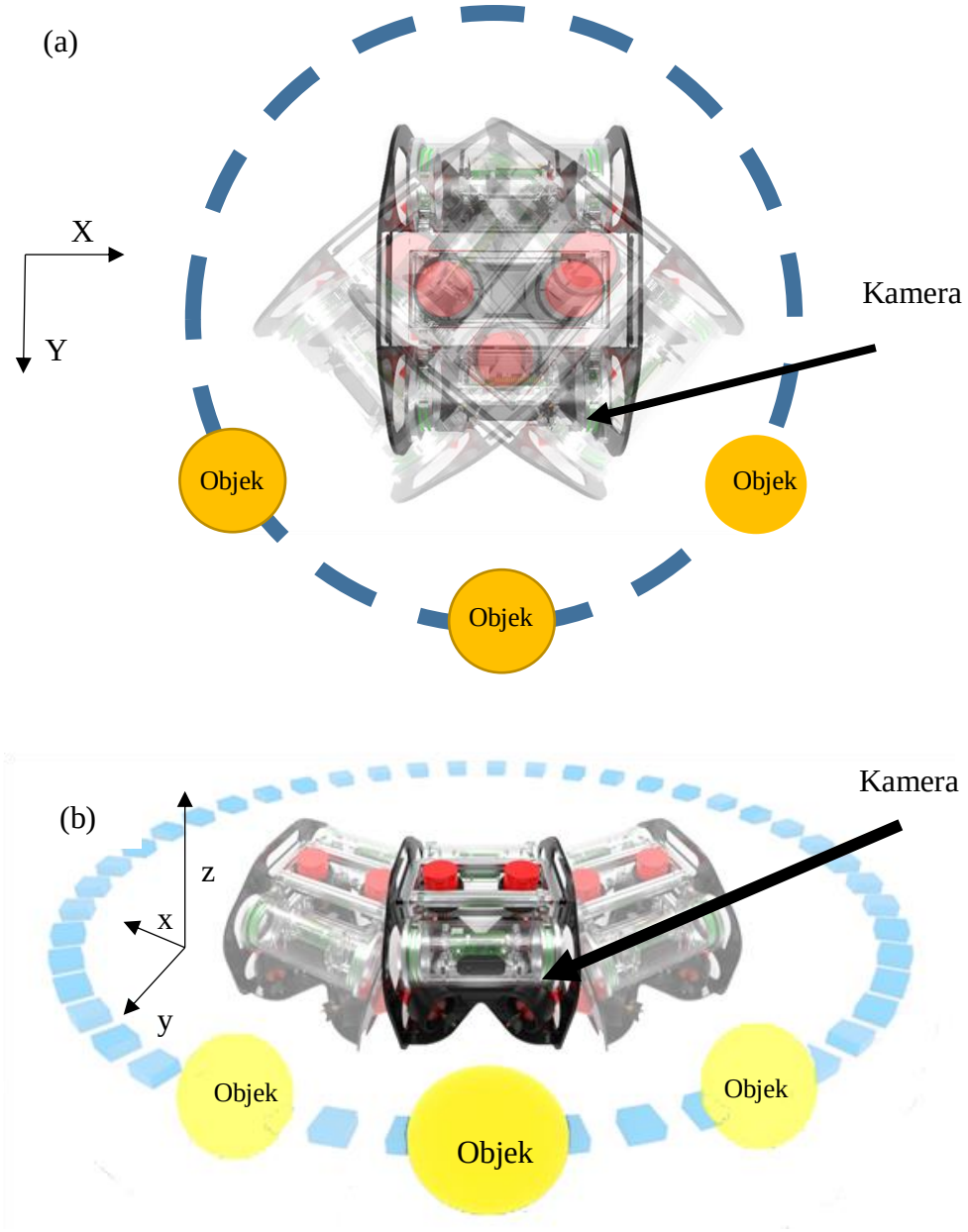
$$Speed = \frac{(X_{act}-160)x(255-0)}{(320-160)+0} \tag{3.5}$$

Batasan untuk kendali motor CCW diketahui nilai *range* piksel minimum 0 dan piksel maximum 160. Dengan piksel minimum pwm 0 dan piksel maxium 225. Maka untuk menentukan motor berputar CCW jika Xact lebih kecil dari Xref, Persamaan 3.6 menentukan kecepatan putar CCW,

ROV Rotate CCW

$$Speed = \frac{(X_{act}-0)x(255-0)}{(160-0)+0} \tag{3.6}$$

Gambar 3.28 menunjukkan ilustrasi pergerakan ROV terhadap objek, dimana ROV bergerak rotasi pada sumbu z terhada sumbu x.



Gambar 3.28 Ilustrasi Pergerakan ROV Terhadap Objek (a) Tampak Atas (b) Tampak Depan.

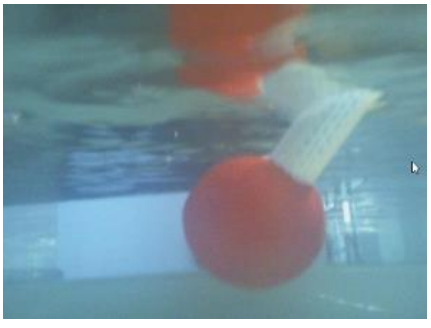
Halaman ini sengaja dikosongkan

BAB 4
HASIL DAN PEMBAHASAN

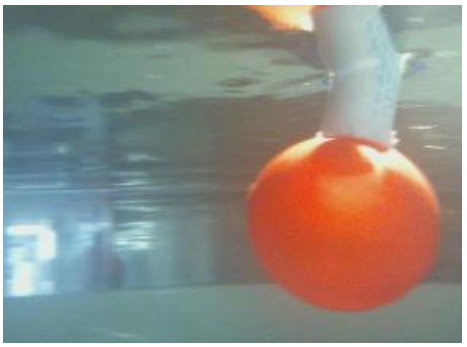
Pada bab ini berisi tentang pengujian dan analisa terhadap hasil perancangan dan pembuatan metode pada bab sebelumnya. Adapun pengujian yang dilakukan terdiri dari :

4.1 Pengujian Kamera

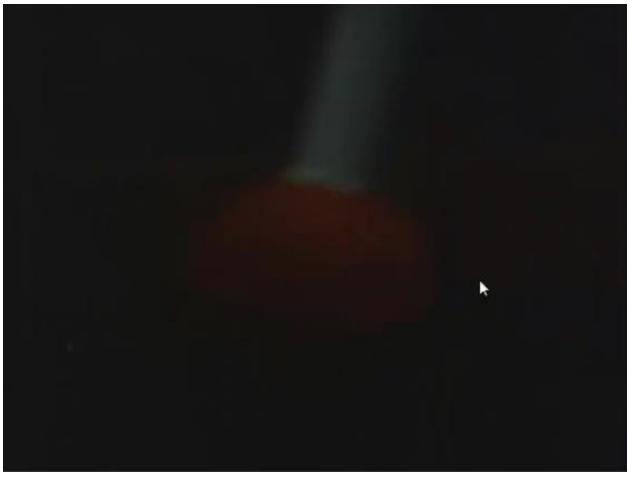
Pengujian kamera dilakukan di dalam air, dengan mencoba memasukan kamera ke dalam tabung yang kedap udara. Pengujian ini dilakukan dengan beberapa kondisi. Kondisi dalam keadaan tanpa cahaya dan dalam keadaan dengan cahaya. Cahaya yang digunakan berupa lampu LED 12 volt. Area percobaan kamera ini di lakukan dengan menggunakan akuarium di tempat yang memiliki cahaya baik dan di dalam kolam dengan lantai yang gelap. Percobaan pertama dilakukan pada akuarium yang menampilkan hasil seperti Gambar 4.1. Hasilnya terlihat jelas pada jarak dekat. Selanjutnya pada akuarium yang diberi cahaya dari lampu LED ditunjukan pada Gambar 4.2 maka objek yang terlihat lebih cerah. Ketika pada area kolam dengan lantai gelap yang ditunjukan pada Gambar 4.3 hasilnya objek tersebut terlihat tidak jelas. Terakhir percoban pada area lantai gelap dengan pencahayaan yang ditunjukan pada Gambar 4.4 dimana cahaya dari objek terlihat jelas dalam jarak dekat. Dari hasil pengujian tersebut diketahui bahwa pencahayaan sangat penting dalam pembuatan suatu ROV. Pencahayaan yang minim sangat menyulitkan operator dalam mengendalikan ROV tersebut.



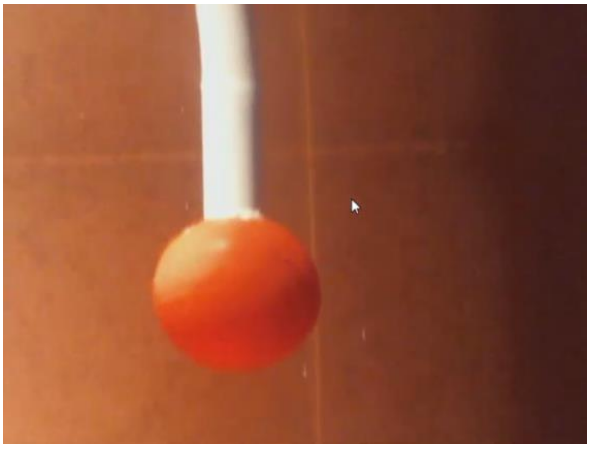
Gambar 4.1 Pengujian Kamera Pada Akuarium Tanpa Cahaya



Gambar 4.2 Pengujian Kamera Pada Akuarium Dengan Cahaya



Gambar 4.3 Pengujian Kamera Pada Kolam Tanpa Cahaya

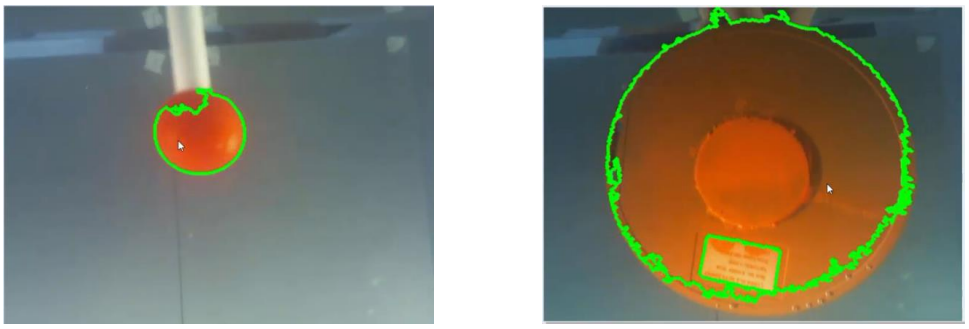


Gambar 4.4 Pengujian Kamera Pada Kolam Dengan Cahaya

4.2 Pengujian Objek

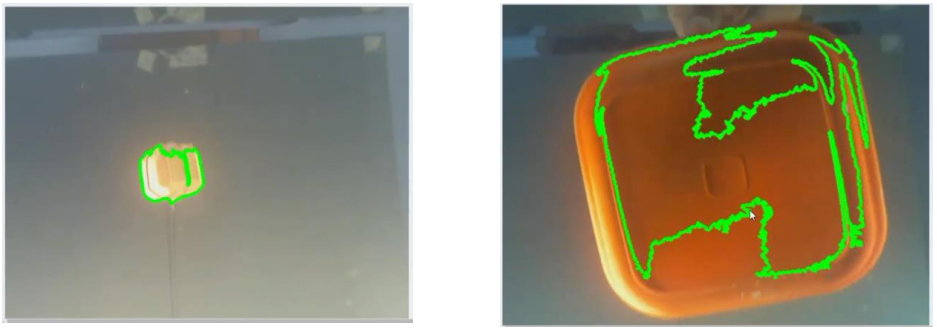
Pengujian objek ini dilakukan dengan beberapa bentuk dan ukuran. Objek yang digunakan ditunjukkan pada Gambar 4.5. Pertama dilakukan dengan objek berbentuk bola dengan ukuran diamerter 6 cm dan lingkaran besar dengan ukuran diameter 20 cm. Maka hasil pendeteksian akan ditunjukkan pada Gambar 4.6. Dilanjutkan pada benda persegi dengan ukuran masing-masing sisi 3 cm dan ukuran yang ditunjukkan pada Gambar 4.7, yang dapat menghasilkan objek yang terdeteksi. Selanjutnya dilakukan pada benda berbentuk tabung dengan panjang 17 cm ditunjukkan pada Gambar 4.8. Dari semua percobaan yang dilakukan pada jarak 30 cm didapatkan hasil batasan terkecil yang dicoba dengan ukuran 3 cm dan paling besar ukuran 20 cm dapat dilakukan pendeteksian.

Gambar 4.5 Objek Pengujian



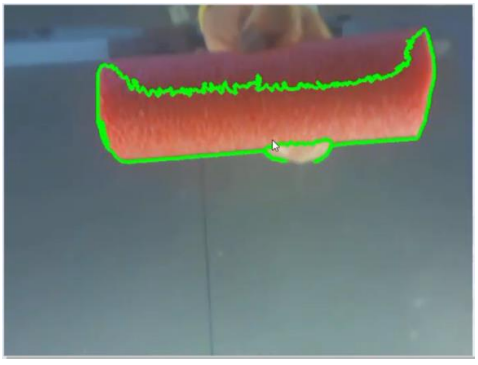
(a) (b)

Gambar 4.6 (a) Objek Bola Diameter 6cm (b) Objek Lingkaran Diameter 20cm



(a) (b)

Gambar 4.7 (a) Objek Persegi 3cm (b) Objek Persegi 17cm

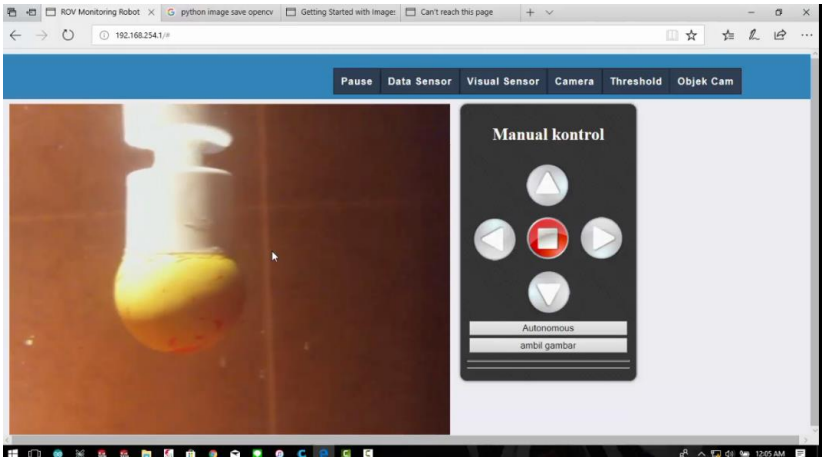


Gambar 4.8 Objek Berbentuk Tabung Dengan Panjang 17cm

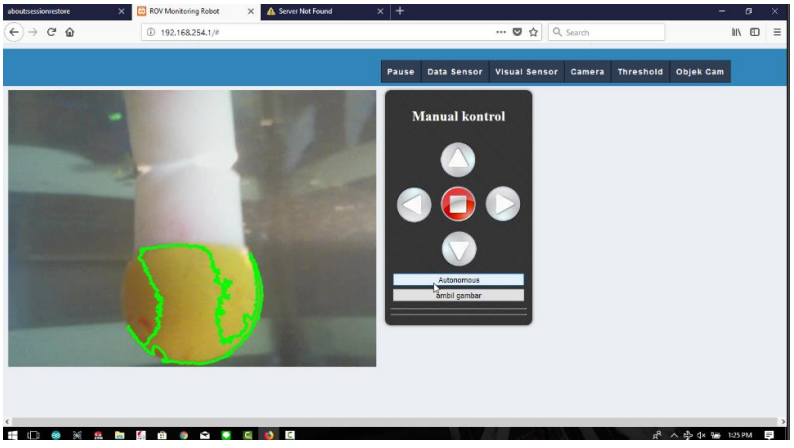
4.3 Pengujian GUI ROV

Graphical User Interface atau sering disebut GUI adalah hal terpenting dalam sistem operasi ROV. GUI berisi sejumlah informasi yang dapat membantu operator menggunakan ROV. Bagian yang terdapat dalam GUI adalah tampilan dari kamera, tombol kontroler, dan informasi lainnya berupa data-data. Pada GUI ini juga kita dapat melakukan beberapa aksi antara lain menggerakan ROV sesuai pada posisi yang kita inginkan dan sistem *tracking*.

Pengujian GUI dilakukan untuk mengetahui apakah GUI yang dibuat dapat melakukan tampilan kamera dan sistem *tracking* atau tidak. Gambar 4.9 menunjukan pengujian GUI saat menampilkan hasil dari kamera yang ada pada ROV. Selanjutnya pada Gambar 4.10 menujukkan tampilan GUI dalam *tracking* objek.



Gambar 4.9 Tampilan GUI Saat Menampilkan Hasil Video



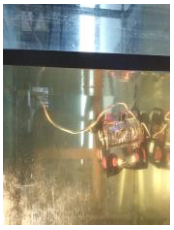
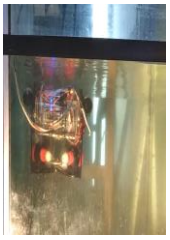
Gambar 4.10 Tampilan GUI Saat Menampilkan Hasil *Tracking*

4.4 Pengujian Kendali Manual

Kendali manual *underwater* ROV tehubung langsung pada GUI ROV. Terdapat 5 buah tombol perintah kendali manual. 4 buah tombol ini memiliki fungsi masing masing sesuai dengan namanya seperti perintah *UP*, *DOWN*, *RIGHT*, *LEFT*, dan *STOP*. Hasil dari percobaan ditampilkan pada Tabel 4.1. Dimana diketahui bahwa untuk penekanan tombol masing masing perintah dapat merespon dengan cepat. Rata-rata dalam 1 detik robot akan langsung bergerak sesuai perintahnya. Hanya saja ketika penekanan tombol stop atau berhenti, robot mendapatkan respon yang sangat lambat. Sekitar 15 - 46 detik robot baru akan berhenti. Lambatnya respon yang didapatkan ini dipengaruhi oleh pengiriman data dari kabel LAN PC yang dihubungkan ke *underwater* ROV tersebut.

Tabel 4.1 Percobaan Kendali Manual

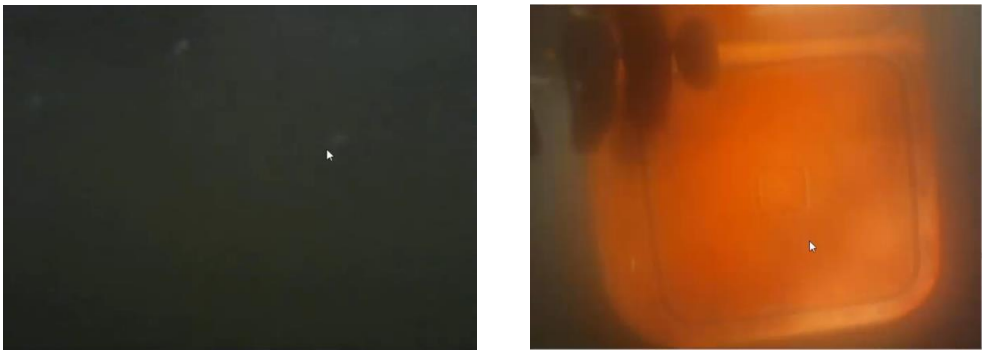
No	Tombol Perintah	Hasil	Kecepatan Aksi Reaksi	Tombol Berhenti	Keterangan
1	Up 		2 detik	46 detik	Robot berhasil bergerak naik dalam waktu 2 detik setelah penekanan tombol UP tetapi butuh waktu sekitar 46 detik untuk berhenti.
2	Down 		1 detik	32 detik	Robot berhasil bergerak turun dalam waktu 1 detik setelah penekanan tombol Down dan membutuhkan waktu sekitar 32 detik untuk berhenti.

3	Right 		1 detik	17 detik	Robot berhasil bergerak kekanan dalam waktu 1 detik setelah penekanan tombol Right, untuk berhenti membutuhkan waktu sekitar 17 detik.
4	Left 		1 detik	18 detik	Robot berhasil bergerak kekiri dalam waktu 1 detik setelah penekanan tombol Left, untuk berhenti membutuhkan waktu sekitar 18 detik.

4.5 Pengujian Pada Air Keruh

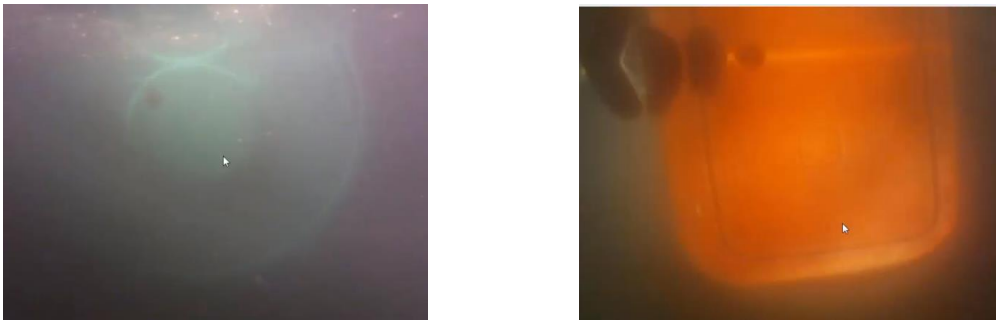
Pengujian juga dilakukan pada kolam dengan air keruh. Awal percobaan dilakukan tanpa mendeteksi objek yang ditetapkan. Pada Gambar 4.11 menunjukan kondisi sebenarnya saat ROV berada di dalam air tanpa menggunkan cahaya dan ketika diberi cahaya sebesar 945 Lux. Dari hasil percobaan pertama diketahui bahwa dalam keadaan air keruh, ROV tidak dapat menangkap gambar apapun tanpa dibantu dengan pencahayaan tambahan.

Selanjutnya dilakukan penggujian warna objek yang dapat dideteksi oleh ROV tersebut. Objek yang dapat dideteksi menggunakan warna jingga dan hijau. Hasilnya menunjukkan bahwa warna hijau hampir menyerupai warna dari air tersebut. Bentuk dari objek tersebut tidak begitu jelas, berbeda pada objek yang berwana jingga yang lebih menampilkan perbedaan warna dengan air tersebut. Bentuk dari objek tersebut juga hampir terbentuk sempurna. Gambar 4.12 menunjukan hasil dari pengujian tersebut.



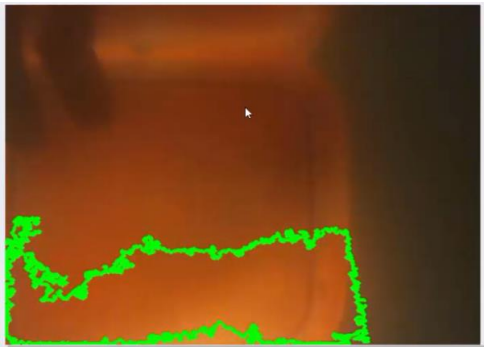
(a) (b)

Gambar 4.11 (a) Kondisi Dalam Air Keruh Tanpa Cahaya (b) Kondisi Dalam Air Keruh Dengan Tambahan Cahaya



(a) (b)

Gambar 4.12 Hasil Tampilan Kamera (a) Objek Berwana Hijau (b) Objek Berwarna Jingga



Gambar 4.13 Deteksi Warna Pada Air Keruh

Percobaan selanjutnya dilakukan deteksi dan *tracking* pada objek yang dilihatkan pada gambar Gambar 4.13. Pada kondisi air keruh menunjukan deteksi objek tersebut terlihat tak beraturan. Terjadinya bentuk warna yang tidak sempurna mengakibatkan warna yang ditunjukan menjadi blur atau tidak pasti. Percobaan pada Gambar 4.13 dilakukan pada jarak sekitar 30 cm.

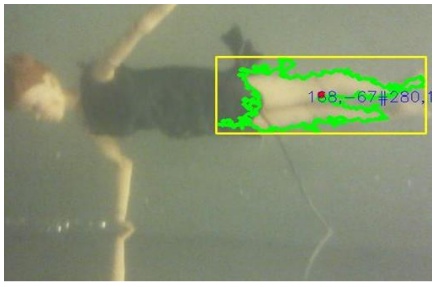
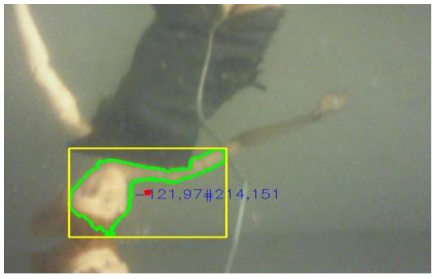
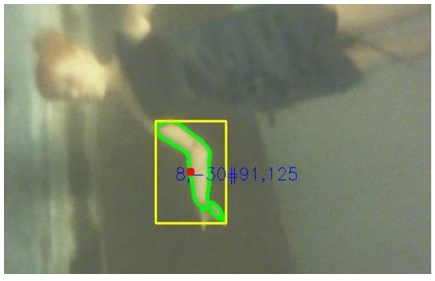
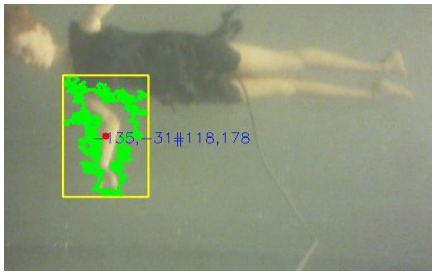
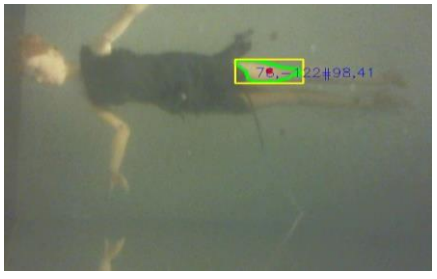
4.6 Pengujian Menggunakan Boneka Sebagai Objek

Boneka yang akan diujikan ialah boneka berbentuk menyerupai manusia dengan ukuran tinggi sekitar 33 cm yang ditunjukkan pada Gambar 4.14. Boneka ini akan dimasukan ke dalam air keruh yang akan diuji pendeteksian oleh *underwater* ROV tersebut. Hasil yang diperoleh dapat dilihat pada Tabel 4.2 dan hasil dalam *threshol* ditunjukkan pada Gambar 4.15 . Dari hasil yang didapat ini menunjukkan bahwa sebagian dari anggota tubuh dapat terdeteksi tetapi sebagian lagi tidak dapat terdeteksi. Pendeteksian pada objek boneka tersebut tidak menunjukkan keseluruhan bentuk tubuh yang dapat terdeteksi.

Gambar 4.14 Objek Boneka

Gambar 4.15 Hasil Deteksi Boneka Dalam Thereshold 1

Tabel 4.2 Hasil Deteksi Pada Boneka

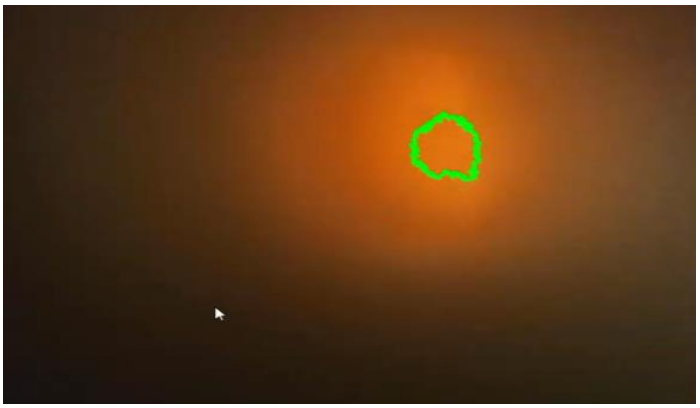
No	Hasil Deteksi	Tingkat Keberhasilan	Keterangan
1.		40%	Hanya bagian kaki yang dapat terdeteksi
2.		25%	Bagian yang terdeteksi antara lain wajah dan sebagaian dari lengan sebelah kanan
3.		10%	Hanya bagian lengan sebelah kanan yang terdeteksi
4.		15%	Bagian lengan sebelah kanan dan terdapat beberapa peluasan deteksi
5.		5%	Minimnya cahaya sehingga hanya bagian paha kiri atas yang terdeteksi

Dari hasil Tabel 4.2 diketahui cahaya yang diterima oleh objek sangat berpengaruh besar pada pendeteksian. Pada jarak sekitar 20 cm pada keadaan air keruh, pendeteksian boneka hanya bagian tertentu yang dapat dideteksi, anatara lain pada bagian kaki dan ketika objek sangat sedikit mendapatkan cahaya, hanya sekitar 5% kemungkinan objek dapat terdeteksi.

4.7 Pengujian Tracking Objek Berdasarkan Jarak

Pada pengujian ini dapat dilakukan pendeteksian objek dengan sebuah benda berwarna jingga pada air yang keruh. Percobaan ini dilakukan pada jarak yang terdekat sampai jarak yang paling jauh dari pandangan ROV. Dilakukan percobaan ini dengan menggunakan jarak antara 20 cm sampai 2 m. Kondisi pada jarak 20 cm ditunjukkan pada Gambar 4.16 dimana deteksi objek hanya beberapa bagian saja. Pada jarak 2 m objek terdeteksi kecil, ditunjukkan pada Gambar 4.17. Dengan nilai cahaya sebesar 945 Lux pandangan ROV masih sangat kurang dalam kondisi air keruh. Jarak pandang tidak fokus, disebabkan kekurangan cahaya yang ditangkap oleh kamera ROV ini.

Gambar 4.16 Deteksi Objek Pada Jarak 20 cm



Gambar 4.17 Deteksi Objek Pada Jarak 2 m

4.8 Pengujian Kecepatan Respon ROV Terhadap Objek

Kecepatan respon ROV sangat dibutuhkan dalam *tracking* objek ini. Di mana objek dapat bergerak ke kiri dan ke kanan dalam waktu dan kecepatan tertentu. Tabel 4.3 menunjukkan respon ROV dari berberapa percobaan yang telah dilakukan. Dari hasil percobaan yang dilakukan, respon ROV saat *tracking* di air jernih lebih cept. Respon ROV dalam berpindah posisi memerlukan waktu rata-rata selama 1 detik.

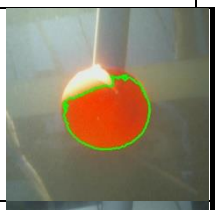
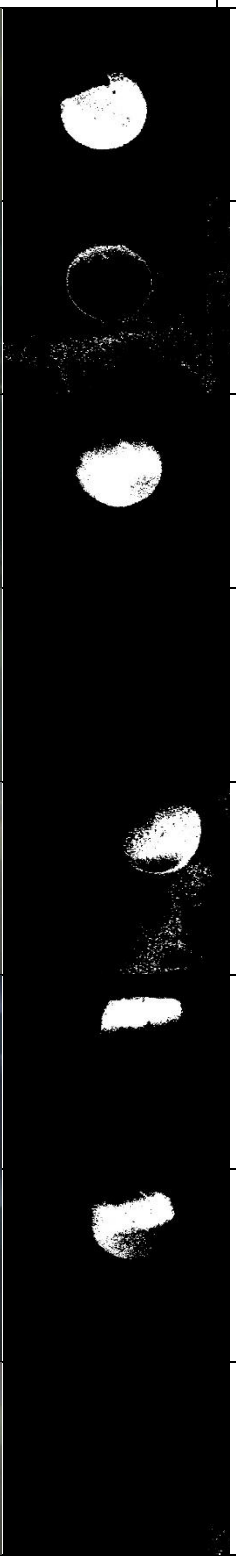
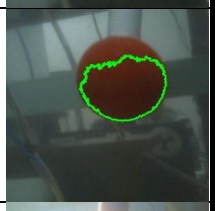
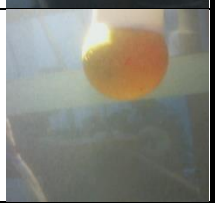
Tabel 4.3 Percobaan Respon Waktu *Tracking*

Percobaan	Waktu Perpindahan	Keterangan
1	1.2 detik	Pada air jernih
2	1 detik	Pada air jernih
3	1 detik	Pada air jernih
4	2 detik	Pada air keruh
5	1.40 detik	Pada air keruh
6	2.3 detik	Pada air keruh

4.9 Pengujian Pada Mode RGB

Pada pengujian mode RGB ini diambil beberapa contoh dari deteksi objek dengan keterbatasan cahaya. Batasan cahaya yang diuji adalah dengan kecerahan cahaya sebesar 29 Lux dan 945 Lux. Hasil dari pengujian ini ditampilkan pada Tabel 4.4.

Tabel 4.4 Hasil Deteksi Objek dengan RGB

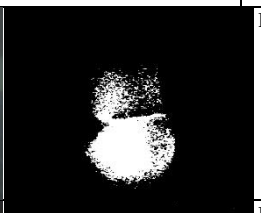
Lux	Range	Image	Threshold	Ukuran objek asli	Ukuran yang terbaca	Kesalahan
945	R=0-245 G=54-255 B=0-43			Pixel=35458 Tinggi=219 Lebar=217	Pixel=27476 Tinggi=184 Lebar=211	22.51%
29	R=0-245 G=54-255 B=0-43			Pixel=31495 Tinggi=191 Lebar=210	Pixel=1847 Tinggi=196 Lebar=291	94.14%
29	R=52-93 G=21-44 B=10-34			Pixel=37575 Tinggi=312 Lebar=442	Pixel=24973 Tinggi=169 Lebar=217	33.54%
945	R=52-93 G=21-44 B=10-34			Pixel=34084 Tinggi=244 Lebar=208	Pixel=0 Tinggi=0 Lebar=0	100%
945	R= 0-160 G=91-255 B=0-79			Pixel=33090 Tinggi=233 Lebar=222	Pixel=17054 Tinggi=231 Lebar=328	48.46%
29	R= 0-160 G=91-255 B=0-79			Pixel=27987 Tinggi=173 Lebar=210	Pixel=12352 Tinggi=92 Lebar=205	55.87%
29	R=67-122 G=54-116 B=24-54			Pixel=30900 Tinggi=180 Lebar=217	Pixel=17064 Tinggi=170 Lebar=208	44.78%
245	R=67-122 G=54-116 B=24-54			Pixel=29837 Tinggi=199 Lebar=215	Pixel=0 Tinggi=0 Lebar=0	100%
					Error rata rata	62.41%

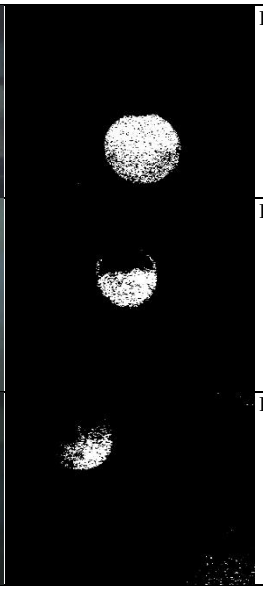
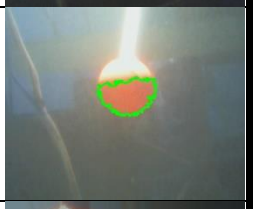
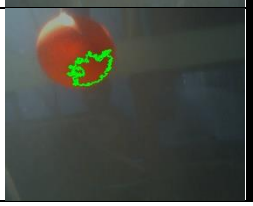
Dari hasil tabel diatas maka diketahui nilai rata-rata *erorr* yang di akibatkan oleh cahaya sebesar 62.41%, ukuran asli objek tersebut tidak terlalu banyak yang di tangkap pada pendeteksian ini.

4.10 Pengujian Pada Mode HSV

Pengujian yang dilakukan pada mode HSV dapat dilakukan sama halnya dengan melakukan pengujian pada RGB. Dengan keterbatasan cahaya yang diberikan sebesar 29 Lux dan 945 Lux. Maka ditunjukan hasilnya pada Tabel 4.5.

Tabel 4.5 Hasil Deteksi Objek dengan HSV

Lux	Range	Image	Threshold	Ukuran objek asli	Ukuran yang terbaca	error
945	R=6-17 G=50-255 B=50-255			Pixel=33541 Tinggi=190 Lebar=225	Pixel=24955 Tinggi=197 Lebar=226	25.6%
29	R=6-17 G=50-255 B=50-255			Pixel=28319 Tinggi=166 Lebar=217	Pixel=140 Tinggi=201 Lebar=252	99.51%
29	R=18-28 G=50-255 B=50-255			Pixel=28769 Tinggi=172 Lebar=220	Pixel=32517 Tinggi=239 Lebar=266	- 13.03%
945	R=18-28 G=50-255 B=50-255			Pixel=23485 Tinggi=245 Lebar=327	Pixel=5399 Tinggi=164 Lebar=196	77.01%
945	R= 1-7 G=50-255 B=50-255			Pixel=24246 Tinggi=169 Lebar=181	Pixel=22966 Tinggi=220 Lebar=210	5.28%

29	R= 1-7 G=50-255 B=50-255			Pixel=27976 Tinggi=181 Lebar=205	Pixel=19090 Tinggi=173 Lebar=192	31.76%
29	R= 2-8 G=48-255 B=48-255			Pixel=16206 Tinggi=149 Lebar=143	Pixel=8406 Tinggi=124 Lebar=152	48.13%
945	R= 2-8 G=48-255 B=48-255			Pixel=28956 Tinggi=200 Lebar=198	Pixel=4521 Tinggi=90 Lebar=110	84.39%
					Error rata rata	44.83%

Dari hasil yang didapatkan, diketahui bahwa nilai error pengujian tersebut lebih kecil. Pendeteksian menggunakan HSV ini meminimalisir *error* yang terjadi saat perubahan cahaya yang ditujukan pada objek tersebut.

Halaman ini sengaja dikosongkan

BAB 5
PENUTUP

Berdasarkan hasil perancangan, pembuatan, dan pengujian sistem pada penelitian dengan judul "RANCANG BANGUN METODE *TRACKING OBJECT* PADA *REMOTELY OPERATED UNDERWATER VEHICLE* (ROV)", penulis dapat memberikan kesimpulan dan saran untuk kemajuan dan perkembangan penelitian berikutnya.

5.1 Kesimpulan

Dari hasil percobaan pertama yang dilakukan dengan menenggelamkan kamera di area yang berbeda. Didapatkan hasil yang menunjukan bahwa perbedaan area antara gelap dan terang membutuhkan cahaya yang lebih baik, guna untuk menambah kualitas dari citra objek yang dapat dicari. Berdasarkan percobaan di kolam yang berlantai gelap ROV tidak dapat melakukan *tracking* objek. Jarak pandang terlalu minim, mengakibatkan *blind vision.* GUI yang digunakan pada ROV ini sangat membatu dalam pengoperasian ROV. Terdapat data-data yang dapat membatu dalam pencarian. Baik sebagai tampilan visual ataupun dalam *tracking* objek.

Percobaan yang dilakukan dengan menggunakan sebuah boneka sebagai objek, dihasilkan beberapa pendeteksian yang didapat. Pada jarak sekitar 20 cm hanya sekitar 40% objek akan terdeteksi. Dan ketika keadaan objek minim cahaya maka hanya mendapatkan sekitar 5% pendeteksiaan. Dapat disimpulkan dari pecobaan yang didapat antara laian adalah bagian tertentu yang mendapatkan cahaya lebih akan mendapatkan hasil deteksi yang lebih baik. Dan kemungkinan jika objek sangat kurang mendapatkan cahaya, secara otomatis objek tersebut tidak dapat terdeteksi.

Pada penelitian ini yang dilakukan dengan beberapa kondisi menghasil berbagai macam efek dari intensitas cahaya yang ditangkap. Beberapa diantaranya ketika pada RGB pengujian pertama dengan objek awal saat cahaya 945 *Lux range* R=0-245, G=54-255, B=0-43, maka didapatkan kesalahan deteksi 22.51%. Ketika

cahaya diredupkan menjadi 29 *Lux* maka nilai error yang didapat 94.14%. Pada percobaan kedua objek awal saat cahaya 29 *Lux* denga *range* yang di dapat R=52-93, G=21-44, B=10-34 didapatkan *error* sebesar 33.54% dan ketika cahaya ditingkatkan menjadi 945 *Lux* maka nilai *error* menjadi 100%. Terlalu besar untuk pendeteksian dan akibatnya tidak dapat mendeteksi objek yang ditargetkan.

Selanjutnya ketika menggunakan mode HSV dengan kasus pertama objek awal saat cahaya 945 *Lux range* R=6-17, G=50-255, B=50-255, didapatkan error sebesar 25.6% dan ketika cahaya diturunkan menjadi 29 *Lux,* maka didapatkan error sebesar 99.51%, contoh kedua objek awal dengan cahaya 29 *Lux* dengan besar nilai R=18-28, G=50-255, B=50-255, didapatkan *error* sebesar 13.03% dan ketika cahaya dinaikkan menjadi 945 *Lux* maka didapatkan *error* 13.03%.

Dari beberapa contoh kasus yg dijelaskan makan perbandingan persentase kesalahan yang didapat antara RGB dan HSV adalah 62.41% : 44.83%. Nilai HSV masih bisa diunggulkan dalam pendeteksian objek bawah air. Tetapi pada dasarnya efek yang paling besar dalam pendeteksian objek bawah air ini adalah efek dari cahaya yang diserap oleh objek itu sendiri.

5.2 Saran

Cahaya adalah faktor paling besar dalam metode *Image Processing,* lakukan menggunakan metode yang lebih baik lagi dari RGB dan HSV. Pada ROV sendiri wajib dilengkapinya cahaya internal agar dapat membatu sistem *image processing* yang dibangun.

DAFTAR PUSTAKA

[1] Melky, "Rov Si Robot Pencari Air Asia QZ8501," Tempo, 3 January 2015. [Online]. Available: http://nasional.tempo.co/read/news/2015/01/03/058632553/rov- si-robot-pencari-air-asia QZ8501. [Accessed 12 Juni 2015].

[2] L. D., "TROJAN: Remotely Operated Vehicle," *IEEE Journal of Oceanic Engineering,* vol. 11, no. 3, pp. 364-372, 1986.

[3] N. Shahriar and F. Pezhman, "An ROV Stereovisio System for Sip-Hull Inspection," *IEEE Journal Oceanic Engineering,* vol. 31, no. 3, pp. 551-564, 2006.

[4] C. K. George, J. P. Dimitra and K. J. Kyriakopoulos, "Target-referenced Localization of an Underwater Vehicle using a Laser-based Vision System," *OCEANS,* pp. 1-6, 2006.

[5] D. C. J.P. V. Soares, C. R. R. and H. Liu, "Design of a High Performance Variable Structure Position Control of ROV's," *Oceanic Engineering,* vol. 20, pp. 42-55, 1995.

[6] L. D. M., C. M. J. and D. Dongyong, "Robust Tracking of Multiple Objects in Sector-scan Sonar Image Sequences Using Optical Flow Motion Estimation," *IEEE Journal Oceanic Engineering,* vol. 23, pp. 31-46, 1998.

[7] Z. Side and Y. J., "Experimental Study on Advanced Underwater Robot Control," *IEEE Transactions on Robotics,* vol. 21, no. 4, pp. 695-703, 2001.

[8] S. J. and D. G., "On the Performance of Color Tracking Algorithms for Underwater Robot under Varying Lightig and visibility," in *Proceedings of the 2006 IEEE International Conference on Robotic and Automation*, Orlando, Florida, 2006.

[9] R. J. and R. S.M., "Segmentation Methods for Visual Tacking of Deep-Ocean Jellyfish Using a Convensional Camera," *IEEE Journal of Oceanic Engineering,* vol. 28, no. 4, pp. 595-608, 2003.

[10] S. Linbo and X. Guangming, "Real-Time Tracking Of Moving Objects On A Water Surface," in *Proceedings of the 2012 IEEE International Conference on Mechatronics and Automation*, Chengdu, China, 2012.

[11] S. Pritpal, D. B.B.V.L., S. Tanjot and D. P. M. Meta, "Real-time object detection and Tracking using color feature and motion," in *2015*

International Conference on Communications and Signal Processing (ICCSP), Melmaruvathur, India, 2015.

[12] Ç. Yunus, A. Mahmut and G. Mahit, "Color based moving object tracking with an active camera using motion information," in *2017 International Artificial Intelligence and Data Processing Symposium (IDAP)*, Malatya, Turkey, 2017.

[13] Q. Yuang and A. Gang, "Infreared Moving Target Detection Based on Optical Flow Estimation," *Proceedings of 2011 International Conference on Computer Science and Network Technology,* vol. 4, pp. 2452 - 2455, 2011.

[14] W. Zhen and Y. Xiaojun, "Moving Target Detection and Tracking Based on Pyramid Lucas-Kanade Optical Flow," *2018 IEEE 3rd International Conference on Image, Vision and Computing (ICIVC),* pp. 66 - 69, 2018.

[15] C. Yanfeng and W. Qingxiang, "Moving Vehicle Detection Based on Optical Flow Estimation of Edge," *2015 11th International Conference on Natural Computation (ICNC),* pp. 754 - 758, 2015.

[16] C. Yun-Won, C. Yun-Su, L. Soo-In and L. Suk-Gyu, "Rear Object Detection Method Based on Optical Flow and Vehicle Information for Moving Vehicle," *2017 Ninth International Conference on Ubiquitous and Future Networks (ICUFN),* pp. 203 - 205, 2017.

[17] C. Zhiwen, C. Jianzhong, T. Yao and T. Linao, "Tracking of Moving Object Based on Optical Flow Detection," *Proceedings of 2011 International Conference on Computer Science and Network Technology,* pp. 1096 - 1099, 2011.

[18] K. Donghoon, L. Donghwa, M. Hyun and C. Hyun-Tak, "Object Detection and Tracking for Autonomous Underwater Robots Using Weighted Template Matching," *2012 Oceans - Yeosu,* pp. 1-5, 2012.

[19] C. Wisarut and H. Yo-Sung, "Object detection based on fast template matching through adaptive partition search," *2015 12th International Joint Conference on Computer Science and Software Engineering (JCSSE),* pp. 1 - 6, 2015.

[20] B. Kawisara, K. Narongdech and C. Nutchanun, "Real Time Automatic Object Detection by Using Template Matching for Protecting Pipelines," *2018 International Workshop on Advanced Image Technology (IWAIT),* pp. 1-4, 2018.

[21] S. G. Chetan and S. K. Atish, "Object Detection Using Color Clue and Shape Feature," *2016 International Conference on Wireless*

Communications, Signal Processing and Networking (WiSPNET), pp. 464-468, 2016.

[22] K. Sang-Hoon, K. Nam-Kyu, C. A. Sang and K. Hyoung-Gon, "Object Oriented Face Detection Using Range and Color Information," *Proceedings Third IEEE International Conference on Automatic Face and Gesture Recognition,* pp. 76-81, 1998.

[23] W. Hanzhang, X. Qingguo, Y. Qingwei and W. Xiaodong, "Cross Camera Object Tracking in High Resolution Video Based on TLD Framework," *2015 IEEE International Conference on Multimedia Big Data,* pp. 264-267, 2015.

[24] N. S. Swati, K. Ajitkumar and M. Dilip, "Multi-Object Tracking Using TLD Framework," *2016 IEEE International Conference on Recent Trends in Electronics, Information & Communication Technology (RTEICT),* pp. 1766-1769, 2016.

[25] N. S. Swati, K. Ajitkumar and M. Dilip, "Real Time Multi-Object Tracking Using TLD Framework," *2016 IEEE International Conference on Recent Trends in Electronics, Information & Communication Technology (RTEICT),* pp. 1766 - 1769, 2016.

[26] F. A. Hermawati, Pengolaan Citra Digial: Konsep dan teori, Yogyakarta: Andi, 2013.

[27] H. Figur, A. Kusworo and C. E. Widodo, "Aplikasi Pengolahan Citra Pada Raspberry PI untuk Membedakan Benda Berdasarkan Warna dan bentuk," *Youngster Physics Journal,* vol. 5, pp. 157-162, 2016.

[28] C. Francois and H. Seth, "Visual Servo Control 1. Basic Approaches," in *IEEE Robotics & Automation agazine,* vol. 13, 2007, pp. 82-90.

[29] C. Feifei, K. Dong Joong and P. Jun Hyub, "New measurement method of Poisson's Ratio of PVA Hydrogels Using an Optical flow Analysis for a Digital Imaging System," *Measurement Sciencend Technology,* vol. 24, pp. 1-8, 2013.

[30] B. Jean Yves, "Pyramidal Implementation of The Lucas-Kanade Feature Tracker," *Intel Corporation, Mircroprocessor research Labs,* 2000.

[31] L. Brian and E. Alex, "Efficient Control of An AUV-Manipulator System: An Aplication for the Exploration of Europa," *IEEE Journal of Oceanic Engineering,* vol. 39, pp. 552-570, 2014.

[32] T. Sutoyo and E. Mulyanto, Teori Pengolahan Citra Digital, Yogyakarta: Andi, 2009.

LAMPIRAN

GUI Code

http.py

```
import cv2
from PIL import Image
from BaseHTTPServer import
BaseHTTPRequestHandler,HTTPServer
import StringIO
from threading import Thread,Lock,Event
import numpy as np
import serial_util as ser
import img_detek
import subprocess
import time

simulasi=True

capture=None
w_img=640
h_img=480 #590

s_img = cv2.imread("www/nav.png")
s_img_r = cv2.imread("www/nav_r.png")
s_img_y = cv2.imread("www/nav_y.png")

rows0,cols0,_ = s_img.shape
rows1,cols1,_ = s_img_r.shape

ofx0=w_img/2-cols0/2
ofy0=h_img/2-rows0/2

ofx1=w_img/2-cols1/2
```

```
ofy1=h_img/2-rows1/2

lock = Lock()
dict1 = {}
contoh=0

capture = cv2.VideoCapture(0) #kamera 0

class kameraku(Thread):

def __init__(self, event,auto):
            Thread.__init__(self)
            self.stopped = event
            self.auto=auto
            self.kotak=0.14

def run(self):
            lock.acquire()
            dict1[0]=np.zeros((480,640,3),np.uint8)
            dict1[1]=np.zeros((480,640,3),np.uint8)
            dict1[2]=np.zeros((480,640,3),np.uint8)
            i=0
            try:
            pr=False
            while not self.stopped.isSet():
                pos=ser.get(1)
                rc,img = capture.read()
                img =
                cv2.cvtColor(img,cv2.COLOR_BGR2RGB)
                dict1[0]=img.copy()
```

```
            ppp,dict1[2],dict1[1]=img_detek.read_objek(img,2000)
            #ppp=posisi objek [x,y]  ,,, dict1[2] gambar camera
            ,,, dict1[1] threshold
                        if(self.auto.isSet()):
                            ser.write("x"+str(-ppp[0])+"\n")
                        time.sleep(0.1)
                finally:
                    lock.release()
                    capture.release()
                    print "kamera terhenti"
                    ser.write("x0\n")
                    #ser.write("y0\n")


proses=False

class myHandler(BaseHTTPRequestHandler):
    i=0
    def do_GET(self):
        global s_img,s_img_r,s_img_y
        global
ofx0,ofy0,ofx1,ofy1,rows0,cols0,rows1,cols1
        global proses,mulpro,dict1
        global fuzzyFlag

        if(proses):
            return
        patweb=self.path[0:5]
        if(patweb in ["/cam?","/bin?","/ori?"]):
            if patweb =="/bin?":
                img=dict1[1]
            elif patweb =="/cam?":
```

```
                img=dict1[2]
            else:
                img=dict1[0]

            if self.path[7:8]=="0":
                mpu=ser.get(1)
                pi=float(mpu[3])
                ro=float(mpu[4])
                ya=float(mpu[5])
                ya=ya*180/np.pi
                Mr =
cv2.getRotationMatrix2D((cols1/2,rows1/2),ro,1)
                My =
cv2.getRotationMatrix2D((cols1/2,rows1/2),ya,1)

                s_img_r1 =
cv2.warpAffine(s_img_r,Mr,(cols1,rows1))
                s_img_y1 =
cv2.warpAffine(s_img_y,My,(cols1,rows1))

                s_img1=np.copy(s_img);
                s_img1 = self.d_line(s_img1,0,pi,0,0)


                dst = cv2.add(s_img_r1,s_img_y1)
                dst = cv2.add(s_img1[1:241,25:265],dst)
                s_img1[1:241,25:265]=dst
                dst =
cv2.add(s_img1,img[ofy0:ofy0+rows0, ofx0:ofx0+cols0])
                img[ofy0:ofy0+rows0,
ofx0:ofx0+cols0]=dst
            jpg = Image.fromarray(img)
```

```
            tmpFile = StringIO.StringIO()
            jpg.save(tmpFile,'JPEG')
            self.send_response(200)
            self.send_header('Content-
type','image/jpeg')
            self.send_header('Content-
length',str(img.size))
            self.end_headers()
            self.wfile.write(tmpFile.getvalue())
        elif(patweb=="/wts?"):
            self.send_response(200)
            self.send_header("Content-type",
"text/html")
            self.end_headers()
            self.wfile.write(img_detek.lstx)
        elif(patweb=="/tom?"):
            proses=True
            self.tombol(self.path[7:])
            proses=False
            self.send_response(200)
        elif(patweb=="/fuz?"):
            proses=True
            pos=ser.get(1)
            try:
                x,y=float(pos[0]),float(pos[1])
            except:
                pass
            fuzzyFlag.set()
            proses=False
            self.send_response(200)
        elif(patweb=="/sfu?"):
```

```
                proses=True
                x,y=self.path[7:].split(',')
                try:
                    x,y=float(x),float(y)
                except:
                    pass
                fuzzyFlag.set()
                proses=False
                self.send_response(200)
            elif(patweb=="/nav?"):
                proses=True
                x,y=self.path[7:].split(',')
                ser.write("fuz="+str(x)+","+str(y)+"\n")
                proses=False
                self.send_response(200)
            elif(patweb=="/gdt?"):
                dt=ser.serial_data(self.path[7:])
                self.send_response(200)
                self.send_header("Content-type",
"text/html")
                self.end_headers()
                self.wfile.write(dt)
            elif(patweb=="/adt?"):

                mpu=ser.get(1)

web="<table><tr><td>Pitch</td><td>:</td><td>"
                web+=mpu[3]

web+="</td></tr><tr><td>Roll</td><td>:</td><td>"
                web+=mpu[4]
```

```
web+="</td></tr><tr><td>Yaw</td><td>:</td><td>"
            web+=mpu[5]

web+="</td></tr><tr><td>X</td><td>:</td><td>"
            web+=str(img_detek.ix) #mpu[0]

web+="</td></tr><tr><td>Y</td><td>:</td><td>"
            web+=str(img_detek.iy) #mpu[1]

web+="</td></tr><tr><td>Z</td><td>:</td><td>"
            web+=mpu[2]
            web+="</td></tr></table>"
            self.send_response(200)
            self.send_header("Content-type",
"text/html")
            self.end_headers()
            self.wfile.write(web)
        elif(patweb=="/obj?"):
            web="<table>"
            web+="<tr><td>Data Objek yang
terdetaksi</td></tr>"
            for (i, pos) in
enumerate(img_detek.posisi):
                #for pos in img_detek.posisi:
                web+="<tr><td><input type=\"image\"
src=\"/imo?v="+str(i)+"\"
onClick=\"objek('"+pos[0]+"','"+pos[1]+"');\"></td></tr
>"
            web+="</table>"
            self.send_response(200)
```

```
            self.send_header("Content-type",
"text/html")
            self.end_headers()
            self.wfile.write(web)
        elif(patweb=="/mou?"):
            x_0,y_0,x_1,y_1=self.path[7:].split(',')

x0,y0,x1,y1=int(x_0),int(y_0),int(x_1),int(y_1)
            img=dict1[2]
            img_detek.set_objek(img[y0:y1,x0:x1],"HSV")

#img_detek.set_warna(rmin,gmin,bmin,rmax,gmax,bmax)
            #img_detek.set_templat(img[y0:y1,x0:x1])
            self.send_response(200)
        elif (patweb=="/pet?"):
            x_0,y_0=self.path[7:].split(',')
            img=cv2.resize(img[730:860,710:860]*40,
(640,480), interpolation = cv2.INTER_AREA)
            jpg = Image.fromarray(img)

            tmpFile = StringIO.StringIO()
            jpg.save(tmpFile,'JPEG')
            self.send_response(200)
            self.send_header('Content-
type','image/jpeg')
            self.send_header('Content-
length',str(img.size))
            self.end_headers()
            self.wfile.write(tmpFile.getvalue())
        elif (patweb=="/imo?"):
            i=int(self.path[7:])-1
            if(len(img_detek.objek)>0):
```

```
                img=cv2.resize(img_detek.objek[i],
(300,225), interpolation = cv2.INTER_AREA)
            else:
                img=np.zeros((480,640,3),np.uint8)
            jpg = Image.fromarray(img)

            tmpFile = StringIO.StringIO()
            jpg.save(tmpFile,'JPEG')
            self.send_response(200)
            self.send_header('Content-
type','image/jpeg')
            self.send_header('Content-
length',str(img.size))
            self.end_headers()
            self.wfile.write(tmpFile.getvalue())
        else:
            if(self.path=="/"):
                self.path="/index.html"
            f = open("./www"+self.path)
            self.send_response(200)
            self.end_headers()
            self.wfile.write(f.read())

    def d_line(self,img,x,y,ofx0,ofx1):
        y0=int(121+121*x/45+121*y/45)
        y1=int(121-121*x/45+121*y/45)

cv2.line(img,(ofx0,y0),(ofx0+29,y1),(255,255,255),5)
        cv2.line(img,(ofx1-
29,y0),(ofx1,y1),(255,255,255),5)
        return img
```

```
    def tombol(self,tom):
        global fuzzyFlag,kotak,contoh
        if(tom=='0'):#stop
            fuzzyFlag.clear()
            ser.write('stop\n')
            ser.write("x0\n")
            ser.write("y0\n")
        elif(tom=='1'):#maju
            ser.write('up\n')
        elif(tom=='2'):#mundur
            ser.write('down\n')
        elif(tom=='3'):#kanan
            ser.write('shri\n')
        elif(tom=='4'):#kiri
            ser.write('shle\n')
        elif(tom=='5'):#kiri
            process =
subprocess.call("reboot",shell=True)
            output, error = process.communicate()
        elif(tom=='6'):#ambil gambar

cv2.imwrite("img/"+str(contoh)+np.array2string(img_dete
k.lower)+np.array2string(img_detek.upper)+str(img_detek
.ix)+"
"+str(img_detek.iy)+"asli.jpg",cv2.cvtColor(dict1[0],cv
2.COLOR_BGR2RGB))

cv2.imwrite("img/"+str(contoh)+np.array2string(img_dete
k.lower)+np.array2string(img_detek.upper)+str(img_detek
.ix)+" "+str(img_detek.iy)+"thres.jpg",dict1[1])

cv2.imwrite("img/"+str(contoh)+np.array2string(img_dete
```

```
k.lower)+np.array2string(img_detek.upper)+str(img_detek
.ix)+"
"+str(img_detek.iy)+"objek.jpg",cv2.cvtColor(dict1[2],c
v2.COLOR_BGR2RGB))
            contoh=contoh+1

fuzzyFlag = Event()

def main():
    global fuzzyFlag
    file2write=open("t_proses.txt",'w')
    file2write.write("")
    file2write.close()
    stopFlag = Event()

    thread = kameraku(stopFlag,fuzzyFlag)
    thread.start()
    try:
        httpd = HTTPServer(('', 8080), myHandler)
        httpd.serve_forever()
    except KeyboardInterrupt:
        httpd.socket.close()
        ser.close()
        stopFlag.set()
        capture.release()

if __name__ == '__main__':
    main()
```

Deteksi Objek

Img_detek.py

```
import cv2
import numpy as np

md="HSV"
lower = np.array([60,50,50], dtype = "uint8")
upper = np.array([149,255,255], dtype = "uint8")

templat=np.zeros((10,10),np.uint8)+255
w,h=0,0
ix,iy=0,0
objek=[]
posisi=[]
posisi_x=[]
posisi_y=[]
lstx=0

def set_objek(img1,mode="HSV"):#crop
    global md
    md=mode
    if (mode=="RGB"):
        rmin=img1[:,:,0].min()
        gmin=img1[:,:,1].min()
        bmin=img1[:,:,2].min()

        rmax=img1[:,:,0].max()
        gmax=img1[:,:,1].max()
        bmax=img1[:,:,2].max()
        set_warna(rmin,gmin,bmin,rmax,gmax,bmax)
```

```
    elif (mode=="HSV"):
        img=cv2.cvtColor(img1,cv2.COLOR_RGB2HSV)
        hmin=img[:,:,0].min()
        smin=50#img[:,:,1].min()
        vmin=50#img[:,:,2].min()

        hmax=img[:,:,0].max()
        smax=255#img[:,:,1].max()
        vmax=255#img[:,:,2].max()
        set_warna(hmin,smin,vmin,hmax,smax,vmax)

def read_objek(img,batas): #batas dari th luas area
    if(md=="RGB"):
        return read_warna(img.copy(),img,batas)
    elif(md=="HSV"):
        return
read_warna(cv2.cvtColor(img[:,:],cv2.COLOR_RGB2HSV),img
,batas)

def set_warna(H_min,S_min,V_min,H_max,S_max,V_max):
    global lower,upper

    lower = np.array([H_min,S_min,V_min], dtype =
"uint8")
    upper = np.array([H_max,S_max,V_max], dtype =
"uint8")


def read_warna(img,image,batas):
    global lower,upper,md,lstx,ix,iy
    mask = cv2.inRange(img, lower, upper) #threshold
```

```
    contours,hierarchy = cv2.findContours(mask.copy(),
1, 1) #menccari berdasarkan data threshold menghasilkan
Vector of poin mode retr list metode approxnone
    for cnt in contours:
        if cv2.contourArea(cnt)>=batas: #mencari
conture batas minum

cv2.drawContours(image[:,:],[cnt],0,(0,255,0),1)
#menggambar garis hijau
            x,y,w,h=cv2.boundingRect(cnt) #

cv2.rectangle(image[:,:],(x,y),(x+w,y+h),(255,255,0),1)
#menggambar kotak kuning
            ix,iy=x+w/2-320,y+h/2-240
            #M=cv2.moments(cnt)
            #ix,iy =int(M["m10"]/M["m00"])-320,
int(M["m01"]/M["m00"])-240
            lstx=ix #upate xterakhir

cv2.circle(image[:,:],(ix+320,iy+240),5,(255,0,0),-1)
#mengambar titik bulat merah pada tengah objek

cv2.putText(image[:,:],str(ix)+","+str(iy)+"#"+str(w)+"
,"+str(h),(ix+320-
20,iy+240+10),cv2.FONT_HERSHEY_SIMPLEX,0.7,(0,0,255))
            return
[ix,iy],image,cv2.cvtColor(mask,cv2.COLOR_GRAY2RGB)

    return [0,0],image,mask
```

Serial Raspberry

Serial_util.py

```
# -*- coding: utf-8 -*-
"""
Created on Mon May 02 10:00:08 2018

@author: Tri Susanto
"""
from threading import Thread,Event,Lock
import Queue
import serial
import time

lock = Lock()
shared_dict = {1:["0","0","0","0","0","0"]}
class MyThread(Thread):
    def __init__(self, event,q):
        Thread.__init__(self)
        self.stopped = event
        self.q = q
        self.thread = Thread(target=self.run)

    def run(self):
        self.ser = serial.Serial('com13',9600)
        lock.acquire()
        while not self.stopped.isSet():
            if not workQueue.empty():
                a,lokasi=self.q.get()
                self.ser.flushInput()
                self.ser.flushOutput()
                self.ser.write(a)
```

```
                print a
        self.ser.close()
        self.stopped.set()
        print "berhenti"

    def serial_data(self,dt):
        self.ser.write(dt)
        print dt
        time.sleep(0.1)
        out=""
        while self.ser.inWaiting():
            out+=self.ser.read(1)
        return out

stopFlag = Event()
ser = serial.Serial('com13',9600)
workQueue = Queue.Queue(25)
thread = MyThread(stopFlag,workQueue)
def write(dt,lokasi=0):
    #ser.write(dt)
    global workQueue
    if not dt in workQueue.queue:
        workQueue.put((dt,lokasi))
    se_open()

def get(lokasi):
    if shared_dict.has_key(lokasi):
        return shared_dict[lokasi]
    else:
        return 0

def close():
```

```
    global stopFlag
    stopFlag.set()

def se_open():
    global thread,stopFlag,workQueue
    if not thread.is_alive():
        stopFlag.clear()
        thread = MyThread(stopFlag,workQueue)
        thread.start()

def atur(lokasi,data):
    shared_dict[lokasi]=data
```

Motor.ino

```
#include "Wire.h"
#include "EEPROM.h"
#include "RunningAverage.h"

#define cw    0
#define ccw   1
#define brake 2

//motor_dc control
int inApin[6] = {22, 24, 26, 28, 30, 32};  // INA:
Clockwise input
int inBpin[6] = {23, 25, 27, 29, 31, 33}; // INB:
Counter-clockwise input
int pwmpin[6] = {2, 3, 5, 6, 7, 8}; // PWM input
//int enpin[2] = {0, 1}; // EN: Status of switches
output (Analog pin)

void setup() {
  // put your setup code here, to run once:
  Serial.begin(9600);
  Serial1.begin(9600);
  Serial3.begin(115200);
  Wire.begin();

  // Initialize braked
  for (int i=0; i<7; i++)
  {
    digitalWrite(inApin[i], LOW);
    digitalWrite(inBpin[i], LOW);
  }
```

```
void motorGo(int motor, int direct, int pwm)
{
  if (motor <= 6)
  {
    if (direct == 0)
    {
      digitalWrite(inApin[motor], HIGH);
      digitalWrite(inBpin[motor], LOW);
    }
    else if (direct == 1)
    {
      digitalWrite(inApin[motor], LOW);
      digitalWrite(inBpin[motor], HIGH);
    }
  }
  analogWrite(pwmpin[motor], pwm);
}

void up()
{
  motorGo(4,cw,220);
  motorGo(5,ccw,220);
}
void down()
{
  motorGo(4,ccw,220);
  motorGo(5,cw,220);
}
void forward()
{
  motorGo(0,ccw,220);
  motorGo(1,cw,220);
```

```
  motorGo(2,cw,220);
  motorGo(3,ccw,220);
}
void back()
{
  motorGo(0,cw,220);
  motorGo(1,ccw,220);
  motorGo(2,ccw,220);
  motorGo(3,cw,220);
}
void shift_left()
{
  motorGo(0,cw,220);
  motorGo(1,cw,220);
  motorGo(2,cw,220);
  motorGo(3,cw,220);
}
void shift_right()
{
  motorGo(0,ccw,220);
  motorGo(1,ccw,220);
  motorGo(2,ccw,220);
  motorGo(3,ccw,220);
}
void right()
{
  motorGo(0,cw,220);
  motorGo(1,cw,220);
  motorGo(2,ccw,220);
  motorGo(3,ccw,220);
}
void left()
```

```
{
  motorGo(0,ccw,220);
  motorGo(1,ccw,220);
  motorGo(2,cw,220);
  motorGo(3,cw,220);
}
void berhenti()
{
  motorGo(0,cw,0);
  motorGo(1,cw,0);
  motorGo(2,cw,0);
  motorGo(3,cw,0);
  motorGo(4,cw,0);
  motorGo(5,cw,0);
}
void track_kiri(int sped)
{
  motorGo(0,ccw,sped);
  motorGo(1,ccw,sped);
  motorGo(2,cw,sped);
  motorGo(3,cw,sped);
}
void track_kanan(int sped)  //kendali gerak dan
kecepatan
{
  motorGo(0,cw,sped);
  motorGo(1,cw,sped);
  motorGo(2,ccw,sped);
  motorGo(3,ccw,sped);
}

//program trancking objek
```

```
void tracking(int nilai){
  int kec = map(abs(nilai),0,320,0,200);  //konversi
pixel 0-320 ke kec.motor(PWM) 0-200
  if(nilai>0){
    track_kanan(kec);
  }
  else if(nilai<0){
    track_kiri(kec);
  }
  else{
    berhenti();
  }
}
```

Serial arduino ke raspberry

serial_3.ino

```
//Serial data arduin - raspberry
void serialEvent3(){
  while (Serial3.available()) {
     char inChar = (char)Serial3.read();     //menerima
data serial
     if (inChar == '\n') {
      Serial.println(inputString3);
      if(inputString3=="up"){
          up();
//          Serial.println("UP");
//          track_kanan(20);
      }
      else if(inputString3=="down"){
          down();
//          Serial.println("DOWN");
//          track_kanan(80);
      }
      else if(inputString3=="forw"){
          forward();
//          Serial.println("FORWARD");
      }
      else if(inputString3=="back"){
          back();
      }
      else if(inputString3=="stop"){
          berhenti();
//          Serial.println("STOP");
      }
      else if(inputString3=="righ"){
```

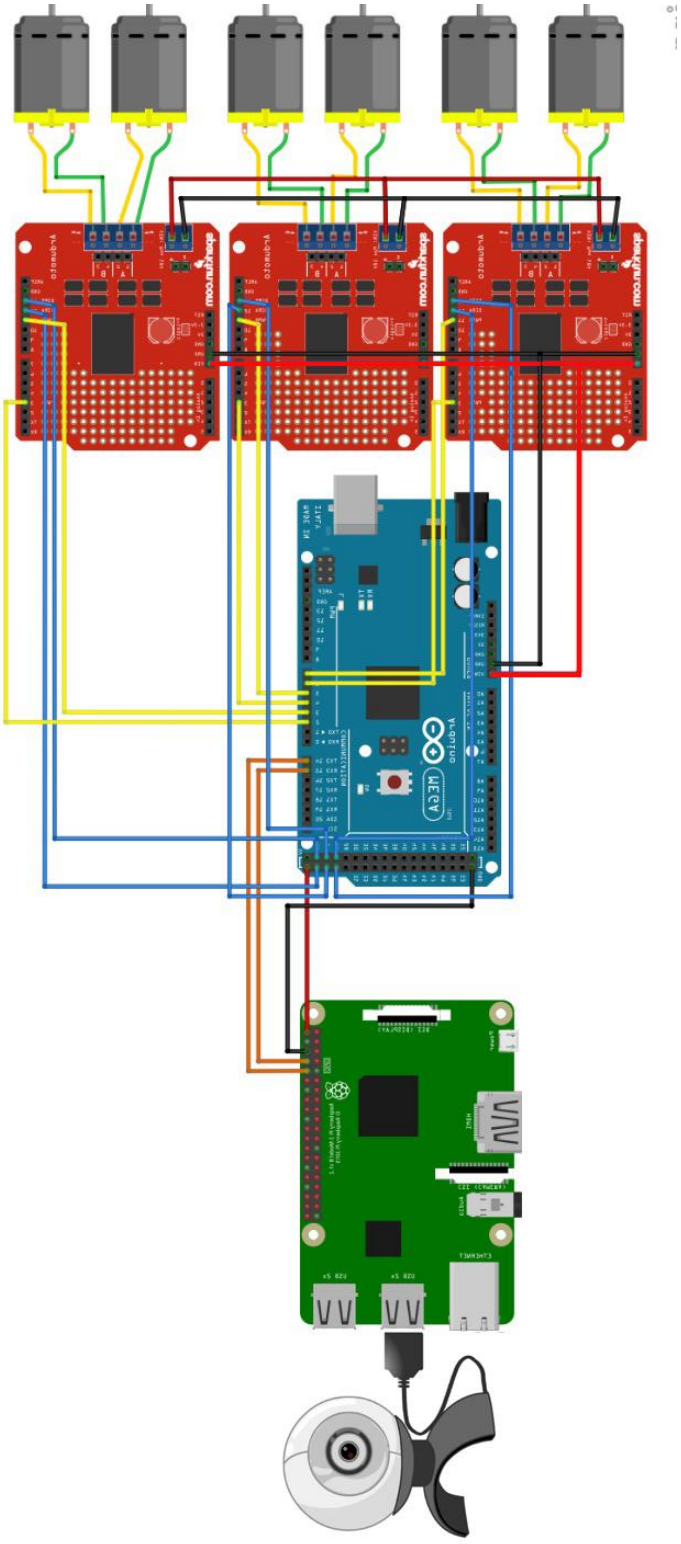
```
            right();
        }
        else if(inputString3=="left"){
            left();
        }
        else if(inputString3=="shri"){
            shift_right();
//            Serial.println("Shift_kanan");
        }
        else if(inputString3=="shle"){
            shift_left();
//            Serial.println("Shift_kiri");
        }
        else if(inputString3.substring(0,1)=="x"){
//x120 merupakan data pixel objek
          pixel_x =inputString3.substring(1).toInt();
//parsing data
          tracking(pixel_x);
          //Serial.println(pixel_x);
        }

        inputString3="";
      }
      else{
        inputString3 += inChar;
      }
    }
}
```

Bahan dan Komponen Elektrik pada *Underwater* ROV

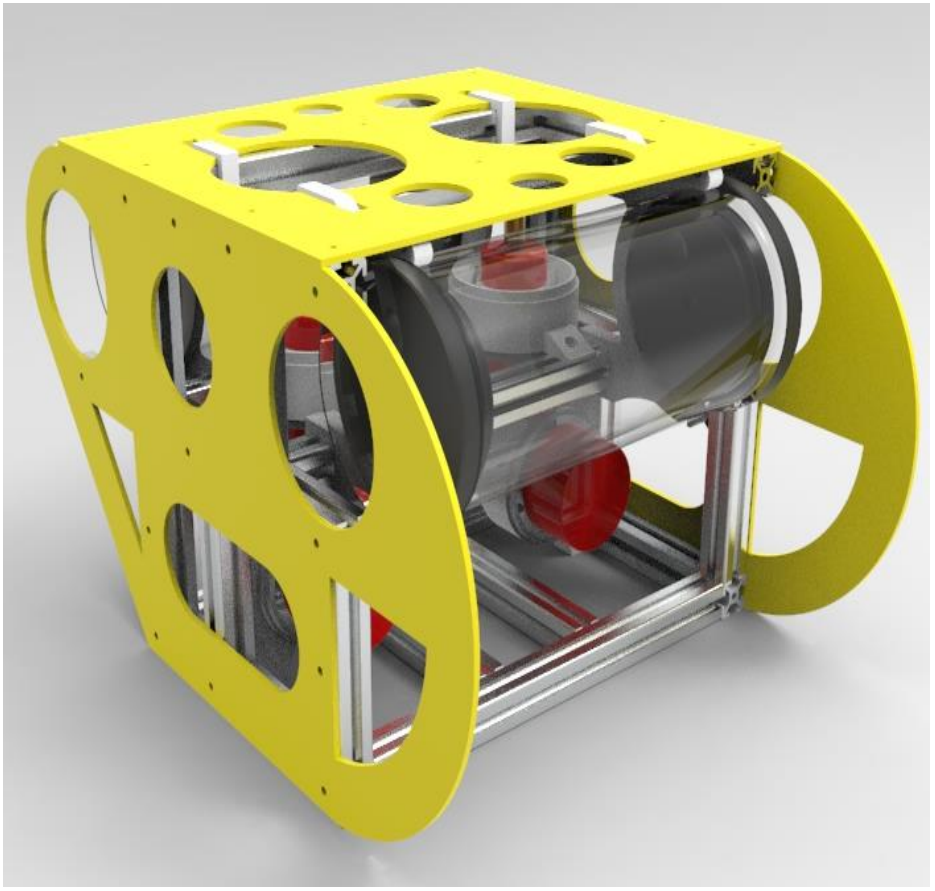
Mekanik ROV	Elektrik
1. Alumunium profile 2. Acrylic lembaran 3. Acrilic tube 4. PLA 3Dprint 5. Seal oring 6. Penetrator kabel	1. Raspberry pi 3 2. Arduino Mega 3. Monster motor shield driver motor 4. Ubec 5 A 5. Kamera Webcam Logitech c525 6. Kabel LAN 7. Lampu LED 12 volt 8. Motor DC Bilge Pump

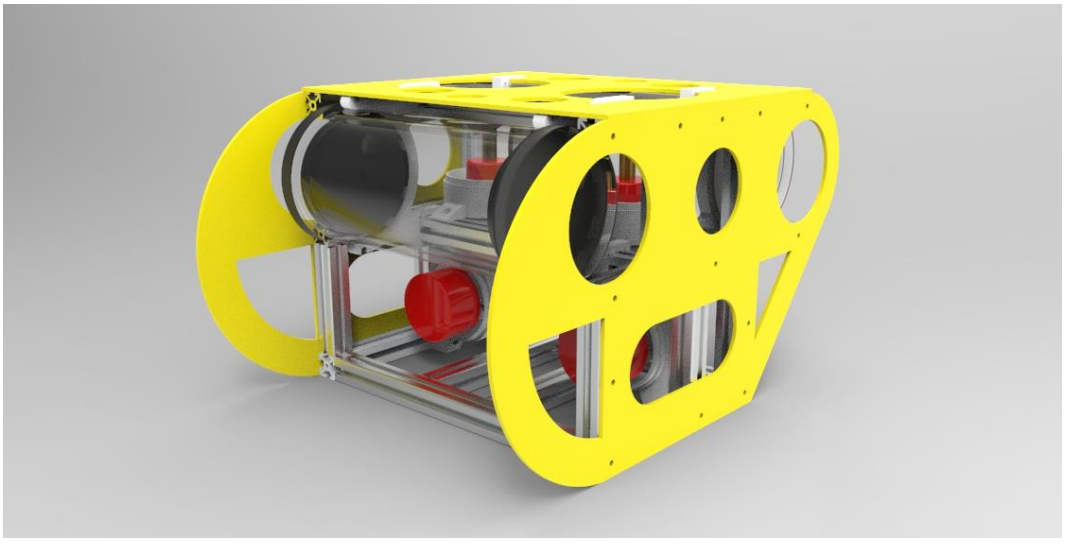
Skematik rangkaian keseluruhan

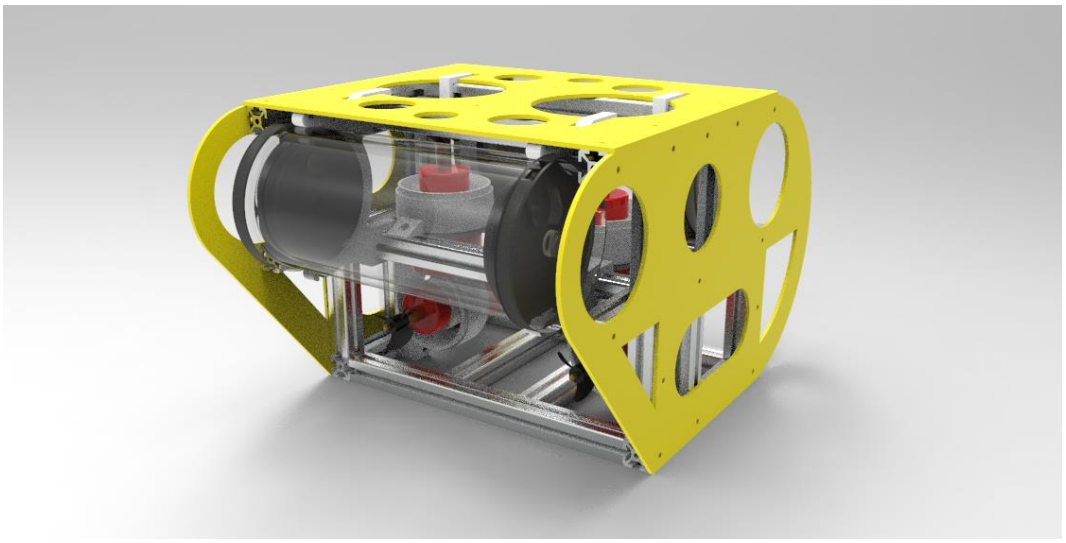


Desain ROV 1 dan 2

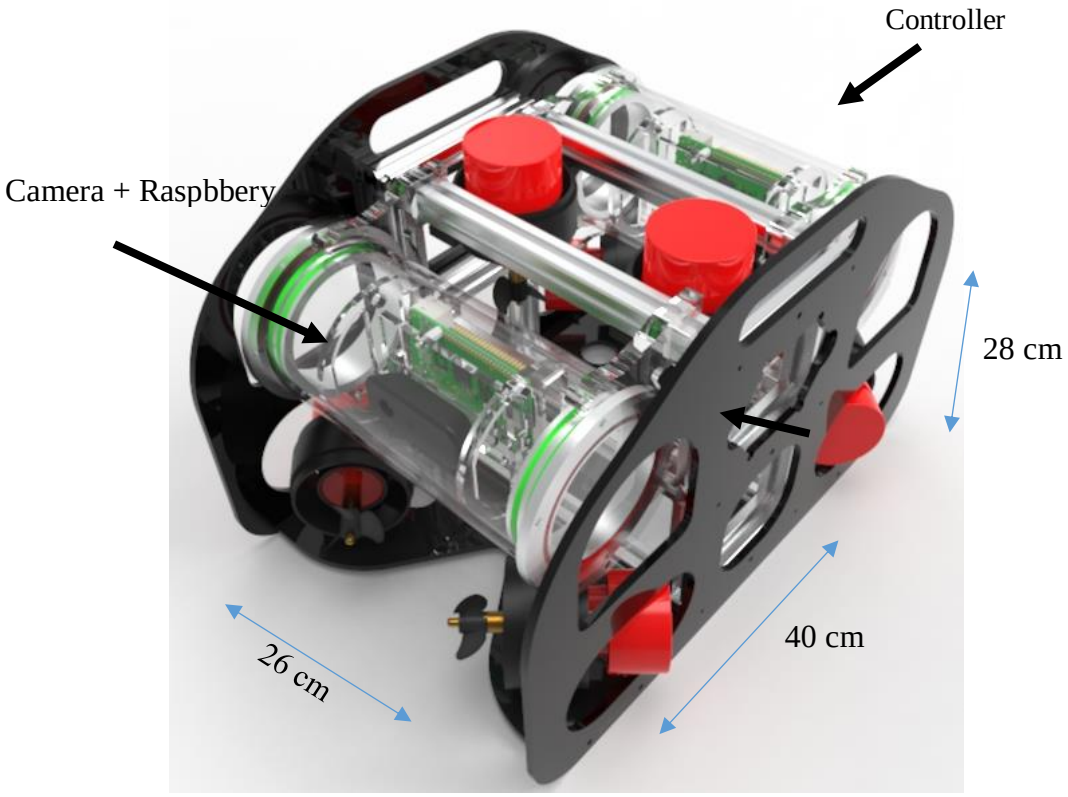
Desain ROV versi 1







Desai ROV versi ke-2



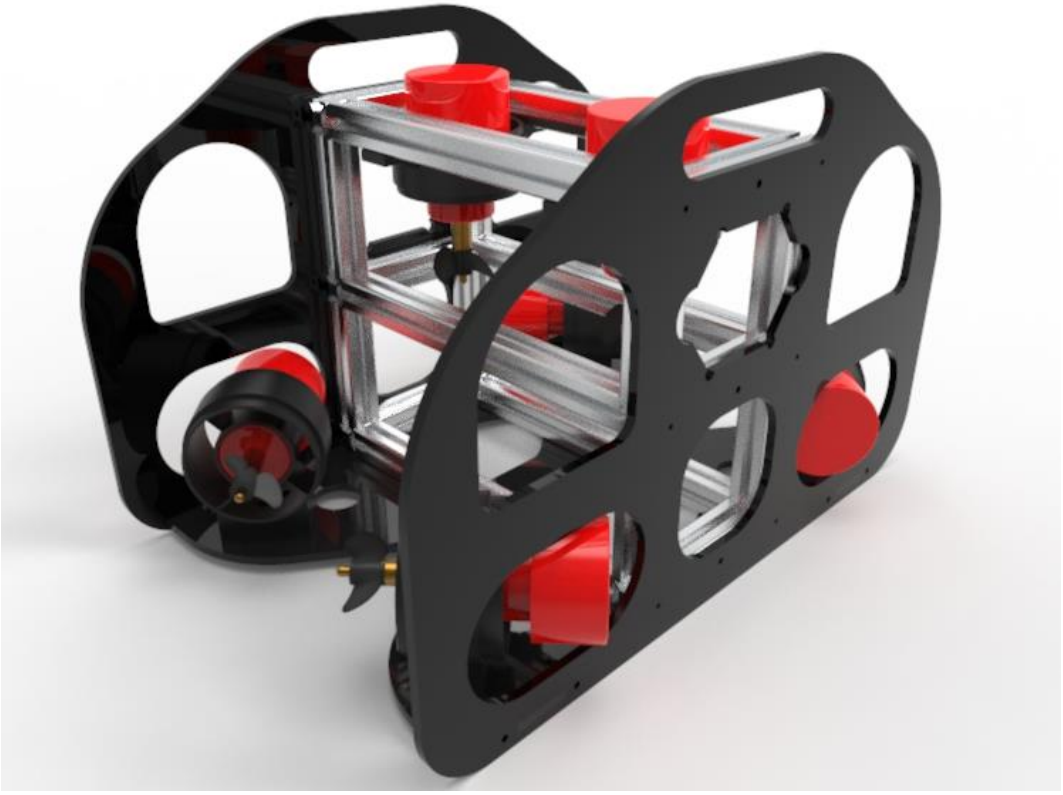
Perakitan Frame

Base menggunakan Alumunium Profile

Dengan base dasar menggunakan Acrylic

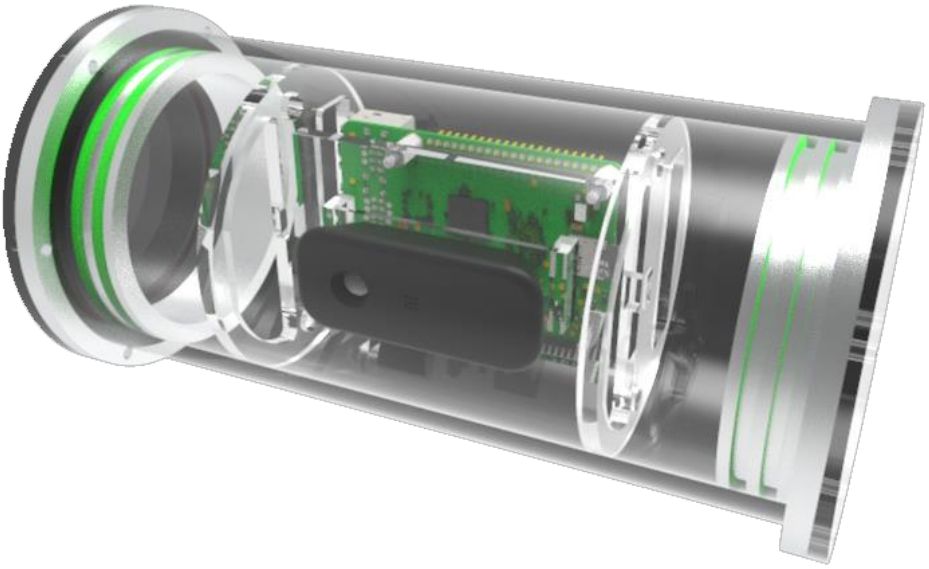
Posisi peletakan motor penggerak

Base samping menggunakan Acrylic



Buoyancy (pelampung beserta rumah komponen)

Terbuat dari tabung acrylic yang ditutupi denagn acrylic yang menggunakan seal o-ring.



RIWAYAT PENULIS

Tri Susanto di lahirkan di Palembang pada 02 Nopember 1988, putra ke tiga dari pasangan Tukiman dan Mudjiati. Penulis memulai pendidikan Taman Kanak-kanak di TK Pembina UNSRI pada tahun 1993-1994 dan Sekolah Dasar di SD Negeri 93 pada tahun 1994 - 2000, kemudian menjalankan sekolah menengah pertama di SMP Negeri 17 Palembang pada tahun 2000 - 2004, selanjutnya penulis melanjutkan di sekolah menengah atas di SMA Srijaya Negara Palembang pada tahun 2004 – 2007, setelah itu penulis menempuh pendidikan tinggi pada program sarjana di Universitas Sriwijaya pada tahun 2007-2012, dan melanjutkan pendidikan program magister Jurusan Teknik Elektro di Institut Teknologi Sepuluh November Surabaya dengan bidang keahlian Teknik Elektronika pada tahun 2014.