TESIS – TE142599

SISTEM PENDERMAGAAN OTOMATIS PADA *REMOTELY OPERATED VEHICLE* DENGAN MENGGUNAKAN KAMERA SEBAGAI UMPAN BALIK POSISI DAN ORIENTASI

MUHAMMAD QOMARUZZAMAN
07111650040009

DOSEN PEMBIMBING
Ronny Mardiyanto, ST., MT., Ph.D.

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

TESIS – TE142599

SISTEM PENDERMAGAAN OTOMATIS PADA *REMOTELY OPERATED VEHICLE* DENGAN MENGGUNAKAN KAMERA SEBAGAI UMPAN BALIK POSISI DAN ORIENTASI

MUHAMMAD QOMARUZZAMAN
07111650040009

DOSEN PEMBIMBING
Ronny Mardiyanto, ST., MT., Ph.D.

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

LEMBAR PENGESAHAN

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

oleh:

Muhammad Qomaruzzaman
07111650040009

Tanggal Ujian : 3 Juli 2018
Periode Wisuda : September 2018

Disetujui oleh:

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

Dekan Fakultas Teknologi Elektro

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

Halaman ini sengaja dikosongkan

PERNYATAAN KEASLIAN TESIS

Dengan ini saya menyatakan bahwa isi keseluruhan Tesis saya dengan judul "**SISTEM PENDERMAGAAN OTOMATIS PADA *REMOTELY OPERATED VEHICLE* DENGAN MENGGUNAKAN KAMERA SEBAGAI UMPAN BALIK POSISI DAN ORIENTASI**" adalah benar-benar hasil karya intelektual mandiri, diselesaikan tanpa menggunakan bahan-bahan yang tidak diijinkan dan bukan merupakan karya pihak lain yang saya akui sebagai karya sendiri.

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

Surabaya, Mei 2018

Muhammad Qomaruzzaman
07111650040009

Halaman ini sengaja dikosongkan

SISTEM PENDERMAGAAN OTOMATIS PADA *REMOTELY OPERATED VEHICLE* DENGAN MENGGUNAKAN KAMERA SEBAGAI UMPAN BALIK POSISI DAN ORIENTASI

Nama mahasiswa : Muhammad Qomaruzzaman
NRP : 07111650040009
Pembimbing : Ronny Mardiyanto, S.T., M.T., Ph.D.

ABSTRAK

Peningkatan gas karbon di udara menyebabkan pemanasan global. Peningkatan temperatur juga terjadi di lautan yang menyebabkan kematian karang secara masal. Kematian karang mengurangi keberagaman spesies dalam air sehingga ekosistem menjadi tidak seimbang dan hancur dengan sendirinya. Untuk mengatasi hal ini, berbagai komunitas penyelam telah membuat upaya restorasi karang dengan menanam terumbu karang yang baru. Demi keamanan, penyelaman membutuhkan persiapan yang matang dan mahal. Oleh karena itu, sejumlah peneliti membuat sistem dengan mengombinasikan *Remotely Operated Vehicle* (ROV) dan kapal pelampung (*buoyant boat*) tanpa awak untuk membantu penyelam dalam usaha restorasi karang. ROV adalah kendaraan bawah air tanpa awak (*Unmanned Underwater Vehicle*) yang terhubung dengan kabel (*tether*) untuk dikendalikan dari jarak jauh. Dermaga (*dock*) adalah tempat untuk mengangkut atau menyimpan ROV yang diletakkan dibawah *buoyant boat*. Pendermagaan otomatis adalah proses menurunkan dan menaikkan ROV secara otomatis dari dermaga. Proses pendermagaan otomatis dilakukan dengan mempertahankan posisi dan orientasi ROV terhadap marka yang ada pada dermaga melalui informasi visual dari kamera. Marka pada dermaga dikenali dengan filter ukuran dan filter bentuk. Tingkat pendeteksian marka dengan ukuran 11x11 cm^2 adalah 80% dengan jarak maksimal 120 cm dan meskipun telah dimiringkan 50^o. Waktu yang dibutuhkan untuk mendeteksi marka tiap cuplikan adalah 0.26 detik. Sudut orientasi dapat diestimasi dengan rata-rata eror 1.75^o dan standar deviasi 0.5^o. Kontrol posisi menggunakan metode kontrol PID berhasil dilakukan dengan *rise time* 9.07 detik. Pendermagaan dengan kamera sebagai umpan balik posisi dan orientasi berhasil dilakukan dengan kesalahan posisi 8.67% dan kesalahan orientasi 7.43% dari kesalahan maksimal. Pendermagaan berlangsung selama 80 detik dengan jarak mula ROV ke dermaga 65 cm.

Kata kunci: ROV; Pendermagaan; Estimasi pose; Pendeteksi marka

Halaman ini sengaja dikosongkan

AUTOMATIC DOCKING SYSTEM ON *REMOTELY OPERATED VEHICLE* USING CAMERA AS POSITION AND ORIENTATION FEEDBACK

By	: Muhammad Qomaruzzaman
Student Identity Number	: 07111650040009
Supervisor(s)	: Ronny Mardiyanto, S.T., M.T., Ph.D.

ABSTRACT

The increasing amount of carbon gas in atmosphere causes global warming. The raising temperature also happened in oceans causing massive death of corals. Coral death reduces biodiversity which leads to the collapse of the ecosystem. To overcome this problem, many diver community have initiated restoration program by planting new corals. In the sake of safety, diving needs a thorough preparations. A system consisted of Remotely Operated Vehicle (ROV) and buoyant boat is developed by scientists to helm coral restoration program. ROV is an unmanned underwater vehicle that is connected by a tether to be controled remotely. Dock is a platform to lock ROV under buoyant boat. Automatic docking is a process to launch and recover ROV form the dock autonomously. Automatic docking is done by maintaining ROV position and orientation to a marka on the dock using visual informatif from a camera. Marka is recognized using shape and size filter. The detection rate of 11x11 cm^2 marka is 80% with maximum distance of 120 cm. Marka is still can be detected even if it's tilted 50^o. The average detection time each in frame is 0.26 s. Marka orientation can be estimated with 1.75^o average error and 0.5^o of standar deviation. Position control using PID control method is done with rise time of 9.07 s. Camera as position and orientation feedback gives 8.67% of position error and 7.43% of orientation error compared to maximum error. The automatic docking process is done in 80 s with 65 cm being the distance of ROV to dock.

Key words: Docking; Pose estimation; Marka detection

Halaman ini sengaja dikosongkan

KATA PENGANTAR

Dengan menyebut asma Allah yang Maha Pengasih lagi Maha Penyayang, Segala puji saya ucapkan kepada Allah SWT tuhan semesta alam. Semoga semua rahmat tercurahkan kepada baginda Muhammad SAW. Tiada manusia yang sempurna seperti Beliau, namun saya berusaha dengan hati yang tulus untuk selalu menyempurnakan dan menyelesaikan tesis ini. Saya sangat berterimakasih kepada Bapak Ronny Mardiyanto sebagai pembimbing saya yang paling sabar dan tidak pernah marah. Saya juga berterimakasih kepada orang tua. Tak lupa saya berterimakasih kepada kakek dan nenek, seluruh jajaran guru dan guru dari guru saya. Terimaksih kepada setiap daun yang setia memeberikan oksigen, lautan yang menyejukkan jiwa, dan gunung yang menghibur dikala penat. Terimakasih kepada seorang atau dua orang yang saya tidak berani sebutkan namanya karena itu rahasia. Dialah yang menemani saya dan memberikan banyak pelajaran kehidupan. Saya berharap penelitian ini dapat menyadarkan betapa bodohnya manusia dan betapa luasnya ilmu Tuhan. Semoga penelitian ini dapat mendekatkan seluruh pihak yang terlibat dalam penelitian, serta pembaca kata pengantar ini kepada Tuhan Yang Maha Esa.

Surabaya, Mei 2018

Penulis

Halaman ini sengaja dikosongkan

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

Halaman ini sengaja dikosongkan

BAB 1
PENDAHULUAN

1.1 Latar Belakang

Peningkatan gas karbon dioksida oleh aktivitas manusia menyebabkan meningkatnya temperatur global. Fenomena ini sering disebut dengan pemanasan global. Para meneliti telah menemukan hubungan antara pemanasan kenaikan temperatur dengan pemutihan karang secara global (*coral bleaching*) [1]. Peningkatan gas karbon dioksida juga menyebabkan asidifikasi laut, yaitu kondisi dimana air laut menjadi semakin asam. Kualitas air laut juga semakin diperburuk dengan adanya limbah yang dibuang ke laut. Kerusakan karang juga mengurangi keragaman hayati laut sehingga merusak keseimbangan ekosistem laut [2]. Hal-hal tersebut merupakan faktor yang teridentifikasi mempercepat kerusakan karang. Meskipun tidak merusak karang secara fisik, faktor tersebut cukup untuk mematikan fungsi karang yaitu sebagai rumah atau tempat berlindung berbagai macam spesies laut.

Karang laut adalah komponen laut yang sangat penting untuk dilestarikan. Karang laut merupakan pelindung pantai dari ombak, pengontrol deposito kalsium dan nitrogen, tempat berlindung berbagai spesies [3]. Kerusakan karang merupakan kabar buruk bagi perekonomian masyarakat. Rusaknya karang dapat mengakibatkan berkurangnya keragaman hayati sehingga mengurangi pendapatan nelayan. Ribuan potensi pariwisata akan hilang beserta puluhan ribu pekerja yang ada didalamnya apabila hal tersebut terus menerus lalai diperhatikan. Oleh karena itu, berbagai usaha konservasi karang laut telah dilakukan untuk mempertahankan simbiosis laut.

Usaha konservasi karang yang telah dilakukan adalah menanam terumbu karang baru. Pemerintah pun melakukan usaha restorasi terumbu karang dan perlawanan terhadap pemutihan karang secara global. Kementrian Kelautan dan Perikanan (KKP) memberikan panduan dalam pemantauan kesehatan karang [4]. Pemerintah melakukan usaha tersebut karena terumbu karang memiliki pengaruh yang signifikan terhadap pemasukan sektor paiwisata. Pemerintah mendukung

pengumpulan data tentang ketahanan dan ketangguhan karang karena sangat dibutuhkan oleh komunitas global dan juga lokal.

Penanaman terumbu karang saat ini dilakukan secara tradisional oleh penyelam. Tantangan alam yang setiap saat harus dihadapi para penyelam adalah perubahan arus yang tidak menentu, dinding batu yang tajam, dan berbagai spesies beracun dan berbahaya didalam air. Termasuk dalam risiko penyelaman adalah penyakit fisik yang biasa dialami penyelam yaitu kram, penyakit pernapasan, perubahan suara, bahkan gangguan pendengaran maupun penglihatan [5]. Tantangan tersebut sewaktu-waktu dapat mengancam keselamatan nyawa penyelam.

Divers Alert Network (DAN) merupakan organisasi yang mengumpulkan data kecelakaan penyelam. DAN melaporkan adanya 127 kematian penyelam sejak tahun 2015 [5]. Hal ini merupakan pengingat akan pentingnya persiapan yang baik sebelum menyelam. Persiapan sebelum menyelam meliputi kondisi fisik, tangki oksigen, peralatan keselamatan, dan bahkan sahabat menyelam (*buddy diving*). Oleh karena itu, untuk mengurangi risiko menyelam, para peneliti membuat suatu jenis robot bawah air, yaitu *Remotly Operated Vehicle* (ROV).

ROV adalah jenis kendaraan bawah air tanpa awak (*Unmanned Underwater Vehicle*) yang terhubung dengan kabel dan dikendalikan dari jarak jauh oleh operator [6]. Kabel pada ROV digunakan sebagai media komunikasi data dan penyalur daya. Jarak maksimal ROV bergantung pada panjang kabel. Kabel digunakan karena sinyal frekuensi radio tidak dapat menembus air yang memiliki tingkat atenuasi tinggi. Frekuensi radio yang rendah dapat menembus air karena memiliki gelombang yang lebih panjang. Namun dengan berkurangnya frekuensi, kecepatan transmisi data akan berkurang, sedangkan ROV membutuhkan kecepatan transmisi yang tinggi, terutama untuk mengirimkan sinyal multimedia [6].

Pada umumnya ROV digunakan oleh fasilitas pengeboran minyak lepas pantai. Saat ini, ROV sudah dikembangkan dalam berbagai ukuran. Pada perkembangannya ROV telah dilengkapi berbagai alat seperti lengan robot. Perkembangan ROV saat ini juga meliputi kontrol manuver, visibilitas kamera, dan juga kendali otomatis.

Penelitian ini merupakan bagian dari serangkaian penelitian tentang pemantauan dan restorasi karang menggunakan bantuan robot. Pada penelitian tersebut didesain suatu kapal pelampung (*buoyant boat*) yang dapat dioperasikan dari jarak jauh. Kapal pelampung juga dilengkapi dengan panel surya serta *Global Positioning System* (GPS) sehingga dapat beroperasi secara otomatis selama berhari-hari. Dibawah kapal ini lah diletakkan sebuah ROV untuk memantu kondisi bawah air. Karena *buoyant boat* didesain tanpa awak, maka tidak ada operator yang menaikkan dan menurunkan ROV dari *buoyant boat*. Oleh karena itu dibutuhkan sistem pendermagaan otomatis untuk melengkapi sistem tersebut.

Dermaga (*dock*) digunakan sebagai alat untuk mengangkut atau menyimpan. ROV. Dermaga juga digunakan untuk memudahkan mobilitas ROV dikala kapal bergerak. Pada penelitian ini dermaga ditempatkan dibawah kapal tanpa awak yang dapat dikendalikan dari jarak jauh. Pendermagaan otomatis dilakukan untuk mempermudah proses kembalinya ROV pada dermaga yang berada dibawah kapal tanpa awak. Kamera dipilih sebagai sensor untuk mengenali dermaga. Pada jarak dekat kamera dapat digunakan untuk meningkatkan ketelitian pengestimasi posisi dibandingkan sensor akustik.

1.2 Rumusan Masalah

Berdasarkan latar belakang yang telah disebutkan, rumusan masalah secara global dari penelitian ini adalah untuk menaikkan dan menurunkan ROV secara otomatis pada dermaga. Masalah spesifik dari penelitian ini adalah:

1. Bagaimana cara ROV mengenali dermaga?
2. Bagaimana cara ROV kembali ke dermaga
3. Bagaimana cara ROV memposisikan dirinya pada dermaga?

1.3 Tujuan

Tujuan utama dari penelitian ini adalah membuat metode yang dapat digunakan ROV untuk naik dan turun dari dermaga secara otomatis. Tujuan spesifik dari penelitain ini adalah:

1. Memberikan kemampuan pada ROV untuk mengenali dermaga.

2. Memberikan kemampuan pada ROV sehingga dapat memposisikan dirinya pada dermaga

Berdasarkan tujuan tersebut diharapkan penelitian ini dapat memberikan manfaat diantaranya:

1. Memudahkan dan mengurangi risiko penyelam konservasi karang.
2. Mengurangi biaya yang dibutuhkan untuk konservasi karang.
3. Memberikan sumbangan pengetahuan terutama dalam bidang elektronika dalam pengembangan ROV.

1.4 Batasan Masalah

Pendermagaan otomatis yang didesain pada penelitian ini digunakan untuk perairan dengan kedalaman maksimal 10 meter. Pendermagaan dilakukan pada kondisi cuaca cerah. Penelitian ini hanya membahas pendermagaan satu ROV pada satu dermaga.

1.5 Kontribusi

Kontribusi yang dihasilkan pada penelitian ini adalah sebagai berikut:

1. Penerapan kamera sebagai sensor visual untuk pendermagaan otomatis.
2. Perumusan metode pengenalan marka yang dilengkapi dengan pengestimasi jarak dan orientasi marka secara cepat.
3. Perancangan marka visual yang dapat dikenali dalam air dalam kondisi perairan yang terang maupun gelap.

BAB 2
KAJIAN PUSTAKA

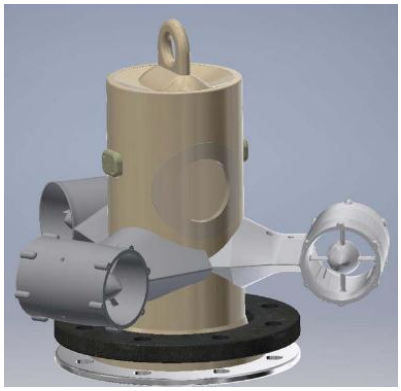
Pada bagian ini akan dijelaskan berbagai kajian pustaka yang berisi tentang referensi penelitian sebelumnya yang menjadi acuan penulis. Pada bagian ini juga dijelaskan dasar teori yang digunakan penulis dalam penyusunan penelitian.

2.1 Kajian Penelitian Terkait

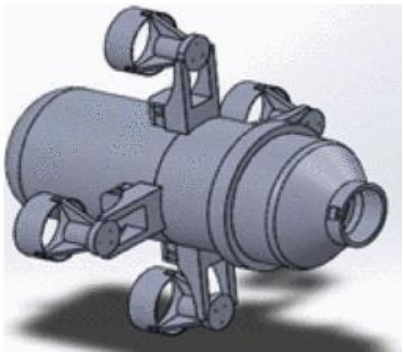
Penelitian ini berkaitan dengan penelitian-penelitian sebelumnya. Beberapa penelitian tersebut dapat dijadikan acuan dalam menyelesaikan permasalahan dalam penelitian ini. Di antara penelitian tersebut adalah penelitian tentang desain ROV, penelitian tentang tipe dermaga, dan penelitian tentang pendermgaan otomatis.

2.1.1 Penelitian tentang desain ROV

Beberapa peneliti membuat desain ROV sebagai sarana inspeksi pertanian dalam air dengan hanya menggunakan tiga penggerak [7]. Penggerak diletakkan sejajar membentuk segitiga. Penggerak tersebut dapat digunakan ROV untuk berputar, maju atau mundur, dan bergerak ke samping. ROV dikaitkan dengan *tether* yang kuat yang juga difungsikan untuk menaikkan dan menurunkan ROV. Tanpa menggunakan penggerak naik dan turun, ROV bermanuver dengan mengayunkan diri pada *tether*. Sistem ini hemat biaya dan efektif untuk observasi, namun ROV akan kesulitan mempertahankan posisinya sehingga sulit untuk mendapatkan sudut pandang observasi yang baik. Gambar ROV dari penelitian tersebut dapat dilihat pada Gambar 2.1.



Gambar 2.1 ROV dengan untuk observasi pertanian dalam air [7]

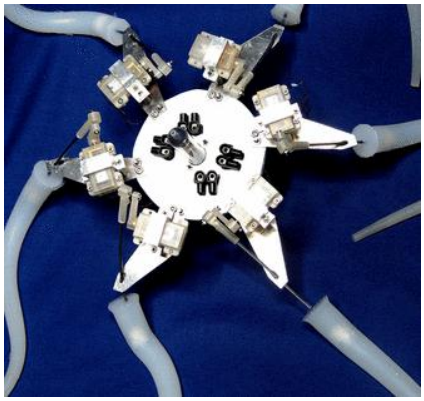


Gambar 2.2 ROV dengan desain menyerupai torpedo tanpa sirip [8]

Terdapat pula desain ROV yang menyerupai torpedo tanpa sirip [8] seperti pada Gambar 2.2 ROV digerakkan ke depan dengan menggunakan empat penggerak. Dengan penggerak tersebut, ROV dapat bergerak menuju segala arah dengan berbelok terlebih dahulu. ROV dengan desain ini sangat lincah dalam bermanuver dan cocok untuk tugas pengejaran. Kelemahan ROV ini adalah kesulitan dalam mempertahankan muka ROV saat diam. Hal tersebut kurang cocok untuk misi pengambilan gambar tetap atau diam.

Penambahan kapabilitas manuver seringkali berkaitan dengan penambahan jumlah penggerak. Beberapa peneliti menggunakan tujuh penggerak untuk mencapai 6 derajat kebebasan [9]. ROV dapat digerakkan secara translasi dan rotasi meliputi seluruh sumbu x, y, dan z. Desain ROV ini dapat digunakan untuk memenuhi kebutuhan pengambilan gambar secara stabil. Desain ROV yang digunakan pada penelitian ini digunakan ROV dengan jumlah derajat kebebasan yang sama dengan ROV tersebut. Gambar ROV yang digunakan pada penelitian tersebut dapat dilihat pada Gambar 2.3.

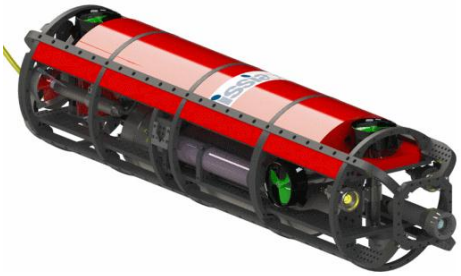
Gambar 2.3 ROV dengan 6 derajat kebebasan [9].



Gambar 2.4 ROV dengan bahan lunak [10].

Beberapa peneliti juga mengembangkan ROV dengan menggunakan bahan yang lunak untuk merakit ROV [10]. ROV berbahan lunak dapat bertahan lebih baik pada tekanan tinggi karena dibuat dari bahan yang tidak mudah pecah. ROV berbahan lunak juga dapat didesain sehingga terlihat lebih natural. Dengan bentuk ROV yang lebih menyerupai bentuk natural, ROV dapat mengambil gambar lebih dekat dengan ikan atau makluk hidup lainnya. Namun kelemahan dari ROV tipe ini adalah arah geraknnya yang terbatas. Kecepatan gerak ROV tipe ini juga lambat sehingga jenis ROV ini hanya dibuat dalam bentuk yang kecil. Ilustrasi dari ROV ini dapat dilihat pada Gambar 2.4.

Ada pula ROV yang didesain dengan bentuk silindris memanjang [11]. ROV ini didesain untuk penelitian dibawah permukaan es. ROV dengan bentuk silindris dapat dengan mudah bererak mengguling ke kanan dan ke kiri dengan cepat. Bentunya yang panjang dapat memperlambat gerakan mengangguk sehingga kamera yang diletakkan didepan ROV dapat melihat gambar dengan stabil. Kekurangannya adalah ROV dengan tipe ini mudah sekali terbalik, sehingga membuthkan kontrol tambahan. Gambar dari ROV tipe ini dapat dilihat pada Gambar 2.5

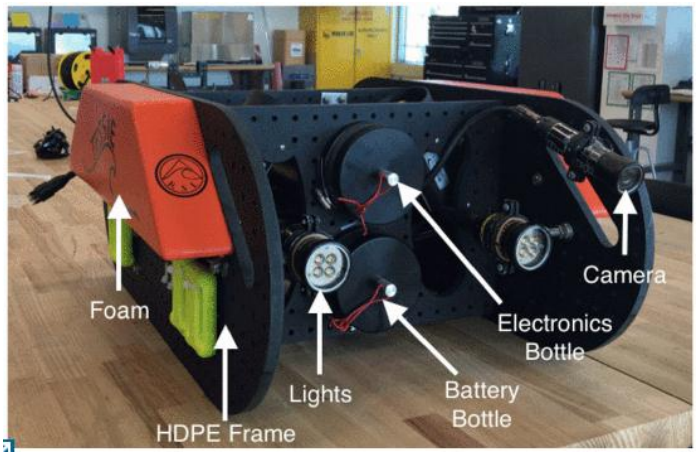


Gambar 2.5 ROV dengan bentuk silindris memanjang [11].

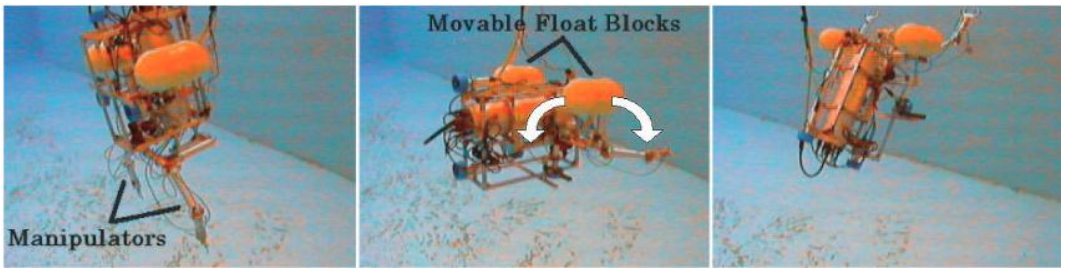
Gambar 2.6 ROV untuk inspeksi bangunan dalam air [12].

Sebuah ROV telah didesain khusus untuk inspeksi struktur bawah air [12]. ROV ini didesain untuk merekam gambar dengan kualitas tinggi pada perairan yang berbahaya bagi peyelam. Sebuah sumber cahaya dipasang untuk menerangi pengambilan gambar. ROV dikontrol secara manual untuk mempermudah menjangkau daerah yang sempit. Untuk mendukung navigasi ROV dilengkapi dengan GPS, sensor kedalaman dan juga DVL. Kelemahan dari ROV tersebut adalah kontrol kestabilan dari pose ROV yang masih manual. Kestablian ROV sangat mempengaruhi kestabilan gambar yang direkm ROV. Gambar dari ROV tersebut dapat dilihat pada Gambar 2.6.

Sejumlah mahasiswa dari Santa Clara University menunjukkan bahwa sangat dimungkinkan untuk membuat ROV dengan biaya minimal. ROV yang dibuat pada penelitian tersebut diaplikasikan untuk ke dalam laut dangkal [13]. Plastik HDPE merupakan bahan pilihan dari ROV. Perangkat elektronik beserta baterai berada didalam ROV dengan pelindung yang terbuat dari PVC. Komunikasi dari ROV menggunakan protokol serial RS-485. ROV memiliki konfigurasi penggerak yang digunakan untuk kontrol geleng dari ROV (*Yaw*). ROV tidak memiliki penggerak untuk kontrol sudut guling dan sudut angguk. Hasil dari desain ROV dapat dilihat pada Gambar 2.7.



Gambar 2.7 ROV untuk observasi laut dangkal [13].



Gambar 2.8 ROV dengan pelampung yang dapat diputar [14].

Kestabilan ROV juga dapat diperoleh dengan memindahkan posisi titik berat ROV. Sejumlah peneliti telah mendesain ROV dengan pelampung yang dapat diputar sehingga dapat memutar balik ROV [14]. Pelampung diputar dengan motor sehingga menghasilkan kontrol sudut angguk dari ROV. Dengan kamera berada di depan, ROV dapat dikontrol sehingga mendongak ataupun melihat kebawah dengan kamera yang sama. ROV ini menggunakan dua pelampung sehingga sudut guling dari ROV stabil secara mekanis. Kelemahan dari metode kontrol ini adalah dibutuhkan suatu pemberat dan juga pelampung secara bersamaan. Desain dari ROV ini dapat dilihat pada Gambar 2.8.

2.1.2 Penelitian tentang tipe dermaga

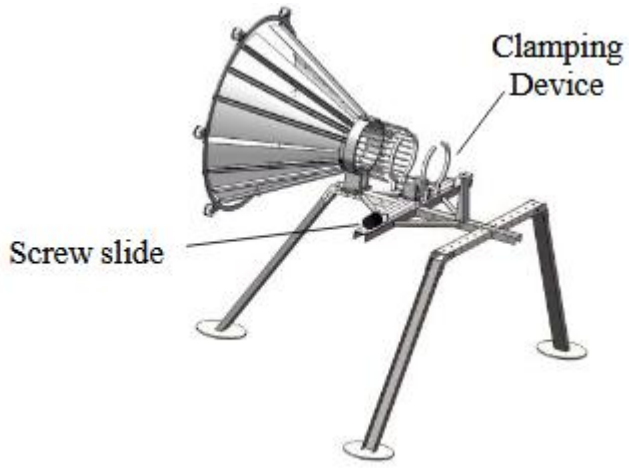
Pendermagaan otomatis sangat umum digunakan pada AUV (*Autonomous Underwater Vehicle*). Metode dermaga yang digunakan pun bermacam-macam dan bersesuaian dengan desain dermaga. Contoh dari metode pendermagaan adalah pendermagaan dengan dermaga berbentuk corong (*funnel docking*), pendermagaan dengan pencapit (atau *latch docking*), atau pendermagaan dengan batang pengunci

(*pole docking*) [15]. Pada penelitian ini digunakan pendermagaan menggnakan elektro magnet yang mirip dengan sistem *latch docking*. Hal ini dilakukan karena paling sesuai dengan cara pendermagaan dari bawah air ke permukaan.

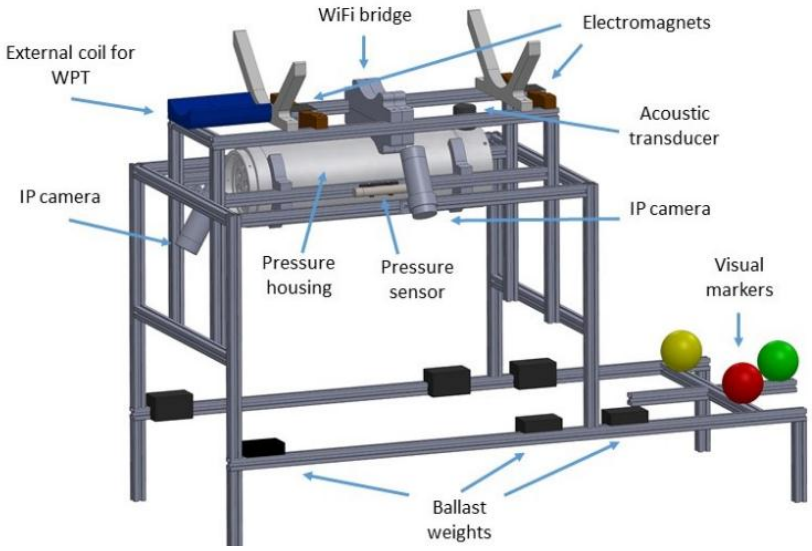
Dermaga model corong adalah yang paling sering digunaan kerana dapat digunakan untuk ROV dalam berbagai bentuk dan ukuran. Dermaga yang memiliki pencapit atau pengunci adalah dermaga yang memiliki tingkat keamanan paling baik, namun dibutuhkan akurasi yang lebih tinggi dan desain yang biasanya tidak terstandardisasi. Beberapa peneliti menggabungkan dermaga corong dengan dermaga bercapit untuk mengatasi masalah ini [16]. Dermaga pada penelitian tersebut dapat dilihat pada Gambar 2.9.

Penelitian pendermagaan terhadap panggung (*platform*) bergerak juga telah dilakukan. Beberapa peneliti membuat sistem pendermagaan AUV terhadap kapal selam bergerak [17]. Pada penelitian ini digunakan panggung yang diam. Sebelum panggung bergerak, ROV diasumsikan sudah berada dalam dermaga.

Salah satu dermaga yang telah berkembang menambahkan marka sebagai tambahan informasi visual untuk dikenali [18]. Sebuah dermaga yang diletakkan di dasar laut, beserta marka untuk dikenali kamera yang menghadap kebawah. Dermaga mengunci ROV ditempat dengan empat buah elektro magnet. Marka yang digunakan pada penelitian tersebut berupa bola tiga warna. Dermaga pada penelitian tersebut dapat dilihat pada Gambar 2.10.



Gambar 2.9 Dermaga corong dengan penjepit [16].

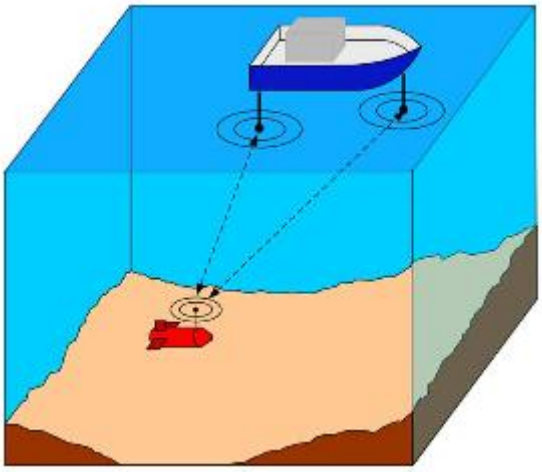


Gambar 2.10 Dermaga dengan marka visual dan penjepit dari elektro magnet [18].

2.1.3 Penelitian tentang pendermagaan

Pendermagaan otomatis dapat dilakukan dengan cara menggulung kabel ROV langsung ke dermaga [19]. Pendermagaan otomatis metode ini membutuhan kabel yang kuat. Keuntungan dalam menggunakan metode ini adalah kebel dari ROV bisa dijaga agar tetap pendek. Metode ini juga dapat menyelamatkan ROV apabila terseret arus.

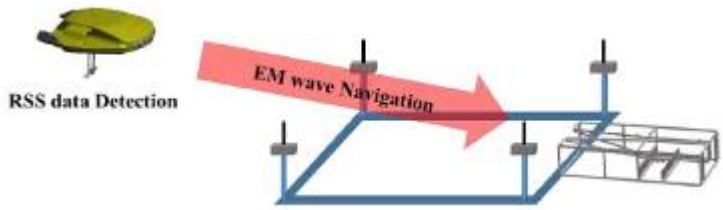
Pendermagaan otomatis biasanya menggunakan pemancar isyarat akustik (*accoustic beacon*) [20]. Pemancar ini digunakan untuk mengetahui posisi relatif ROV terhadap dermaga. Pemancar isyarat akustik juga dapat dipasang dibeberapa titik untuk membantu ROV mengetahui posisi terhadap lingkungan sekitar pemanacar. Cara ini dapat mencakup area yang luas dan biasa digunakan pada ROV pemasang kabel bawah laut. Ilustrasi pendermagaan manggunakan pamancar akustik dapat dilihat pada Gambar 2.11.



Gambar 2.11 Pendermagaan menggunakan pemancar isyarat akustik [20].

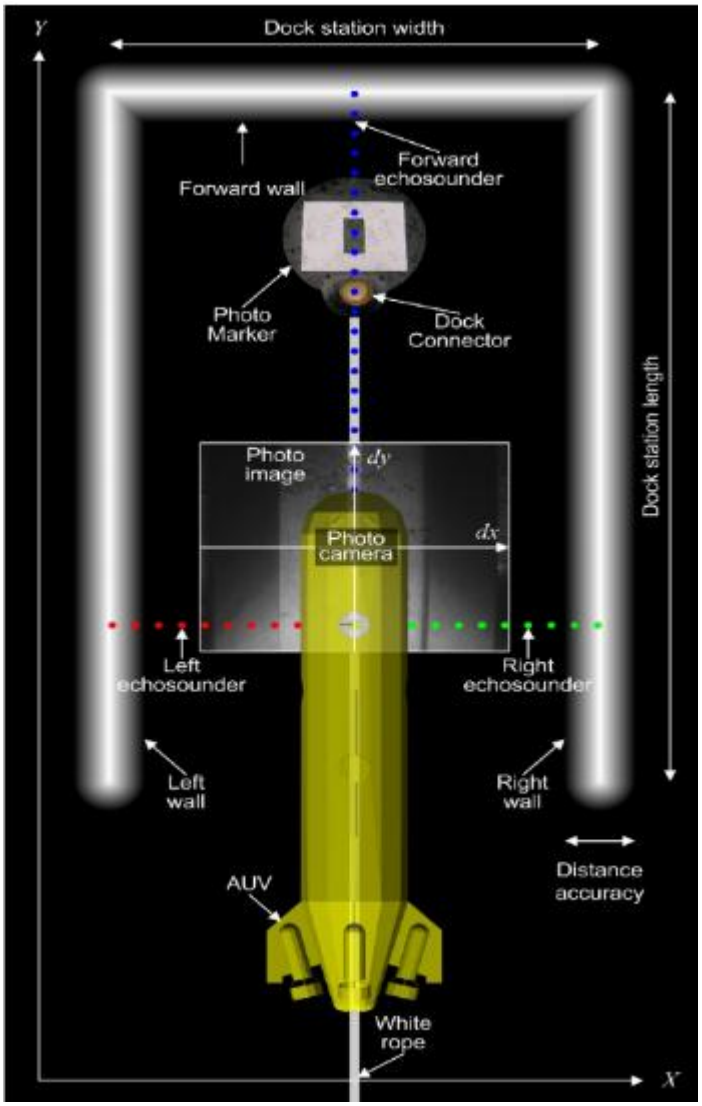
Salah satu teknik yang digunakan dalam pendermagaan otomatis dengan logika fuzzy. Kumpulan sensor jarak akustik dipasang pada dermga. Kemudian dermaga dibagi menjadi beberapa bagian berdasarkan sensor jarak. Berdasarkan data jarak masing-masing sensor, lokasi ROV terhadap dermaga dapat diperkirakan. Dermaga dibagi menjadi beberapa bagian. Setiap bagian memiliki perintah sikap yang harus diambil ROV apabila ROV berada pada bagian tersebut [21].

Medan elektromagnet juga dapat digunakan untuk menandai dermaga [22]. Arah dari dermaga dapat diketahui berdasarkan kekuatan sinyal elektromagnet. Metode pendermagaan otomatis dengan sinyal elektromagnet hanya dapat dilakukan pada jarak dekat. Kelemahan dari metode ini adalah dibutuhkan banyak pemancar untuk menambah resolusi dari pendermagaan. Hal tersebut dikarenakan sinyal elektro magnet tidak dapat merambat dengan baik didalam air. Ilustrasi dari penelitian tersebut dapat dilihat pada Gambar 2.12



Gambar 2.12 Pendermagaan menggunakan sinyal elektro magnet [22].

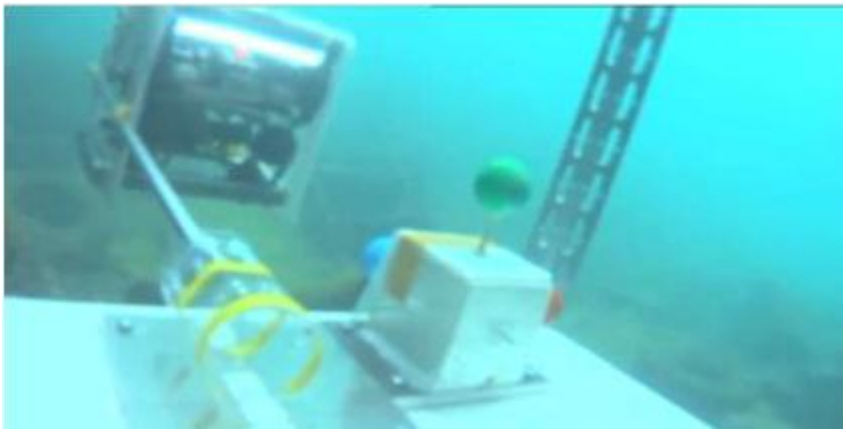
Pendermagaan juga dapat dilakukan menggunakan kamera dan juga *echosounder* secara bersamaan. Beberapa peneliti manggunakan kamera untuk menentukan arah gerak (*heading*) ROV [23]. ROV dapat mengetahui arah gerak dengan mengikuti garis putih menuju dermaga. Garis putih tersebut merupakan data visual yang ditangkap kamera untuk dioalah. *Echosounder* digunakan supaya ROV tidak menabrak dinding dermaga. *Echosounder* juga dipasang pada titik buta kamera untuk antisipasi benturan. Ilustrasi dari penelitian tersebut dapat dilihat pada Gambar 2.13.



Gambar 2.13 Pendermagaan menggunakan kamera dan juga *echosounder* [23].

Cara lain dalam pendermagaan ROV adalah menggunakan kamera. Pada umumnya, pendermagaan menggunakan metode ini melibatkan suatu marka yang dipasang pada dermaga untuk dikenali oleh ROV. Salah satu permasalahan dari menggunakan kamera sebagai alat untuk mengenali dermaga adalah pengestimasian jarak dari ROV ke dermaga. Untuk mengatasi masalah ini, sebuah penelitain telah dibuat sehingga ROV dapat memperkirakan posisinya sekaligus posenya pada lingkungan tertentu [24]. Penelitian tersebut dilakukan dengan cara mengambil gambar setiap sudut pandang dari setiap titik pada lingkungan tertentu. Setiap gambar memiliki data pose dan posisi saat pengambilan gambar. Dengan demikian, apabila ROV mendapati gambar yang mirip dengan salah satu gambar yang telah diambil, ROV dapat segera memutuskan posisi dan pose saat gambar itu diambil. Metode seperti ini membutuhkan kecepatan untuk mencocokkan gambar dengan data gambar yang diambil. Data gambar yang baru harus dikumpulan sebelum ROV dapat menyelam pada lingkungan yang baru. Metode ini akan sangat cocok digunakan pada lingkungan yang tidak berubah-ubah.

Beberapa peneliti menggunakan dua buah kamera (*stereo camera*) untuk mengestimasikan jarak [25]. Sebuah marka 3D di pasang pada dermaga. ROV telah menyimpan gambar marka tersebut dari beberapa sisi. Dengan membandingkan gambar marka dari basis data dan gambar yang tampak oleh kamera, ROV dapat mengestimasikan pose 3D relatif terhadap marka. Metode ini sangatlah tepat digunakan untuk pendermagaan otomatis karena dapat memperkirakan jarak dan posisi. Akan tetapi metode ini membutuhkan kemampuan komputasi yang tinggi, sehingga kurang tepat diterapkan pada robot kecil. Ilustrasi dari penelitian tersebut dapat dilihat pada Gambar 2.14



Gambar 2.14 Pendermagaan menggunakan kamera stereo untuk mengestimasikan jarak [25].

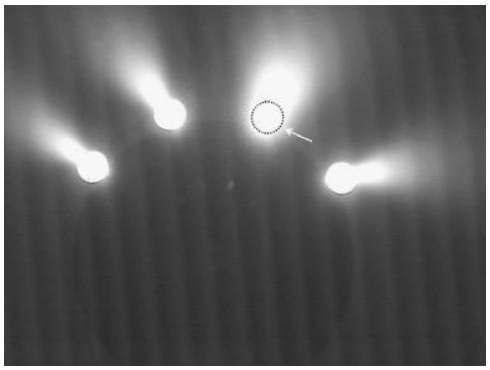
Perkembangan teknologi elektronik telah memungkinkan untuk membuat kamera *light field*. Kamera ini adalah jenis kamera baru yang dapat menangkap gambar secara tiga dimensi. Kamera ini terlihat memiliki lensa tunggal, namun ada banyak lensa dan sensor didalamnya. Setiap *pixel* (elemen gambar) dari gambar yang diambil memiliki data jarak terhadap kamera. Beberapa peneliti memanfaatkan fitur ini untuk keperluan pendermagaan otomatis [26]. Dengan kamera ini, robot dapat membuat peta dari lingkungan sekitar serta letak dermaga pada peta tersebut. Robot dapat mengetahui posisi diri dan posisi dermaga dari peta yang dibuat. Pada penelitian tersebut dermaga diidentifikasi dengan warnanya yang unik. Kelemahan dari penelitian ini adalah perangkat yang mahal karena merupakan teknologi baru.

Selain menggunakan teknologi kamera ganda atau pemetaan, pendermagaan otomatis juga dapat dilakukan menggunakan marka khusus [27]. Marka tersebut didesain supaya mudah dibedakan dengan lingkungan sekitar. Beberapa peneliti memanfaatkan warna unik untuk membedakan marka dengan objek disekitarnya [28]. Untuk mengaplikasikan metode ini, warna yang tepat dan unik harus dipilih. Filter yang tepat untuk warna tersebut juga harus dipilih. Perubahan pencahayaan juga dapat mengubah warna tersebut nampak di kamera. Marka yang digunakan pada penelitian tersebut dapat dilihat pada Gambar 2.15.

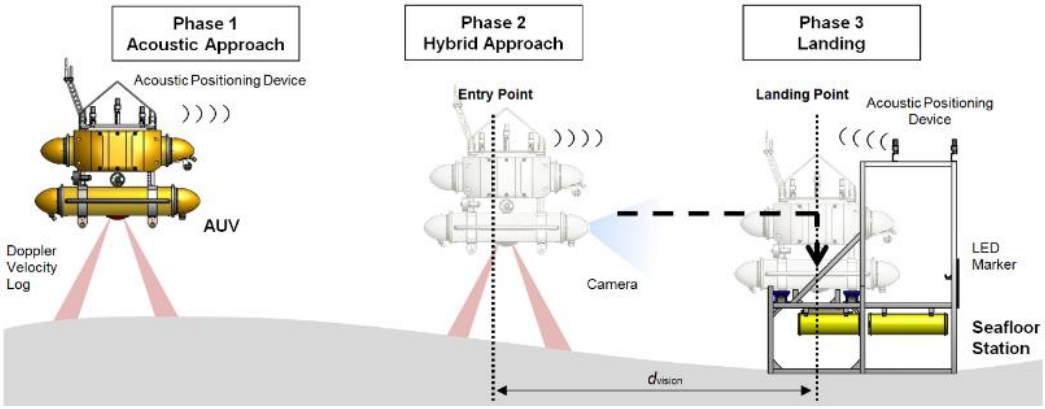
Gambar 2.15 Pendermagaan menggunakan marka dan filter HSV [28].

Perubahan intensitas pencahayaan dapat diatasi dengan sumber cahaya mandiri. Terdapat pula penelitian yang memanfaatkan marka yang terdiri dari lima buah lampu [29]. Dengan cara tersebut marka tidak terpengaruh oleh pencahayaan sekitar. Permasalahan dari menggunakan lampu sebagai marka adalah intensitas lampu yang terlalu terang. Dengan intensitas lampu yang terlalu terang, air disekitar lampu menjadi berpendar dan bentuk marka tidak lagi dapat dikenali. Marka yang nampak berpendar pada penelitian tersebut dapat dilihat pada Gambar 2.16.

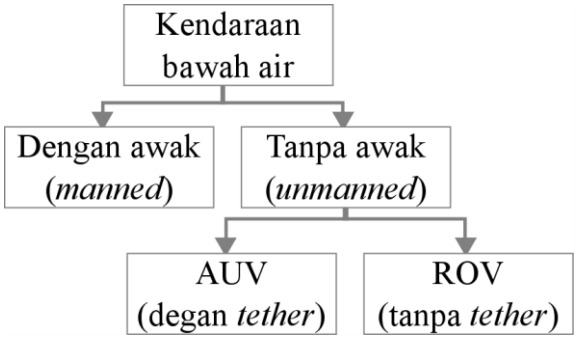
LED juga dapat digunakan sebagai penanda dermaga. Beberapa peneliti memanfaatkan DVL dan juga kamera sebagai sensor pendermagaan otomatis [30]. DVL digunakan saat ROV berada pada jarak yang jauh dari dermaga dan marka digunakan pada jarak dekat. Marka LED dikenali hanya menggunaan transformasi morfologi. Transformasi morfologi sangat begantung pada kondisi lingkungan. Oleh karena itu metode ini cocok hanya ketika tidak terdapat banyak *noise* visual pada lingkungan tersebut. Ilustrasi pendermagaan pada penelitian tersebut dapat dilihat pada Gambar 2.17.



Gambar 2.16 Marka yang terlalu terang dan menyebabkan lingkungan sekitar berpendar [29].



Gambar 2.17 Proses pendermagaan menggunakan marka LED dan pemancar akustik [30].



Gambar 2.18 Klasifikasi kendaraan bawah air

2.2 Teori Dasar

Terdapat beberapa kajian yang menjadi dasar teori pada penelitin ini. Dasar teori tersebut aan dijelaskan pada bagian ini.

2.2.1 ROV (*Remotely Operated Vehicle*)

Pada dasarnya, kendaraan bawah air (*underwater vehicle*) dibagi menjadi dua, yaitu kendaraan bawah air dengan awak (*manned*) dan kendaraan bawah air tanpa awak (*unmanned*). Kendaraan dalam air dengan awak contohnya adalah kapal selam. Diagram pembagian .jenis kendaraan bawah air dapat dilihat pada Gambar 2.1.

Kendaraan bawah air tanpa awak sendiri dibagi menjadi dua yakni dengan kabel (*tethered*) dan tanpa kabel (*non-tethered*). Kendaraan bawah air tanpa kabel biasa disebut dengan AUV (*Autonomous Underwater Vehicle*). Biasanya AUV tidak memiliki kapabilitas untuk dikontrol secara manual, melainkan dikontrol

dengan program yang telah didefinisikan sebelumnya. Kendaraan bawah air tanpa kabel biasa disebut dengan ROV. Kendaraan bawah air dengan kabel inilah yang biasa disebut sebagai ROV. Bentuk ROV paling sederhana adalah sebuah kamera dengan penutup tahan air yang mempunyai pendorong sebagai penggerak dan dengan kabel yang terhubung kepermukaan sebagai media transmisi sinyal video dan pengontrol [6].

2.2.2 Tipe ROV

ROV dapat dibagi menjadi beberapa tipe berdasarkan ukuran. Ukuran dari ROV menentuan tujuan dari penggunaanya. Ukuran dari ROV juga mempengaruhi komponen didalamnya serta komponen penunjang yang lain. ROV dangan ukuran yang besar tentu saja membutuhkan alat tambahan untuk mengeluarkannya dari dalam laut.

Observation class ROVs (OCROV) adalah ROV dengan ukuran terkecil hingga ROV dengan bobot 100 kg. Sesuai dengan namanya, ROV ini biasa digunakan untuk observasi. ROV ini dapat digunakan untuk membantu penyelam atau bahkan menggantikannya. Kemampuan selam ROV ini tidak melebihi 300 m dengan kendala utama yaitu tekanan air. ROV ini didesain supaya tidak memiliki bobot didalam air. ROV ini dapat dimasukkan atau diangkat dalam air tanpa alat bantuan.

Mid-sized ROVs (MSROV) adalah ROV dengan ukuran kerja ringan (*light work class*). Bobot ROV dari 100 kg sampai dengan 1000 kg. Daya yang digunakan pada ROV ini adalah listrik AC. Motor yang digunakan juga mendapatkan daya dari listrik AC. Dengan bobot yang berat, ROV ini membutuhkan alat untuk menurunkan dan menaikkannya dari dalam air yang disebut dengan LARS (*Launch and Recovery System*). TMS (*Tether Management System*) yang merupakan penggulung *tether* dan juga kurungan penyimpan merupakan fitur yang dibutuhkan sebagai pendukung.

Work class ROVs (WCROV) adalah ROV yang biasanya membutuhkan daya dari tegangan listrik AC diatas 3000 V. Daya ini diambil langsung dari kapal diatanya. Daya yang digunakan langsung digunakan pada komponen mekanik seperti hidrolik dan juga motor penggerak.

Special-use vehicles adalah jenis ROV yang tidak masuk dalam kategori sebelumnya. ROV ini tidak memiliki kemampuan berenang, seperti merangkak di dasar laut. ROV ini dibuat dengan tujuan khusus.

Pada penelitian ini dibuat ROV dengan ukuran kecil, sehingga tergolong sebagai OCROV. Namun terinspirasi dari TMS pada kelas MSROV, pada penelitian ini dibuat dermaga untuk ROV. Fungsi TMS selain untuk mengatur panjang kabel *tether* adalah untuk menyimpan ROV.

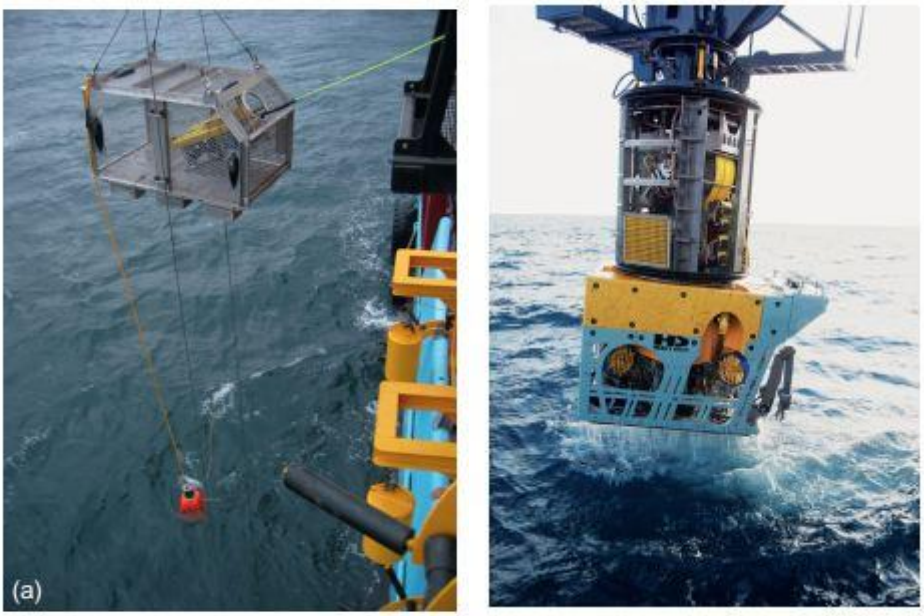
2.2.3 Dermaga (*dock*)

Dock atau *docking station* adalah istilah yang biasa digunakan pada alat elektronik sebagai tempat untuk pengisian daya. Arti *dock* secara harfiah adalah dermaga atau tempat kapal bersandar. Pada penelitian ini dermaga difungsikan dan diartikan sebagai tempat bersandar ROV dibawah kapal induk sehingga ROV dapat bergerak bersama kapal induk didalam dermaga.

Pada umumnya ROV tidak memiliki sistem dermaga. Dermaga lebih sering digunakan pada komunitas AUV. Dermaga pada ROV dikenal dengan istilah yang lain, yaitu TMS (*Tether Management System*). Selain untuk mengangkut ROV, TMS berfungsi untuk menggulung *tether*.

TMS sendiri dibagi menjadi 2 tipe yaitu *clump* (pemberat) dan *cage* (kurungan). Pada tipe *clump* ROV memiliki sistem pengunci sehingga ROV dapat terkunci dengan TMS. Pada tipe *cage* ROV dapat dimasukkan dalam kurungan. Foto TMS tipe *cage* dan *clump* dapat dilihat pada Gambar 2.2.

Kedua tipe TMS ini sengaja dibuat dengan bobot yang berat dan dapat diturunkan dengan kabel yang sangat keras yang disebut dengan *umbilical cable*. ROV yang menggunakan TMS menyelam kebawah air dengan menurunkan TMS sebagai beban, kemudian ROV dapat berenang bebas dimulai dari kedalaman TMS. TMS yang berat dibuat untuk mengurangi seretan ombak.



Gambar 2.19 TMS dengan tipe *cage* (kiri) dan *clump* (kanan) [6]

Pada penelitian ini dibuat model dermaga yang memiliki pengunci. Dermaga yang digunakan pada penelitian ini juga tidak dapat dinaikkan atau diturunkan mengingat ROV yang digunakan pada penelitian ini adalah ROV kelas OCROV yang tidak beroperasi pada laut dalam.

2.2.4 Penggerak ROV

Penggerak yang digunakan ROV dibagi menjadi tiga yaitu, penggerak tenaga listrik, penggerak hidrolik, dan kombinasi dari keduanya. Pemilihan penggerak disesuaikan dengan arus dimana ROV beroperasi. Penggerak dengan tenaga hidrolik biasanya memiliki pompa listrik didalamnya, sehingga ROV dengan penggerak hidrolik membutuhkan daya yang besar.

Penggerak bertenaga listrik sendiri dibagi menjadi tiga yaitu motor listrik, motor DC, motor DC *brushless*. Motor listrik adalah jenis motor yang menggunakan tenaga dari arus bolak balik. Motor listrik dikenal dengan keandalannya dan kemudahan dalam perawatan atau penggunaan. Namun, motor ini memiliki tingkat kerumitan dalam mengontrol kecepatan. Motor DC adalah motor yang paling banyak ditemukan sebagai penggerak ROV kelas observasi. Kemudahan dalam mengontol kecepatan motor ini dan juga kemudahan untuk memodifikasi menjadikan motor ini sangat popular. Motor DC *brushless* adalah

penggerak yang paling banyak dikembangkan pada ROV kelas observasi. Motor ini banyak dikembangkan karena kemudahan dalam perawatan dan umur yang panjang.

Penelitian ini menggunakan motor yang paling mudah ditemukan dan mudah untuk dimodifikasi yaitu motor DC. Motor yang digunakan pada penelitian ini sudah memiliki kemampuan *submersible* atau dapat beroperasi didalam air sehingga tidak membutuhkan segel tambahan. Motor yang digunakan adalah *Bilge pump* yang merupakan motor kuras badan kapal.

2.2.5 Sensor

Desain ROV yang paling minimal tidaklah membutuhkan sensor. Desain ROV paling sederhana adalah suatu wadah kedap air untuk melindungi kelistrikan didalamnya, suatu penggerak, suatu kamera, dan *tether* sebagai sarana mengirimkan perintah dan menerima gambar video.

Pada ROV kelas berat, ROV telah di desain sehingga tidak mudah terhempas arus tanpa bantuan sensor Sensor pada ROV kelas berat lebih banyak digunakan sebagai alat pengambil data saat menjalankan misi. Pada penelitian ini digunakan sensor sebagai alat bantu navigasi. Terinspirasi oleh *quadrotor*, pada penelitian ini digunakan sensor *gyroscope* dan *accelerometer* untuk menstabilkan orientasi ROV.

Sensor *gyroscope* adalah sensor yang digunakan untuk mengukur kecepatan rotasi. Pada kelas industri digunakan sensor *gyroscope* dengan ketelitian mendetail sehingga dapat mengukur kecepatan rotasi bumi. Sensor *gyroscope* bermacam-macam mulai dari sensor mekanis, *ring laser gyro, fiber optic gyro*, hingga MEMS (*Micro-Electro-Mechanical Systems*) *gyro*. MEMS adalah tipe yang paling banyak digunakan pada robot kecil ataupun ponsel. Sensor tipe ini sangat kecil dan memiliki konsumsi daya yang sangat rendah, namun memiliki kemampuan dibawah sensor tipe lain sehingga tidak dapat digunakan untuk memprediksi arah utara.

Sensor *accelerometer* adalah sensor yang digunakan untuk mengukur percepatan translasi. Sensor *accelerometer* paling sederhana dan digunakan pada kapal trdisional adalah pendulum. Sensor sederhana ini memiliki mekanisme untuk

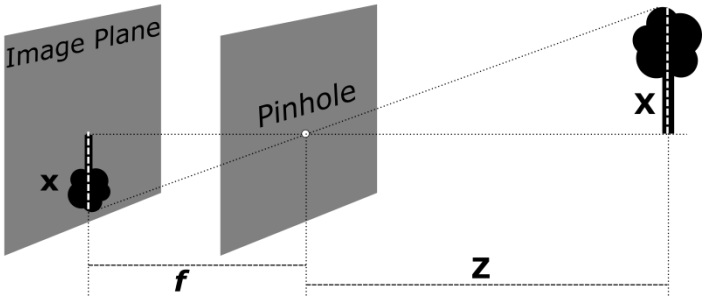
mengukur percepatan translasi sekaligus untuk mengukur sudut kemiringan kapal. Sensor *accelerometer* yang digunakan pada penelitian ini berbasis MEMS. Berbeda dengan sensor *gyroscope* berbasis MEMS, sensor *accelerometer* berbasis MEMS diklaim memiliki kemampuan yang sepadan dengan basis tipe lain.

Sensor lain yang digunakan pada penelitian ini adalah sensor tekanan. Sensor tekanan digunakan untuk mengukur tekanan, sehingga dapat digunakan untuk mengestimasi kedalaman relatif. Kedalaman relatif digunakan untuk mempertahankan kedalaman ROV dalam air.

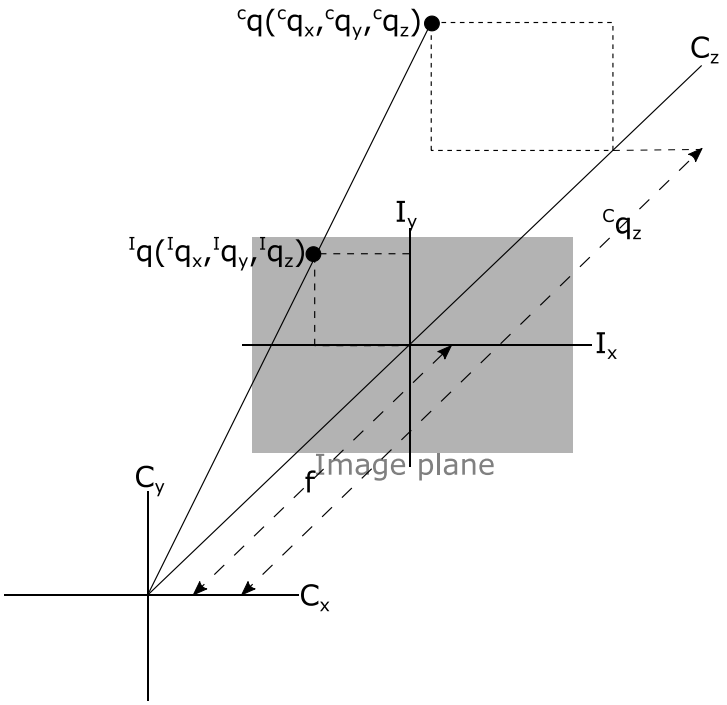
Pada ROV kelas berat, sensor tekanan adalah sensor yang tidak valid digunakan untuk mengukur kedalaman karena perairan laut dalam dapat memiliki densitas yang berbeda-beda. Pada ROV kelas berat digunakan pantulan suara untuk mengukur ketinggian dari dasar laut. Oleh karena itu, ROV kelas berat juga dilengkapi sensor CDT (*conductivity, depth, temperature*). Sensor ini digunakan untuk mengukur kualitas air sehingga diperoleh prediksi ketinggia yang akurat. Pada penelitian ini tidak diperlukan pengukur kedalaman atau ketinggian yang akurat dan mendetil sehingga sensor tekanan dianggap cukup.

2.2.6 Sistem koordinat gambar

Bentuk kamera yang paling sederhana adalah kamera *pinhole*. Pada kamera ini terdapat satu lubang jarum (*pinhole*) dan juga layar penagkap gambar. Setiap cahaya yang melalui lubang jarum akan di proyeksikan ke layar penangkap gambar, oleh karena itu bagian ini juga disebut dengan *image plane*. Besar gambar yang nampak pada layar penangkap gambar bergantung pada lebar jarak antara lubang jarum dengan layar penangkap gambar. Jarak antara layar penangkap gambar dengan lubang jarum disebut dengan *focal length*.



Gambar 2.20 Model kamera lubang jarum (*pinhole*)



Gambar 2.21 Sistem koordinat kamera

Pada gambar diatas terdapat *focal length* (*f*), jarak antara benda dengan lubang jarum (Z), tinggi benda sebenarnya (X), dan tinggi benda pada gambar (x). Dengan jarak dan ukuran yang diketahui pada gambar tersebut terbentuklah dua bangun segitiga [31].

$$\frac{-x}{f} = \frac{X}{Z} \tag{2.1}$$

Pada Persamaan 2.1 tinggi benda pada *image plane* dibuat negatif untuk menandakan bahwa gambar diproyeksikan terbalik. Semakin kecil jarak *focal length* akan menghasilkan ukuran gambar yang semakin kecil pula. Persamaan diatas tidak jauh berbeda dengan persamaan pada kamera sebenarnya. Pada kamera sebenarnya titik fokus gambar ada dibelakang layar.

Gambar 2.2 merupakan ilustrasi posisi titik q terhadap koordinat kamera (C_x, C_y, C_z) dan koordinat gambar (I_x, I_y, I_z). Berbeda dengan gambar yang diciptakan kamera lubang jarum, model kamera tidak menciptakan gambar yang terbalik dan titik fokus model kamera berada dibelakang *image plane*. Dengan menggunakan kesebangunan segitiga maka didapatkan persamaan sebagai berikut.

$$^{I}q_x = f\frac{^{c}q_x}{^{c}q_z} \qquad \text{dan} \qquad ^{I}q_y = f\frac{^{c}q_y}{^{c}q_z} \tag{2.2}$$

Kamera juga dapat digunakan untuk memprediksi jarak benda terhadap kamera ($^{c}q_z$) dengan syarat ukuran benda tersebut telah diketahui. Sehingga dengan ukuran benda yang telah diketahui dapat ditemukan posisi benda terhadap kamera (C_x, C_y, C_z). Jarak benda dengan kamera (d) dirumuskan dengan

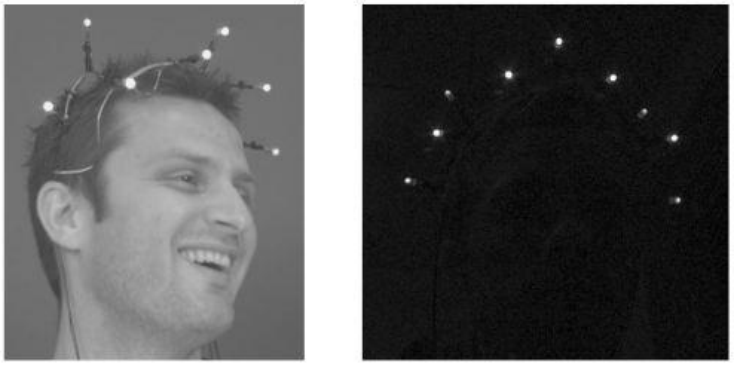
$$d = {}^{c}q_z - f \tag{2.3}$$

2.2.7 Marka kamera

Marka (*marka*) digunakan sebagai penanda dermaga. Kamera adalah sensor visual dengan banyak sekali kegunaan dan flexibilitas. Dari informasi visual, kamera dapat digunakan untuk mengenali bentuk dermaga secara kompleks, namun metode tersebut akan memperberat komputasi.

Mengingat terdapat banyak sekali kombinasi informasi visual, terdapat pula kombinasi marka yang tidak terbatas. Marka yang digunakan dapat berupa warna, bentuk, bahkan kombinasinya seperti wajah, ataupun bahan khusus. Otomasi pengkardusan bisanya menggunakan marka warna untuk menandai ujung dari kardus. Pada industri perfileman digunakan marka yang hanya dapat memantulkan cahaya infra merah (*retro-reflective*) dari arah kamera, dan kamera hanya menangkap pantulan infra merah dari marka [32]. Contoh dari marka yang digunakan dalam industri perfileman dapat dilihat pada Gambar 2.3.

Termasuk marka yang paling sering digunakan adalah marka AR (*Augmented Reality*). Marka ini dapat digunakan untuk mengidentifikasi orientasi marka tersebut. Marka AR biasanya berwarna hitam putih dan dikelilingi dengan garis berwarna hitam. Contoh marka AR dapat dilihat pada Gambar 2.4.



Gambar 2.22 Marka infra merah pada industri perfileman [32].

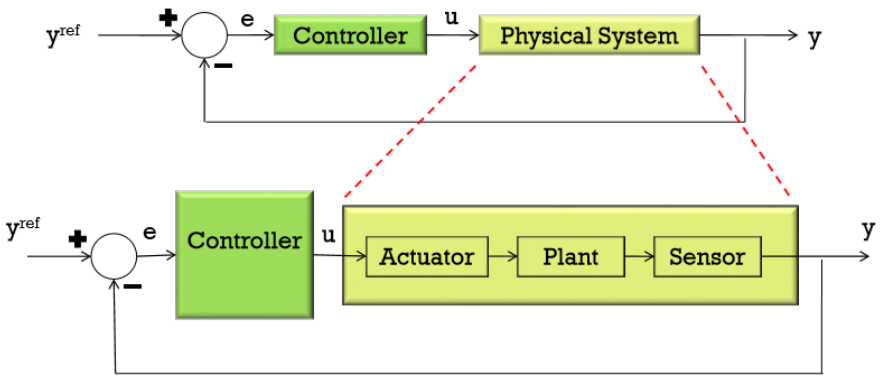
Gambar 2.23 Marka *Augmented Reality* [33].

Pada penelitian ini didesain marka khusus sehingga dapat mengurangi beban komputasi. Marka didesain sehingga dapat dipalikasikan pada komputer skala kecil. Terinspirasi dari marka AR, marka yang digunakan juga di desain supaya dapat memberikan fitur tambahan, seperti orientasi.

2.2.8 Kontrol PID

PID adalah singkatan dari proporsional, integral, dan diferensial yang merupakan tiga komponen parameter kontrol tersebut. Kontrol PID adalah suatu kontrol yang memberikan umpan balik berdasarkan keluaran sistem untuk meningkatkan kstabilan sistem. Tiga komponen parameter PID dapat bekerja sendiri-sendiri (hanya menggunakan komponen parameterproporsional) atau bersamaan (menggunakan tiga komponen atau satu komponen).

Kontrol PID adalah kontrol yang paling banyak digunakan pada industri. Kontrol ini sangat popular karena kemudahan dalam pengoperasian dan kehandalan. Diagram blok kontrol PID dapat dilihat pada Gambar 2.7.



Gambar 2.24 Blok diagram kontrol PID [34].

Blok berwara hijau tua pada Gambar 2.7 diatas adalah kontroler PID. Kontrol PID secara digital dirumuskan sebagai berikut.

$$e = (y^{ref} - y) \tag{2.4}$$

$$u = K_p \left(e + K_i \sum e \, \Delta t + K_d \frac{\Delta e}{\Delta t} \right) \tag{2.5}$$

e pada persamaan di atas adalah *error* atau selisih antara hasil yang diinginkan (y^{ref}) dengan hasil sebenarnya (y). Keluaran kontroler adalah *u*. Konstanta komponen proporsional adalah *Kp*. Konstanta komponen integral adalah *Ki*. Konstanta komponen diferensial adalah *Kd*. Apabila *Ki* dan *Kd* bernilai nol, maka keluaran kontroler berbanding lurus dengan *e* dengan pengali *Kp*. Konstanta *Ki* dikalikan dengan akumulasi error atau integral eror. Konstanta *Kd* dikalikan dengan selisih eror atau diferensial eror.

Keluaran kontrol PID kemudian diaplikasikan pada sistem fisik (*Physical system*). Sistem fisik memberikan keluaran berdasarkan kontrol dengan aktuator (*actuator*). Pada setiap kontrol PID dibutuhkan instrumen untuk mengukur keluaran dari suatu sistem fisik. Oleh karena itu terdapat sensor didalam sistem fisik.

Performa kontrol PID ditentukan berdasarkan empat parameter. *Rise time* adalah waktu yang dibutuhkan untuk mencapai 10% eror menuju 90% eror. *Overshoot* adalah nilai eror paling tinggi setelah melampaui setpoin. *Settle time* adalah waktu yang dibutuhkan dari titik eror hingga ditemukan kestabilan pada setpoin. *Steady state eror* adalah rata-rata eror ketika nilai yang diukur mendekati nilai yang diperintahkan.

2.2.9 Teori pengolahan citra

Pengolahan citra adalah salah satu pokok dari metode yang akan dibuat pada penelitian ini. Terdapat banyak teknik pengolahan citra, namun pada buku ini akan dibahas hanya teknik yang berkaitan dengan penelitian. Teori-teori yang digunaan pada buku ini merupakan teori dasar yang dapat ditemukan dengan mudah pada buku, perkuliahan, ataupun internet.

A. Grayscale

Untuk memahami apa itu *grayscale*, terlebih dahulu harus dipahami apa itu gambar. Gambar adalah fungsi dua dimensi F(x,y) dimana x dan y merupakan koordinat bidang. F adalah amplitudo pada posisi x dan y yang biasa disebut dengan intensitas. Pada gambar digital setiap posisi x dan y memiliki nilai amplitude yang pasti. Suatu gambar digital mempunyai ukuran panjang (*width*) dan lebar (*height*) yang pasti sehingga nilai x dan y memiliki batas dan berhingga. Elemen gambar terkecil adalah F yang biasa disebut *picture element* atau *pixel*. *Pixel* juga dapat diartikan sebagai satuan jumlah dari *picture element*.

Pada umumnya intensitas *pixel* direpresentasikan dalam angka bulat (*integer*) 8 bit yang memiliki 256 kemungkinan nilai. Setiap kemungkinan dari nilai tersebut mewakili warna hitam (0) sampai dengan putih (255). Degradasi warna selain dua nilai tersebut secara matematis disebut dengan abu-abu atau *grayscale*. Gambar berwarna pada umumnya direpresentasikan dengan menggabungkan 3 buah *grayscale* yang masing-masing dari *grayscale* disebut dengan *channel* atau kanal. Tiga kanal tersebut adalah kanal merah, hijau, dan biru atau biasa disebut RGB (*Red Green Blue*). Dengan demikian satu *pixel* berwarna merupakan gabungan informasi dari tiap kanal dimana masing-masing kanal merupakan informasi 8 bit. Sehingga gambar berwarna merupakan suatu *pixel* dengan informasi 24 bit.

Pada umumnya kamera menangkap gambar dalam 3 kanal, yaitu RGB. Untuk mempermudah perhitungan dalam proses pengolahan citra maka informasi 24 harus direduksi menjadi 8 bit atau satu kanal. Proses ini dapat dilakukan menggunakan suatu pustaka pemrograman yang disebut dengan OpenCV. Terdapat beberapa cara paling populer untuk memperoleh hasil ini. Cara pertama adalah dengan rata-rata masing masing kanal.

$$Y(x,y) = \frac{R(x,y)+G(x,y)+B(x,y)}{3} \quad (2.6)$$

Dengan rata-rata maka aka ditemukan hasil gambar satu kanal (Y(x,y)). Cara berikutnya yang popular adalah dengan penjumlahan pembagian

$$Y(x,y) = 0.2126R(x,y) + 0.7152G(x,y) + 0.0722B(x,y) \quad (2.7)$$

Cara kedua menggunakan penjumlahan dengan pemberian porsi tertentu pada masing masing kanal. Pada rumus tersebut telah ditentukan angka porsi masing masing anal. Pada rumus tersebut terlihat bahwa kanal hijau memiliki porsi paling tinggi. Angka tersebut berdasarkan penelitian sensitifitas mata terhadap warna merah, hijau, dan biru. Apabila diperlukan suatu robot yang lebih peka terhadap warna merah, persamaan grayscale dapat dirumuskan dengan menambah porsi pada warna merah. Pada penelitian ini digunakan metode grayscale menggunakan rata-rata yang merupakan pengaturan awal dari OpenCV.

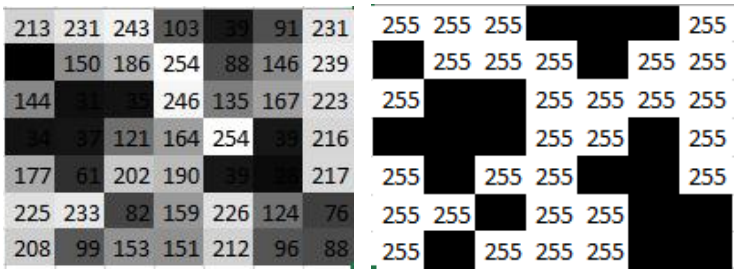
B. Binary image

Binary image atau gambar biner adalah gambar yang setiap *pixel* nya hanya memiliki informasi satu bit, yaitu informasi hitam (0) dan nol (putih). Gambar biner biasanya dihasilkan dengan melakukan *thresholding*, yaitu proses penerapan ambang batas pada gambar abu-abu. Cara ini dilakukan untuk membedakan gambar latar (*background*) dengan gambar muka (*foreground*). Gambar latar merupakan gambar yang tidak digunakan karena mengandung informasi yang tidak signifikan terhadap pengolahan citra berikutnya. Gambar muka adalah gambar yang informasinya diinginkan untuk diambil. *Thresholding* dirumuskan dalam persamaan berikut.

$$Y(x,y) = \begin{cases} 0 & ; \ G(x,y) < Th \\ 255; & lainnya \end{cases} \quad (2.8)$$

Pada persamaan tersebut nilai suatu *pixel* pada gambar Y adalah nol apabila nilai *pixel* pada gambar G lebih kecil dari nilai *threshold* (Th). Nilai 255 adalah nilai putih pada gambar 8 bit yang dapat digantikan dengan 1 pada gambar 1 bit. Nilai ambang batas (*threshold*) dapat pula dibalik sehingga dihasilkan gambar yang berkebalikan antara gambar latar dan gambar muka.

Contoh dari pengubahan gambar abu-abu menjadi gambar biner dapat dilihat pada Gambar 2.25. Gambar tersebut berukuran 7x7 *pixel* dengan nilai setiap *pixel* tercantum. Pengubahan gambar abu-abu (kiri) menjadi gambar biner (kanan) dilakukan dengan memberikan nilai threshold 128. Intensitas *pixel* yang lebih kecil dari 128 diubah menjadi putih (255) dan sebaliknya



Gambar 2.25 Gambar abu-abu (kiri). Gambar biner (kanan)

C. Connected component labeling

Connected component labeling adalah suatu algoritma untuk menentukan jumlah gumpalan (*blob*) gambar muka pada gambar. Algoritma ini biasanya dilakukan setelah gambar biner dibuat. Terdapat banyak cara untuk memperoleh hasil ini namun cara yang paling umum adalah dengan melakukan *scanning* atau pemeriksaan *pixel* tiap barisnya. Algoritma ini sering disebut dengan *row-by-row* atau *two pass*. Berikut adalah *pseudocode* dari algoritma tersebut.

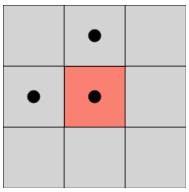
```
L = 0
Lakukan scanning perbaris
        If (intensitas pixel saat ini = 1 dan tidak ada neighbor)
                L = L + 1
                Label pixel tersebut dengan L
        Else if (intensitas pixel saat ini = 1 dan ada neighbor)
                Berikan label sama dengan label terkecil neighbor
                Ingat konflik
Lakukan scanning perbaris
        If (intensitas pixel saat ini = 1)
                Ganti label sesuai dengan konflik label yang diingat
```

Pada *pseudocode* diatas terdapat istilah ‘*neighbor’* yaitu *pixel* tetangga. Neighbor dapat didefinisikan sebagai *pixel* diatasnya dan *pixel* dikirinya, ataupun yang lain. Ilustrasi *neighbor* dapat dilihat pada Gambar 2.26. Pada gambar tesebut dapat dilihat kotak merah dengan titik yang berarti ‘*pixel* saat ini’. *Pixel* saat ini adalah *pixel* yang ingin diperiksa konektifitasnya. *Pixel* diatas dan dikiri (ditandai dengan titik) merupakan neighbor.

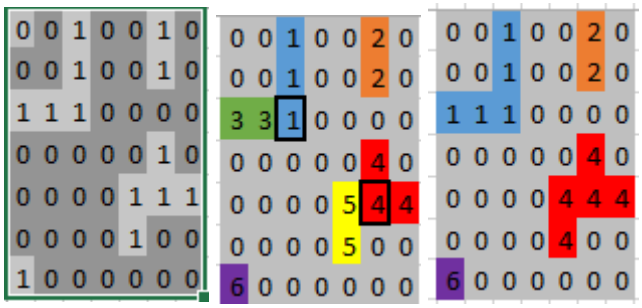
Istilah lain *pseudocode* diatas adalah prosedur ‘mengingat konflik’. Konflik terjadi ketika *pixel* yang diperiksa konektifitasnya memiliki dua atau lebih *neighbor* dengan label yang berbeda. Prosedur menganggap kedua nilai label

tersebut adalah sama merupakan prosedur mengingat konflik. Pada prosedur ‘ganti label sesuai dengan konflik label yang diingat’ dilakukan penggantian seluruh label dengan satu label ekuivalen.

Contoh dari gambar biner yang akan diberi label dapat dilihat pada Gambar 2.27 (kiri). Gambar tersebut adalah gambar dengan ukuran 7x7 *pixel*. Proses pemberian label pada tahap scan pertama dapat dilihat pada Gambar 2.27 (tengah). Pada gambar tersebut telah diberikan kotak pada *pixel* yang terjadi konflik label. Terdapat dua konflik label pada gambar tersebut yaitu label 3 yang semestinya memiliki label yang sama dengan label 1 dan label 5 yang semestinya memiliki label yang sama dengan label 4. Oleh karena itu pada scan kedua seluruh label yang memiliki konflik diganti. Hasil dari scan tahap dua dapat dilihat pada Gambar 2.27 (kanan). Pada gambar tersebut terdapat 4 *blob* karena terdapat 4 label berbeda.



Gambar 2.26 Neighbor

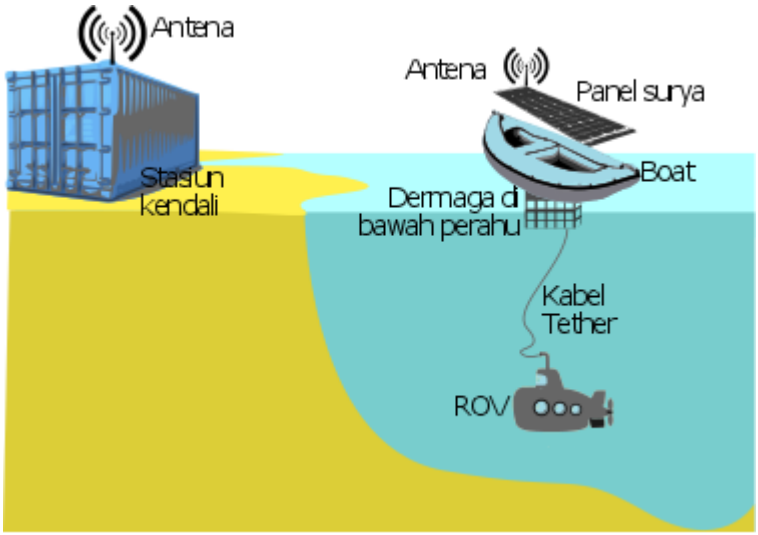


Gambar 2.27 Gambar biner (kiri). Label dari gambar tahap *scan* pertama (tengah). Label dari gambar tahap *scan* kedua (kanan)

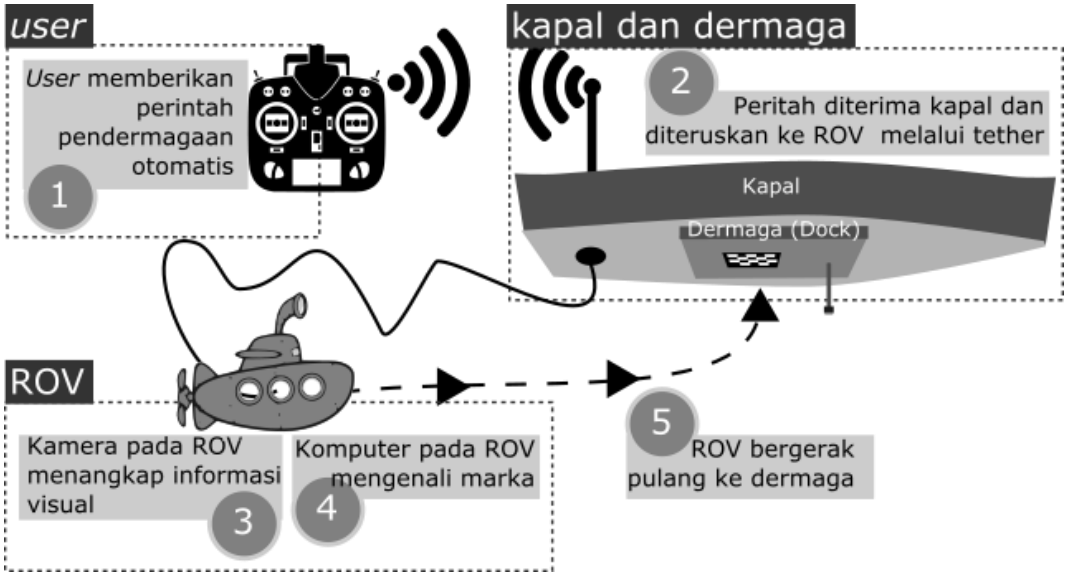
BAB 3
METODOLOGI PENELITIAN

Penelitian ini bertujuan untuk membuat pendermagaan otomatis pada ROV. Penelitian ini adalah bagian dari serangkaian proyek yang bertujuan untuk melestarikan karang. Deskripsi dari keseluruhan penelitian dapat dilihat pada Gambar 3.1.

Dermaga pada penelitian ini dipasang pada perahu yang memiliki panel surya. Perahu tersebut dapat dikontrol secara manual dari stasiun kendali.Perintah yang diberikan pada perahu dikirim secara nirkabel dari stasiun kendali. ROV dihubungkan dengan perahu melalui kabel *tether*. Stasiun kendali juga dapat mengontrol ROV dengan mengirim sinyal perintah ke perahu kemudian perahu meneruskan sinyal perintah pada ROV melalui *tether*. Dengan begitu ROV dapat dikontrol dari jarak yang jauh. ROV juga dapat beroperasi berhari-hari dengan memanfaatkan daya yang dihasilkan panel surya. Pendermagaan ROV pun dapat dilakukan dari jarak jauh. Pendermagaan adalah pekerjaan yang tidak mudah, mengingat pengemudi hanya dilengkapi dengan informasi visual. Untuk melengkapi keseluruhan sistem tersebut, dibutuhkan suatu metode pendermagaan yang lebih baik daripada pendermagaan sepenuhnya.



Gambar 3.1 Gambar keseluruhan sistem pelestarian karang



Gambar 3.2 Gambar skema sistem pendermagaan otomatis yang dirancang

Sistem pendermagaan melibatkan tiga sisi. Sisi pertama adalah *usir* (pengguna). Pada sisi user terdapat remot kontrol dan monitor. Sisi kedua adalah kapal dan dermaga yang merupakan satu kesatuan. Sisi ketiga adalah ROV. Ilustrasi skema sistem pendermagaan dideskripsikan pada Gambar 3.2.

Pada sisi user terdapat remot kontrol yang digunakan untuk memasukkan perintah pada ROV. Terdapat pula layar monitor sebagai alat bantu navigasi. Perintah yang tersedia adalah perintah navigasi, yaitu maju-mundur, bergerak ke atas-bawah, berputar ke kanan-kiri, dan bergeser ke kanan kiri. Perintah yang tidak kalah penting adalah memerintahkan ROV untuk melakukan pendermagaan otomatis. Perintah ini dikirim secara nirkabel ke kapal. Kapal meneruskan perintah kepada ROV melalui kabel. ROV bertugas menerjemahkan perintah dari *user*. Perintah pendermagaan secara otomatis diterjemahkan sebagai menangkap informasi visual kemudian mengolah informasi tersebut untuk mengenali marka. Setelah marka dikenali ROV menalukan manuver untuk kembali ke dermaga.

Garis besar penelitian ini adalah bagaimana cara pendermagaan ROV secara otomatis menggunakan kamera sebagai sensor. Untuk mencapai tujuan pendermagaan secara otomatis, penelitian ini akan menggunakan tiga proses utama pada ROV. Proses pertama adalah pengenalan dermaga yang berisi berbagai teknik pengolahan citra sehingga ROV dapat mengenali dermaga. Proses kedua yaitu pengestimasian orientasi dan posisi ROV yang berisi tentang teknik untuk

mengestimasi orientasi dan posisi yang paling optimal bagi ROV untuk pendermagaan. Proses ketiga adalah penjajak dermaga. Setelah marka pada dermaga telah berhasil terdeteksi oleh ROV, maka ROV akan mendekati dermaga secara otomatis. Ketiga proses tersebut memanfaatkan informasi gambar yang telah ditangkap oleh kamera dan ketiga proses tersebut dilakukan oleh ROV. Keseluruhan proses dieksekusi setelah terdapat perintah untuk pendermagaan secara otomatis. Tahap pendermagaan akan dijelaskan lebih lanjut pada bagian berikutnya.

Berdasarkan studi literatur dapat disimpulkan bahwa sistem pendermagaan otomatis membutuhkan desain ROV dengan kontrol kestabilan. Hal ini ditujukan supaya gambar yang diambil oleh kamera menjadi lebih stabil. Berdasarkan penelitian sebelumnya, penelitian ini akan mengadopsi sistem dermaga dengan kunci (*latch*). Jenis dermaga tersebut dipilih karena paling sesuai untuk pendermagaan dari bawah keatas. Pada penelitian ini juga akan digunakan satu kamera dengan sebuah marka. Cara ini dipilih untuk mengurangi beban komputasi. Beban komputasi harus direduksi supaya sistem pendermagaan dapat diterapkan pada komputer pada komputer kecil seperti Raspberry Pi.

3.1 Perancangan ROV

Dermaga dan marka pada penelitian ini diletakkan dibawah kapal dengan marka menghadap kebawah. Untuk keperluan itu ROV dirancang dengan sistem kontrol *pitch* (sudut angguk) dan *roll* (sudut guling). Kamera pada ROV dihadapkan ke atas karena marka pada dermaga menghadap ke bawah. Untuk mempertahankan posisi ROV terhadap marka, ROV harus bererak maju dan mundur atau bergeser ke kanan dan kiri. Dengan demikian ROV perlu dirancang untuk memenuhi gerak tersebut. Selain untuk pendermagaan, perancangan ROV juga harus mempertimbangkan pergerakan saat dinavigasikan secara manual.

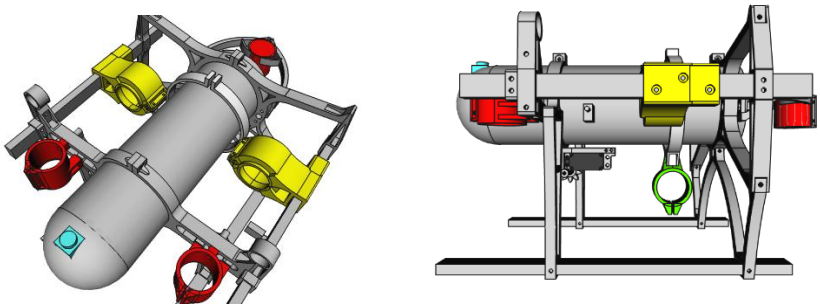
Berdasarkan pertimbangan tersebut di atas, ROV di desain dengan menggunakan enam penggerak. Pada Gambar 3.3, penggerak digambarkan dengan warna merah, kuning, dan hijau. Tiga buah motor berwarna merah merupakan motor untuk bergerak ke atas dan kebawah. Dua buah motor di depan, dan datu buah motor di belakang. Dua motor kuning diletakkan ditengah untuk bergerak

maju dan mundur. Dua motor ini juga digunakan untuk berbelok atau berputar. Yang terakhir adalah sebuah motor berwarna hijau. Motor hijau digunakan untuk bergerak bergeser ke kanan dan ke kiri. Pada gambar diatas juga terdapat ilustrasi kamera yang digambarkan dengan warna biru. Seluruh penggerak yang pada ROV menggunakan motor Bilge Pump.

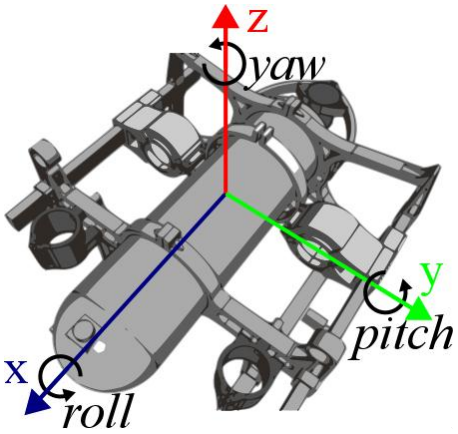
Ketiga motor merah terebut dirancang untuk mengontrol kestabilan sudut angguk (*pitch*) dan sudut guling (*roll*). Ilustrasi konvensi sudut yang digunakan pada ROV terdapat pada Gambar 3.4. Dua buah motor motor merah didepan ROV digunakan untuk menyeimbangkan sudut guling. Sudut angguk di kontrol dengan memanfaatkan kecepatan dua motor merah di depan dan satu motor merah di belakang. Gerakan ROV ke atas dan ke bawah memanfaatkan ketiga motor secara bersamaan. Pada saat ketiga motor merah melaju dengan kecepatan penuh, ROV akan tampak mendongak atau mengangguk karena gaya dorong di depan ROV lebih besar dari pada di belakang. Namun hal tersebut tidak akan menjadi masalah pada pendermagaan otomatis, karena pendermagaan otomatis tidak akan menggunakan kecepatan penuh.

Selain untuk mempertahankan sudut angguk dan guling, ketiga motor merah digunakan untuk mempertahankan kedalaman ROV. Kedalam ROV perlu dipertahankan untuk keperluan navigasi ROV secara manual. Kedalaman juga perlu diperlukan ROV untuk mendekati marka saat pendermagaan otomatis.

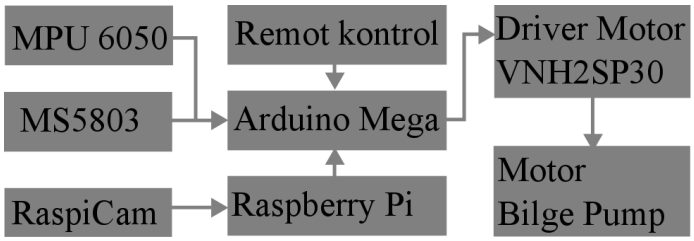
Kontrol sudut angguk dan guling dikontrol menggunakan kontrol PID dan tidak dibahas karena bukan fokus penelitian. Kontrol kedalaman ROV juga menggunakan kontrol PID. Sudut angguk dan sudut guling dikontrol menggunakan umpan balik dari sensor MPU6050, sedangkan kontrol kedalaman ROV menggunakan sensor MS5803.



Gambar 3.3 ROV tampak atas (kiri) dan ROV tampak samping (kanan)



Gambar 3.4 Konvensi sumbu dan sudut pada ROV.



Gambar 3.5 Diagram alur data pada komponen elektronik

Sudut geleng (*yaw*) pada ROV tidak memiliki sistem kontrol. Hal ini dikarenakan sulitnya mengukr sudut geleng dengan sensor inersia (IMU) seperti MPU6050. Sensor yang paling tepat dalam mengukur sudut geleng adalah kompas, namun kompas tidak dapat digunakan pada ROV ini karena pengaruh medan magnet dari motor. Hal tersebut mengurangi performa, namun tidak menjadi masalah dalam pendermagaan otomatis.

Seluruh proses kontrol sudut angguk, sudut geleng, dan kedalaman ROV dilakukan pada Arduino Mega. Arduino Mega membaca sensor MPU6050 dan MS5803 kemudian mengolahnya. *Motor diver* kemudian memberikan keluaran dengan mengaktuasi motor Bilge Pump. Arduino Mega juga mengolah data perintah dari *user*. Data dari *user* dapat berupa perintah navigasi manual atau perintah pendermagaan otomatis. *User* memerintahkan ROV menggunaan remot kontrol. Diagram alur data pada komponen elektronik ROV terdapat pada Gambar 3.5.

Kamera pada ROV dirancang menghadap keatas. Kamera yang digunakan pada ROV adalah RaspiCam. Melalui RaspiCam data visual dari lingkungan sekitar

ditangkap. Data visual dari kamera kemudian diolah menggunakan Raspberry Pi. Raspberry Pi mengolah data visual sehingga diperoleh data yang berguna untuk pendermagaan otomatis. Data tersebut dikirimkan ke Arduino Mega. Arduino Mega kemudian menggunakan data tersebut untuk menggerakkan motor Bilge Pump. Diagram alur data pada Raspberry Pi dan Arduino Mega dapat dilihat pada Gambar 3.5.

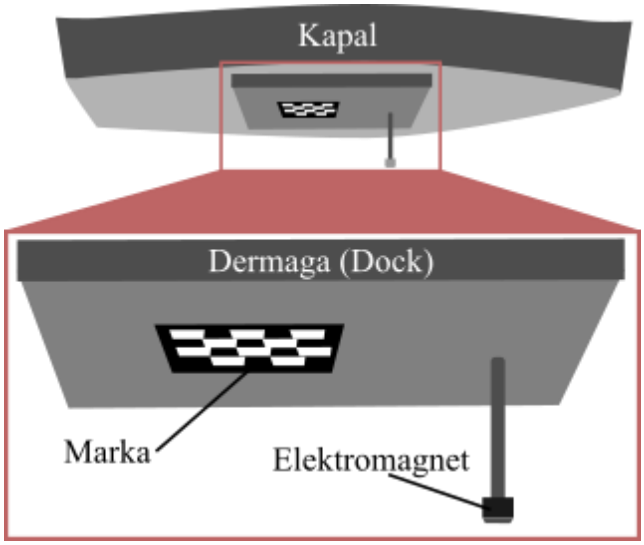
3.2 Perancangan dermaga

Dermaga didesain menghadap kebawah. Penelitian ini menggunakan dermaga dengan penguci. Pengunci ROV yang digunakan adalah elektromagnet dengan kemampuan mengangkat beban sampai dengan 10 kg. Elektromagnet dipilih karena mudah dikontrol. Ilustrasi dermaga terdapat pada Gambar 3.6.

3.3 Perancangan marka

Perancangan dermaga termasuk menjadi fokus penelian ini. Desain marka yang baik sangat membantu kesuksesan pendermagaan otomatis. Oleh karena itu desain dermaga tidak dapat dipisahkan dari metode pendermagaan otomatis.

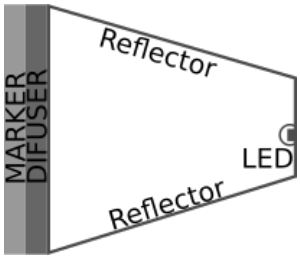
Marka yang digunakan pada penelitian ini terinspirasi dari LED sebagai marka dan marka AR. LED sebagai marka mempunyai satu keunggulan, yaitu sumber cahaya independen. Marka dengan LED tetap dapat terlihat meskipun dalam kondisi gelap. Kelemahan marka LED adalah pancaran cahayanya yang tidak terfokus. Adakalanya marka ini terlalu terang sehingga lingkungan sekitar LED nampak berpendar pada gambar yang ditangkap kamera. Terkadang dua LED yang berdekatan nampak menyatu akibat dari fenomena ini. Dengan menggunakan LED sebagai marka, setiap objek selain LED dianggap sebagai data yang tidak digunakan atau latar belakang (*background*). Marka LED dan latar belakang dibedakan hanya dengan intensitasnya warna, sehingga apabila ada suatu objek dengan intensitas yang sama, objek tersebut dianggap sebagai marka.



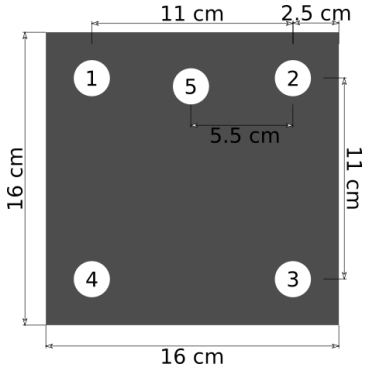
Gambar 3.6 Lingkungan sekitar berpendar karena LED [17].

Gambar 3.7 Lingkungan sekitar berpendar karena LED [29].

Marka AR didesain dengan perubahan warna yang drastis dibandingkan lingkungan sekitar. Hal ini dilakukan dengan memberikan warna hitam disekeliling marka dan warna putih disekitar marka. Marka AR memiliki pola yang unik dari lingkungan sekitar sehingga marka ini mudah ditemukan dan dibedakan dengan latar belakang. Marka AR juga didesain sehingga dapat diketahui bahwa marka tersebut terbalik. Pada penggunaan banyak marka AR biasanya digunakan berbagai pola yang berbeda. Untuk mencapai kombinasi pola yang unik, suatu data biner yang diwakili dengan warna putih dan hitam dapat disematkan dalam marka AR. Dengan demikian setiap marka dapat dibedakan meskipun sangat banyak. Metode tersebut digunakan dalam teknologi *barcode*. Kelemahan teknologi marka AR adalah pencahayaan. Kelebihanya adalah mudah dibedakan dengan lingkungan karena bentuknya.



Gambar 3.8 Desain marka tampak samping



Gambar 3.9 Desain marka tampak depan

Marka AR dapat mengetahui bahwa marka terbalik dengan menggunakan penanda di salah satu ujung marka. Untuk mengatasi masalah pencahayaan pada marka digunakan LED yang disematkan di belakang marka. Cahaya yang dipancarkan LED terlebih dahulu dibentuk menjadi bentuk tertentu sehingga mudah dibedakan dengan latar belakang berdasarkan bentuknya. Cahaya yang dipancarkan juga diredam sehingga tidak menyebabkan lingkungan sekitar berpendar.

Desain marka tampak dari samping dapat dilihat pada Gambar 3.8. Pada gambar tersebut terdapat sebuah LED di sebelah kiri yang memancar menuju *diffuser*. Sebelum cahaya di pancarkan ke *diffuser,* cahaya terlebih dahulu di fokuskan dengan reflektor supaya tidak ada cahaya terbuang. Reflektor terbuat cari cermin. Difuser berfungsi untuk meratakan cahaya yang diterima keseluruh permukaan difuser. Difuser terbuat dari akrilik putih. Bagian tampak depan marka dapat dilihat pada Gambar 3.9.

Pada penelitian ini digunakan marka dengan ukran 16x16 cm^2. Diameter lingkaran adalah 2 cm. Jarak antar lingkaran pojok adalah 11 cm. Ukuran yang paling berpengaruh pada program pengenal marka adalah jarak antar lingkaran. Jarak antar lingkaran mempengaruhi jarak terdekat marka dapat terdeteksi. Semakin lebar jarak antar lingkaran dipojokan, semakin dekat jarak minimal marka

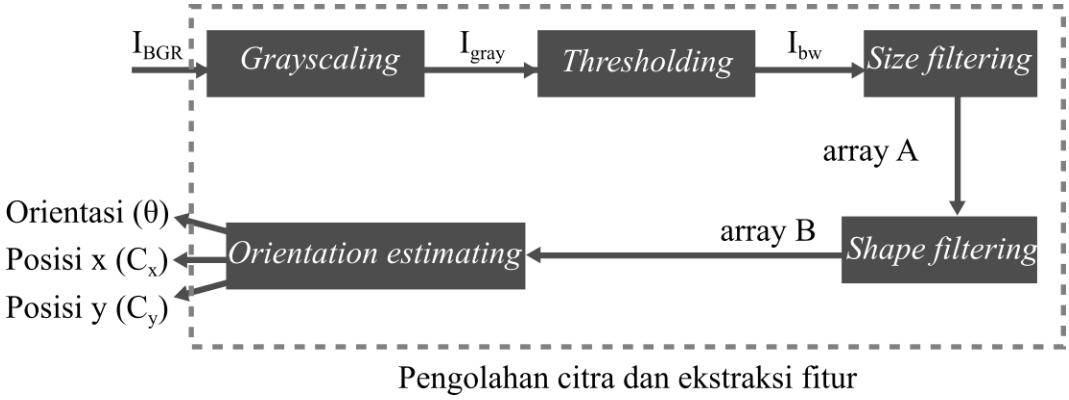
dapat terdeteksi. Ukuran lingkaran menentukan jarak maksimal marka dapat terdeteksi. Untuk memperbesar lingkaran jarak antar lingkaran juga perlu diperlebar. Luasan kotak terluar marka dapat membantu mengisolir lingkaran marka terhadap objek di belakang marka. Kotak terluar marka juga di desan tidak reflektif, sehingga tidak memantulkan cahaya balik. Cahaya terpantul pada marka dapat merusak citra marka.

Marka didesain dari suatu bidang kotak berwarna hitam. Pada marka terdapat lima lubang lingkaran. Cahaya dari *diffuser* nampak dari kelima lingkaran tersebut. Dengan demikian, terdapat lima buah cahaya lingkaran yang tidak berpendar terhadap lingkungan sekitar. Kelima lingkaran memiliki fungsi tersendiri yang akan dijelaskan pada bagian selanjutnya. Gambar marka ketika mati dan dinyalakan dapat dilihat pada Gambar 3.10.

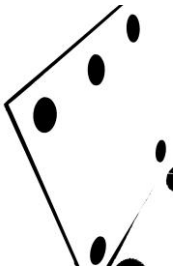
3.4 Pengenalan marka

Pengenalan marka sepenuhnya dilakukan oleh kamera sebagai penerima data visual dan Raspberry Pi sebagai pengolah data visual. Pengenalan marka dibagi menjadi 4 tahap utama. Tahap-tahap berikut khusus disusun untuk pengenalan marka yang telah dibuat. Algoritma pengenalan marka tetap dapat dijalankan pada marka yang sama dengan ukuran yang berbeda.

Gambar 3.10 Marka kertika mati (kiri) dan ketika menyala (kanan)



Gambar 3.11 Tahapan pengenalan marka.



Gambar 3.12 Marka kertika mati (kiri) dan ketika menyala (kanan)

A. Grayscaling

Keseluruhan tahap pengenalan marka dapat dilihat pada Gambar 3.11. Gambar yang ditangkap kamera (I_{BGR}) adalah gambar 3 kanal, yaitu kanal biru (I_B), kanal hijau (I_G), dan kanal merah (I_R). Setiap kanal memiliki ukuran 480x640 *pixel*. Setiap *pixel* memiliki intensitas yang bernilai antara 0 sampai dengan 255 dengan 0 mewakili intensitas paling gelap dan 255 mewakili intensitas paling terang. Setiap kanal Tahapan pertama adalah *Grayscaling* yaitu proses mengubah gambar 3 kanal menjadi 1 kanal dengan mengambil rata-rata intensitas setiap kanal. Gambar yang dihasilkan pada tahap ini adalah gambar abu-abu atau *grayscale* (I_{gray}). Hal ini dilakukan untuk mempermudah perhitungan. Marka yang dibuat juga didesain dalam warna hitam dan putih sehingga informasi warna yang lain tidak terlalu penting.

B. Thresholding

Tahap selanjutnya yaitu *thresholding*. Pada tahap ini dilakukan komparasi intensitas setiap *pixel* gambar terhadap nilai (ambang batas) tertentu. Apabila suatu

pixel pada gambar tersebut memiliki intensitas lebih kecil dari ambang batas yang ditentukan, intensitas *pixel* tersebut di-nol-kan dan sebaliknya. Gambar 3.12 adalah contoh Gambar 3.10 yang telah melalui tahap *thresholding*. Hasil dari tahap ini adalah gambar biner atau gambar yang hanya memiliki dua nilai, yaitu hitam dan putih (I_{bw}). Pada penelitian ini digunakan satu nilai threshold yaitu 220. Nilai ambang batas 220 Perumusan dari *thresholding* dapat dilihat pada persamaan 3.2.

$$I_{bw}(x,y) = \begin{cases} 255 & ; \ I_{gray}(x,y) < 220 \\ 0; & I_{gray}(x,y) \geq 220 \end{cases} \quad (3.2)$$

Setelah tahap *thresholding* dapat dilakukan seleksi bantuk cahaya marka. Cahaya marka yang bagus menghasilkan gumpalan (*blob*) lingkaran hitam yang solid. Cahaya yang terlalu terang menghasilkan bentuk yang tidak beraturan setelah tahap *thresholding*. Desain marka yang baik adalah marka yang menghasilkan lingkaran sempurna setelah tahap *thresholding*.

C. Size filtering

Tahap selanjutnya adalah *size filtering*. Pada tahap ini setiap *pixel* berwana hitam diberikan suatu label dengan algoritma *connected component labeling*. Setiap gumpalan berwarna hitam memiliki label unik. Jumlah *pixel* dengan label yang sama didefinisikan sebagai luas atau ukuran (*size*) dari gumpalan. Algoritma dari size filter dapat dilihat pada *pseudocode* dibawah ini.

```
L = 0 //Hitungan label
IL[480,640] //Buatlah gambar baru dengan ukuran sama seperti gambar Ibw
Ak[n] //Buatlah array untuk menyimpan konflik dengan ukuran yang dapat diubah
Berikan nilai 0 pada setiap pixel IL
For i setiap kolom pada gambar Ibw
    For j setiap baris gambar Ibw
        In = Ibw(i,j) //Ambil nilai intensitas pada baris dan kolom saat ini dalam gambar Ibw
        If (In = 0 dan tidak ada neighbor) then
            L = L + 1
            Ubah ukuran Ak sesuai L
            IL(i,j) = L // Berikan nilai L pada pixel saat ini dalam gambar IL
            Ak(L) = L
        Else if (In = 0 dan ada neighbor)
```

Berikan label yang sama dengan label terkecil neighbor pada *pixel* saat ini dalam gambar I_L
Ak(L) = label terkecil yang ditemui
A[labelKe, dataKe] //Buatlah array dengan ukuran yang dapat diubah untuk menyimpan data posisi setiap *pixel* dengan label yang sama
For i setiap kolom pada gambar I_L
For j setiap baris pada gambar I_L
In = I_L(i,j) //Ambil nilai intensitas pada baris dan kolom saat ini dalam gambar I_L
If (In tidak 0) then
labelKe = Ak(In)
A(In,kolom_terakhir)=(i,j) //Simpan posisi saat ini (i,j) kedalam array A baris ke In kolom terakhir
Tambah jumlah kolom pada baris ke In
For i setiap baris dalam A
countColumn = Jumlah data pada array A baris i
If (countColumn = 0)
Hapus baris tersebut
Banyaknya baris pada A, yaitu banyaknya *blob* yang terdeteksi
Jumlah data pada suatu baris array, A yaitu ukuran (*size*) *blob*
For i setiap baris dalam A
countColumn = Jumlah data pada array A baris i
If (countColumn < nilai *size filter*)
Hapus baris tersebut

Size filtering dilakukan dengan mengisolasi gumpalan yang lebih kecil atau lebih besar dari yang ditentukan pada program. Setelah tahap ini dilakukan, gumpalan kecil yang pada umumnya merupakan *noise* tereliminasi. Gumpalan besar yang pada umumnya merupakan sumber cahaya selain dari marka juga akan tereliminasi. Nilai dari filter ukuran bergantung pada kamera yang digunakan, oleh karena itu nilai filter ukuran ditentukan berdasarkan experimen. Keluaran dari algoritma ini adalah *array* A yang berisi informasi label dan posisi setiap *pixel* pada label yang sama. Dengan kata lain *array* A menyimpan data *blob-blob* yang telah melalui tahap *size filtering* dan posisi setiap *pixel* yang terkandung didalamnya

D. Shape filtering

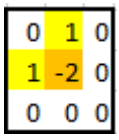
Setelah hanya tersisa gumpalan dengan ukuran yang diinginkan, gambar di proses ke tahap selanjutnya yaitu *shape filtering*. Proses ini menolak bentuk (*shape*) yang tidak diinginkan. Bentuk yang diinginkan adalah lingkaran. Lingkaran

ditentukan dengan derajat lingkaran (*circularity*) yang dihitung berdasaran rumus berikut.

$$c = \frac{4\,\pi\,Luas}{Keliling} \tag{3.3}$$

Derajat lingkaran (c) diukur berdasarkan rasio antara luas ingkaran dan keliling. Derajat lingkaran bernilai antara 0 sampai dengan 1. Gumpalan yang berbentuk lingkaran akan memiliki nilai mendekati 1. Gumpalan yang memiliki lingkaran dibawah batas tertentu dianggap bukan lingkaran marka. Keliling dihitung berdasarkan jumlah *pixel* terluar dari suatu *blob* yang terdeteksi. Jumlah *pixel* yang mengelilingi suatu *blob* diambil dari *edge detection* (pendeteksian tepi) kemudian dijumlahkan. *Edge detection* dilakukan dengan cara melakukan konvolusi antara gambar dengan e*dge detection kernel*. Pada penelitian ini digunakan kernel berukuran 3x3 *pixel* yang dapat dilihat pada Gambar 3.13. Algoritma *shape filter* dapat dilihat pada *pseudocode* berikut:

*Blob*s[480,640] //Buatlah gambar baru dengan ukuran sama seperti gambar I_{bw}
Berikan nilai 0 pada setiap *pixel* dalam gambar *Blob*s
rowNow = baris/*blob* pertama dari array A
For i setiap data dalam rowNow then
    *Blob*s[A[rowNow,i]]=1 //Ambil setiap data posisi dalam baris tersebut dan berikan gambar *Blob*s nilai 1 pada setiap posisi sesuai data posisi yang diambil dari tahap sebelumnya
Edges[480,640] //Buat gambar baru dengan ukuran sama seperti gambar sumber
Edges = Konvolusi gambar *Blob*s dengan kernel edge detection
Hilangkan nilai negetif dengan absolut setiap *pixel* pada gambar Edges
B[labelKe, dataKe] //Buatlah array dengan ukuran yang dapat diubah untuk menyimpan data posisi setiap *pixel* dengan label yang sama
For j setiap baris dalam gambar *Blob*s
    Keliling = Edges[i,j] + Keliling //Keliling *blob* rowNow adalah penjumlahan seluruh nilai *pixel* pada gambar Edges
    Hitung circularity
    If (circularity dari array ke-n > c)
        Simpan data satu baris rowNow dari array A kedalam B
Ulangi algoritma untuk baris berikutnya (rowNow+1) pada *array* A



Gambar 3.13 e*dge detection kernel*

Keluaran dari algoritma ini adalah *array* A yang berisi informasi label dan posisi setiap *pixel* pada label yang sama. Dengan kata lain, *array* B menyimpan data *blob blob* yang telah melalui tahap *shape filtering* dan posisi setiap *pixel* yang terkandung didalamnya. *Size filter* dan *shape filter* dilakukan dengan modul yang tersedia didalam OpenCV

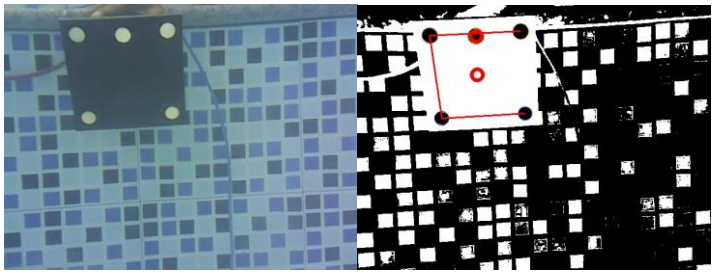
Tahap lain dapat ditambahkan sebagai filter untuk mengurangi probabilitas *noise*. Contoh dari filter lain adalah jumlah lingkaran. Apabila sistem mendeteksi lebih dari 5 lingkaran, maka seluruh data lingkaran ditolak untuk mengurangi eror. Jarak antar lingkaran juga dapat digunakan sebagai filter dengan menentukan kemungkinan jarak terjauh atau terdekat antar lingkaran. Lingkaran diluar batas dianggap bukan cahaya yang dihasilkan marka. Dapat pula digunakan filter jarak antar lingkaran sekaligus jumlah lingkaran.

E. Orientation estimating

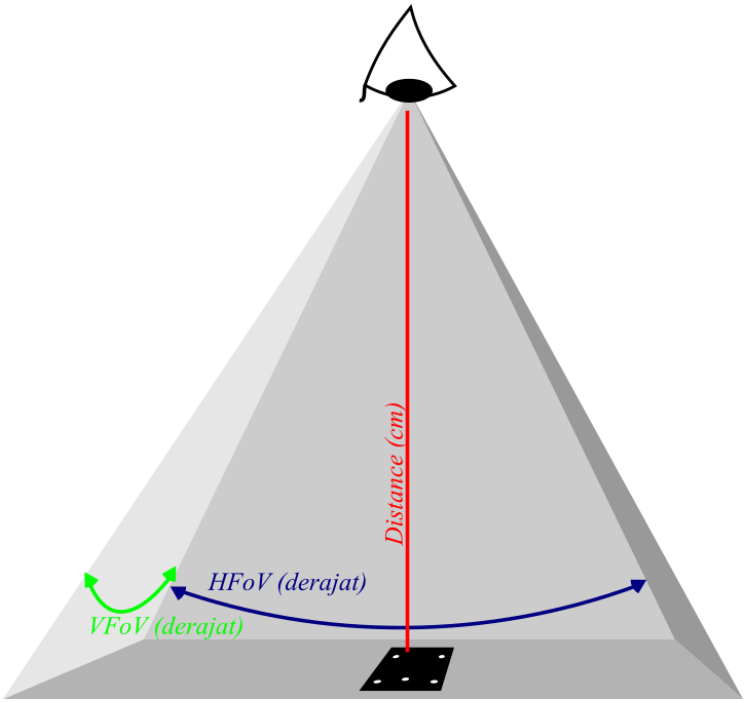
Setelah keseluruhan tahap filter dilakukan, tersisa lima buah lingkaran yang dianggap sebagai cahaya marka. Dengan demikian, marka dikenali berdasarkan bentuk dan ukurannya. Tahap pengestimasian orientasi dijelaskan pada bagian berikutnya. Contoh hasil akhir dari pengenal marka dapat dilihat pada Gambar 3.14 (kanan).

Pada gambar tersebut lima lingkaran marka ditandai dengan lingkaran merah dan dihubungkan dengan garis merah. Lingkaran ditengah menunjukkan titik tengah marka. Dari gambar dapat terlihat jelas latar belakang hitam marka sangat berpengaruh dalam mengisolasi lingkaran marka dengan objek lain disekitarnya. Dapat terlihat pula performa filter ukuran dan bentuk. Lingkaran tetap dapat terdeteksi meskipun banyak *noise* dibelakangnya.

FoV (*Field of View*) diukur dengan satuan derajat dan memiliki ukuran terbatas sesuai kamera yang digunakan. Terdapat dua ukuran sudut dalam FoV, yaitu HFoV (*Horizontal Field of View*) dan VFoV (*Vertical Field of View*). Raspicam yang digunakan pada penelitian ini memiliki FoV 62.2 x 48.8 derajat. Terdapat tiga parameter menggunakan FoV sebagai koordinat posisi marka yaitu jarak dari kamera (*distance*) yang diukur dalam centimeter, HFoV, dan VFoV. Ilustrasi *Field of View* (FoV) dapat dilihat pada gambar 3.15.



Gambar 3.14 Marka sebelum (kiri) dan sesudah (kanan) algoritma pengenalan dilakukan



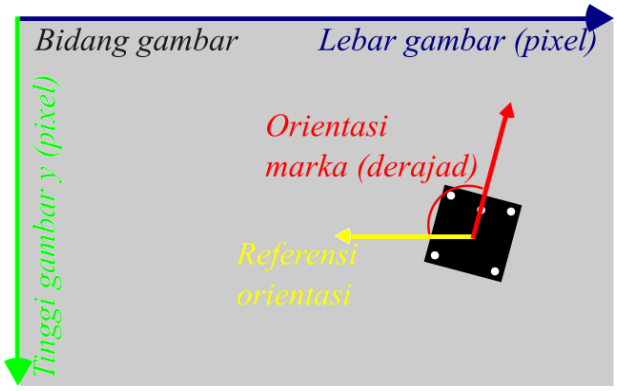
Gambar 3.15. Ilustrasi *Field of View* (bidang fisibilitas kamera)

Pada bidang gambar (*image plane*) posisi marka ditentukan dalam satuan *pixel*. Posisi marka diukur relatif terhadap lebar (*Image width*) dan tinggi gambar (*Image height*). Pada penelitian ini koordinat lebar gambar dianalogikan dengan sumbu *x* dan tinggi gambar dengan sumbu *y*. Koordinat yang berdasarkan lebar dan tinggi gambar diukur dari pojok kiri atas suatu gambar. Pada Raspicam, pertambahan posisi dalam satuan *pixel* pada bidang gambar (*Image plane*) berbanding lurus dengan pertambahan posisi dalam satuan derajat pada FoV. Ilustrasi posisi marka dengan menggunakan koordinat bidang gambar dapat dilihat pada Gambar 3.16. Pada penelitian ini digunakan posisi marka pada bidang gambar sebagai umpan balik posisi.

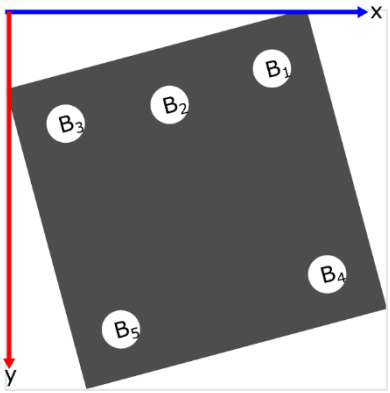
Orientasi marka adalah sudut putar atau kemiringan marka terhadap sumbu tertentu. Satu sumbu orientasi dari marka diilustrasikan pada Gambar 3.16. Pada penelitian ini digunakan hanya satu sumbu orientasi marka karena orientasi terhadap sumbu yang lain tidak memiliki peran yang signifikan terhadap pendermgaan otomatis. Orientasi marka dihitung terhadap referensi yang telah ditentukan dalam program. Referensi orientasi pada penelitian ini diilustrasikan pada Gambar 3.19.

Setelah dilakukan pengenalan marka, langkah selanjutnya yang dapat dilakukan adalah pengestimasian posisi dan orientasi marka. Sebelum melakukan pengestimasian posisi dan orientasi terlebih dahulu dilakukan *blob shorting* atau pengurutan gumpalan. *Blob shorting* dilakukan dengan setikit perhitungan dan logika.

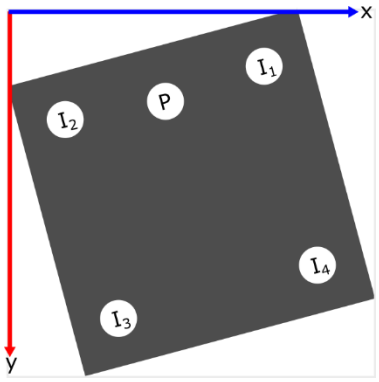
Algoritma *Size* dan *shape filter* yang digunakan pada penelitian ini memberikan hasil array B yang berisi index atau label setiap *blob*. Setiap *blob* memiliki label dengan memperioritaskan kolom kemudian baris gambar. Dengan kata lain, pelabelan *blob* akan dimulai dari pojok kiri atas dan berakhir pada pojok kiri bawah. Label dari gumpalan yang dihasilkan dapat dilihat pada Gambar 3.16



Gambar 3.16. Posisi dan orientasi marka pada bidang gambar.



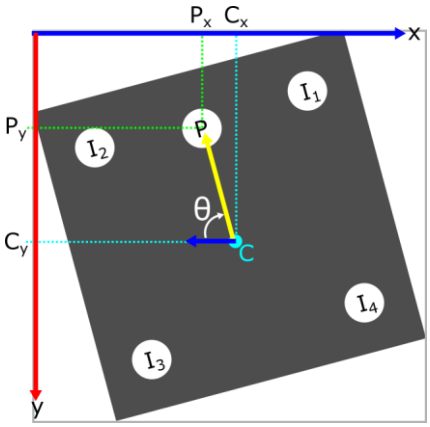
Gambar 3.17. Label *blob* setelah algoritma *shape filtering*.



Gambar 3.18. Label gumpalan setelah diurutkan.

Pada Gambar 3.16 nampak lima lingkaran dengan indeks B_1 (*blob* dengan label 1) sampai dengan B_5 (*blob* dengan label 5). *Blob* B_2 akan digunakan sebagai umpan balik orientasi dan empat gumpalan lainnya sebagai pengestimasi jarak dan posisi. *Blob* tersebut diurutkan menjadi seperti Gambar 3.17.

Blob pertama adalah gumpalan dengan posisi y paling kecil, pada kasus ini adalah B_1. *Blob* kedua adalah *blob* dengan posisi x paling kecil atau gumpalan paling kiri yaitu B_3. *Blob* ketiga adalah *blob* paling bawah yaitu B_5. Gumpalan keempat adalah B_4. *Blob* yang terakhir adalah acuan atau *pivot*. Lingkaran pivot didesain lebih menjorok kedalam sehingga selalu menjadi lingkaran terakhir saat pengurutan (*blob shorting*). Jumlah keseluruhan penomeran atau label *blob* adalah 15. Sehinggan label dari lingkaran *pivot* adalah 15 dikurangi seluruh jumlah label sebelumnya. Hasil dari langkah ini dapat diperiksa dengan menggambarkan garis antar titik berurutan dengan index. Titik yang berurutan akan menghasilkan garis yang berhubungan membentuk kotak seperti pada Gambar 3.14.



Gambar 3.19. Sudut orientasi (θ), posisi pivot (P), dan posisi titik tengah (C).

Empat *blob* yang sudah urut (I_1 dampai dengan I_4) dapat digunakan untuk menentukan titik tengah (C) dari marka. Marka didesain sehingga posisi marka dapat ditentukan dengan sedikit perhitungan. Posisi x dari marka (C_x) berada di antara posisi x dari gumpalan paling kiri dan paling kanan sedangkan posisi y (C_y) dari marka berada di antara posisi y dari gumpalan paling kiri dan paling kanan. Posisi marka dirumuskan sebagai berikut.

$$C_x = (I_{1x} - I_{3x})/2 + I_{3x}$$
$$C_y = (I_{2y} - I_{4y})/2 + I_{2y} \quad (3.4)$$

Hasil perhitungan titik tengah dapat diperiksa dengan menggambarkan titik C pada gambar. Contoh titik tengah yang telah dihitung dan digambarkan dapat dilihat pada Gambar 3.13.

Orientasi marka (θ) dapat ditentukan dari posisi *pivot* (P) dan posisi titik tengah marka (C). Orientasi dapat dihitung dengan mudah karena titik tengah dan titik *pivot* sudah ditemukan. Ilustrasi posisi titik pivot, titik tengah, dan sudut diilustrasikan pada Gambar 3.19. Sudut orientasi adalah sudut tangensial kedua titik. Sudut dihitung dengan persamaan berikut

$$\theta = arctan\left(\frac{P_y - C_y}{P_x - C_x}\right) \quad (3.5)$$

Langkah selanjutnya adalah pengestimasi jarak. Pengestimasian jarak dapat dilakukan dengan persamaan 2.2 dan 2.3. Jarak antara lingkaran I_1 dengan I_2 ($^{I}I_{12}$) dan seterusnya dapat di hitung dengan formula *Euclidean distance*. Titik tersebut dapat dihitung dengan mudah karena telah diurutkan sebelumnya. Jarak tersebut dalam *pixel* dan perlu dilakukan conversi dengan rasio satuan *pixel* ke

satuan centimeter (r). Sehingga jarak I_1 dan I_2 (${}^{C}I_{12}$) dalam satuan cm dapat dihitung sebagai berikut

$$ {}^{C}I_{12} = r\ {}^{I}I_{12} \tag{3.6}$$

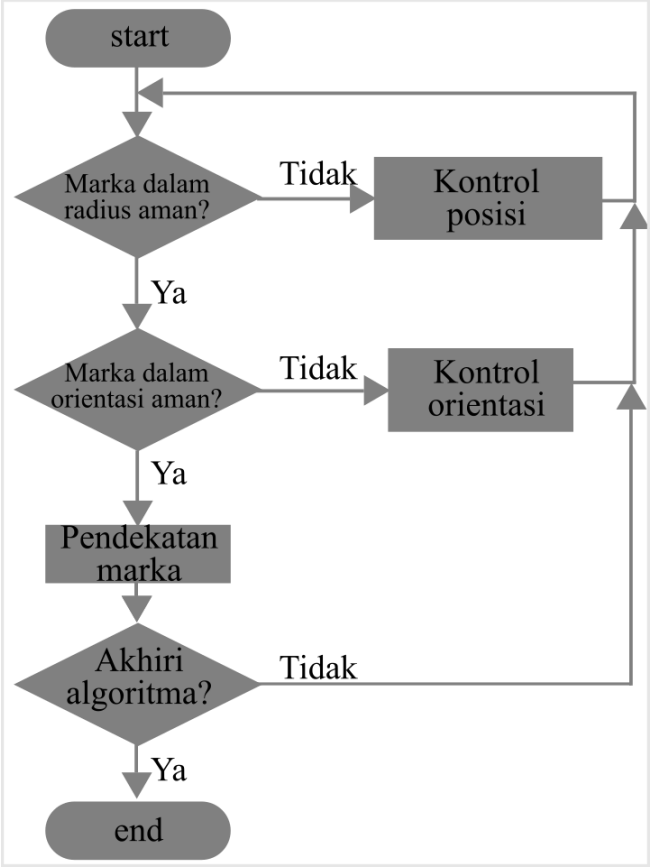
Dengan menggunakan Persamaan 3.6 dan 3.7 maka diperoleh jarak antara kamera dengan marka (d)

$$ d = f - f\frac{{}^{c}I_{12}}{{}^{I}I_{12}} \tag{3.7}$$

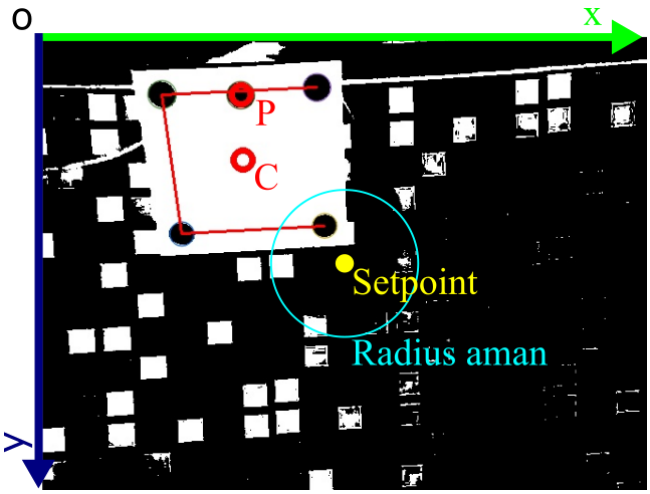
Dengan mengulang proses menggunakan titik I_2 dan I_3 didapatkan pula jarak yang lain. Beberapa kombinasi jarak antar titik dapat digunakan untuk mengestimasi jarak. Pada penelitian ini digunakan 4 jarak antar titik kemudian dihitung rata-rata dari hasil perhitungan. Dengan keseluruhan proses diatas diperoleh posisi marka (C_x dan C_y) dan orientasi marka (θ).

3.5 Penjejakan marka

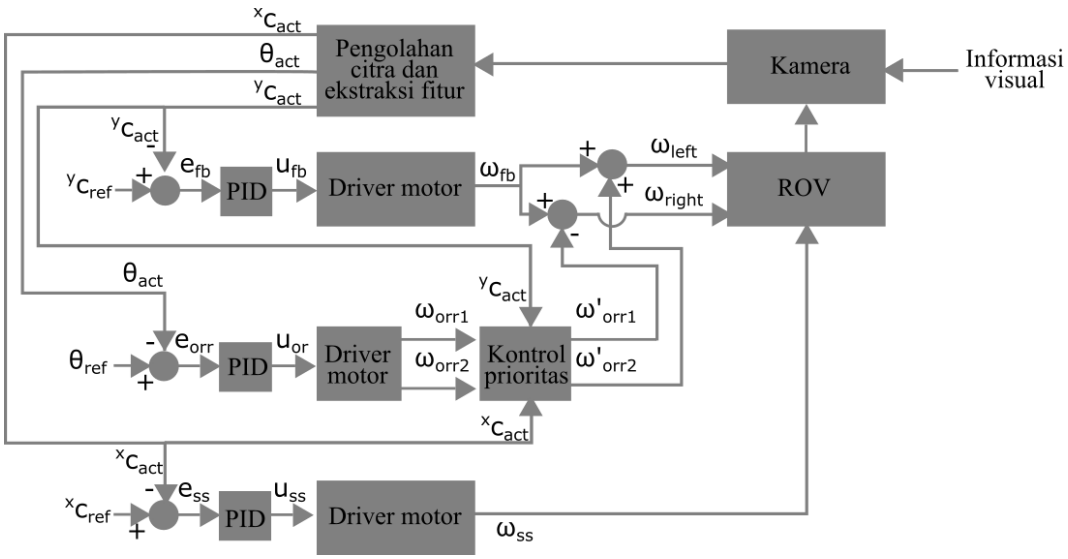
Proses pendermagaan dilakuan dengan memanfaatkan marka. Marka digunakan sebagai patokan posisi dan orientasi dermaga. Oleh karena itu, perlu dilakukan penjejakan (*tracking*) posisi marka. Penjejakan marka adalah proses untuk mempertahankan posisi marka supaya tetap berada dalam FoV dengan menggerakan robot. Lebih dari itu penjejakan juga bertujuan untuk mempertahankan supaya marka selalu di tengah-tengah FoV sehingga dapat dihasilkan perdermagan yang akurat. Untuk melakukan penjejakan ini, kamera digunakan sebagai umpan balik visual dari posisi dan orientasi marka. Teknik kontrol yang digunakan untuk mencapai tujuan ini adalah kontrol PID.



Gambar 3.20. Algoritma penjejakan posisi dan orientasi.



Gambar 3.21. Ilustrasi posisi marka, setpoint kontrol posisi, dan radius aman



Gambar 3.22. Diagram blok kontrol posisi dan orientasi

Penjajakan orientasi dan posisi tidak dilakukan secara bersamaan setiap saat. Penjajakan orientasi dilakukan hanya apabila marka sudah memasuki radius aman. Sebelum memasuki radius aman ROV memprioritaskan kontrol posisi. Orientasi aman adalah toleransi eror orientasi yang diizinkan. Setelah posisi dan orientasi marka berada dalam batas toleransi eror, ROV bergerak mendekati marka yang ditempelkan pada dermaga secara perlahan. Algoritma keseluruhan proses penjejakan diilustrasikan pada Gambar 3.20.

Penjejakan posisi diilustrasikan pada Gambar 3.21. Pada gambar terebut terdapat foto gambar pada aplikasi sebenarnya. Pada gambar tersebut terdapat titik berwarna kuning yaitu *setpoint* (titik acuan) dan lingkaran biru yaitu radius aman. Penjejakan posisi bertujuan untuk menyejajarkan pusat marka (*C*) dengan *setpoint* (terletak di tengah gambar). Proses ini dilakukan dengan mengaktuasi penggerak ROV yang dilakukan pada blok Dirver motor.

Metode kontrol PID digunakan untuk penjejakan posisi dan orientasi. Diagram blok kontrol PID yang digunakan teradpat pada Gamabr 3.23. Pada diagram blok diatas, proses mengaktuasi penggerak ROV dilakukan oleh Arduino. Proses pengolahan citra dan ekstraksi fitur dilakukan oleh Raspberry Pi. Fitur yang diambil dari gambar adalah posisi marka (C_{act}) dan orientasi marka (θ_{act}). Posisi marka tersebut memiliki komponen x ($^{x}C_{act}$) dan y ($^{y}C_{act}$). Kedua fitur tersebut dikirimkan ke Arduino melalui komunikasi serial. Arduino sudah menyimpan nilai

setpoint yang digunakan sebagai acuan kontrol ($^{x}C_{ref}$, $^{y}C_{ref}$ dan θ_{ref}). Selisih dari nilai acuan dan nilai aktual ($^{x}C_{act}$, $^{y}C_{act}$ dan θ_{act}) memberikan nilai error (e_{fb}, e_{ss} dan e_{orr}) yang digunakan untuk menghitung keluaran dari kontrol PID (u_{fb}, u_{ss} dan u_{orr}). Sinyal keluaran dari perhitungan kontrol PID digunakan untuk menggerakkan motor ROV.

Penjejakan marka dilakukan menggunakan 3 motor penggerak. Sebuah motor digunakan untuk bergerak kesamping kanan dan kiri. Motor tersebut diautuasi berdasarkan sinyal u_{ss} yang diperoleh dari kontrol PID posisi x. Sinyal tersebut digunakan untuk menggerakkan motor dengan kecepatan ω_{ss}. Motor mendorong ROV untuk bergerak sehingga ada perubahan posisi $^{x}C_{act}$.

Dua motor digunakan untuk menggerakkan ROV maju mundur dan juga berputar. Kedua motor tersebut digerakkan berdasarkan sinyal ω_{left} dan ω_{right}. Kedua sinyal tersebut merupakan penjumlahan dari kontrol posisi y dan kontrol orientasi. Kontrol posisi y menghasilkan u_{fb} yang digunakan untuk menghasilkan kecepatan motor ω_{fb}. ω_{fb} diaktuasikan pada dua motor dengan kecepatan yang sama sehingga dapat mendorong ROV untuk bergerak maju dan mundur.

Kontrol orientasi menghasilkan sinyal u_{or}. Sinyal tersebut digunakan untuk mengaktuasikan kedua motor dengan kecepatan ω_{orr1} dan ω_{orr2}. ω_{orr1} dan ω_{orr2} memiliki besar yang sama namun arah putar yang berbeda sehingga apabila diaktuasikan dapat mendorong ROV untuk bergerak berputar. Sebelum diaktuasikan ke motor, ω_{orr1} dan ω_{orr2} terlebih dahulu harus melalui tahap kontrol prioritas. Kontrol prioritas mempertimbangkan posisi x ($^{x}C_{act}$) dan posisi y ($^{y}C_{act}$) ROV. Kedua posisi ini digunakan untuk menghitung posisi ROV terhadap radius aman. Apabila ROV tidak berada pada radius aman (jarak 33 *pixel* dari tengah bidang gambar), maka ω'_{orr1} dan ω'_{orr2} dinolkan. Apabila ROV tidak berada pada radius aman, maka $\omega'_{orr1} = \omega_{orr1}$ dan $\omega'_{orr2} = \omega_{orr2}$.

Penjumlahan antara ω'_{orr1}, ω'_{orr2} dengan ω_{fb} menghasilkan ω_{left} dan ω_{right}. Apabila $\omega_{left} = -\omega_{right}$ atau $-\omega_{left} = \omega_{right}$ maka ROV bergerak berputar. Apabila $\omega_{left}=\omega_{right}$ maka ROV bergerak maju atau mundur. Penjumlahan dari keduanya menggerakkan ROV maju dengan berbelok atau mundur dengan berbelok.

Proses kontrol orientasi dan posisi terus diulang hingga posisi marka berjarak 33 *pixel* dari tengah gambar dan orientasi marka berada diantara $\pm 20^{o}$.

Ketika kedua kondisi telah terpenuhi ROV bergerak keatas mendekati marka. Gerakan keatas ditentukan degan kecepatan tertentu. Apabila sesaat didalam proses mendekati marka kedua kodisi tidak lagi terpenuhi, proses penjajakan orientasi dan posisi kembali dilakukan. Pada penelitian ini algoritma dihentikan secara manual setelah ROV berada tepat didermaga.

Halaman ini sengaja dikosongkan

BAB 4
HASIL DAN PEMBAHASAN

Ambisi dari metode yang telah dibuat adalah dapat mengesimasi posisi dan orientasi ROV sehingga dapat digunakan untuk pendermagaan otomatis. Metode ini diharapkan dapat meningkatkan kecepatan deteksi marka dengan probabilitas deteksi yang tinggi. Pada bagian ini diuji berbagai performa metode seperti kecepatan dan akurasi. Ketahanan metode terhadap gangguan seperti kekeruhan dan kemiringan pun diuji pada bagian ini.

4.1 Percobaan penentuan nilai filter ukuran yang optimal

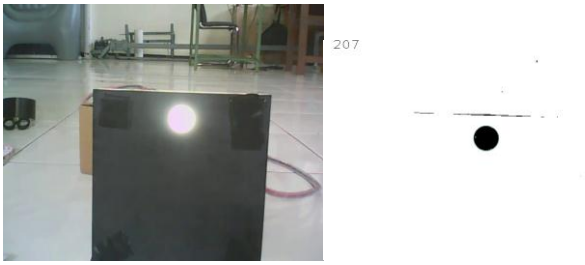
Nilai filter ukuran sangat bergantung pada berbagai faktor seperti pencahayaan, desain marka, intensitas marka, ataupun kamera yang digunakan. Pada eksperimen ini dilakukan percobaan untuk menemukan nilai filter ukuran yang tepat. Karena nilai filter ukuran sangat berpengaruh dengan jarak, pada percobaan ini akan ditemukan ukuran filter yang optimal berdasarkan jarak tertentu.

Percobaan ini dilakukan dengan kamera Logitech C170. Lingkaran pada setiap marka berdiameter 2 cm. Percobaan dilakukan dengan cara mengambil video dari satu lingkaran. Setiap video diambil dari jarak tertentu, kemudian algoritma pendeteksi lingkaran dijalankan. Parameter *circularity* yang digunakan pada percobaan ini dibuat konstan, yaitu 0,8. Karena jarak adalah faktor yang paling mempengaruhi pendeteksian lingkaran, pada percobaan ini akan disimpulkan maksimal pendeteksian disetiap ukuran filter yang di desain. Sebuah cuplikan dari video yang diambil dari percobaan ini dapat dilihat pada Gambar 4.1.

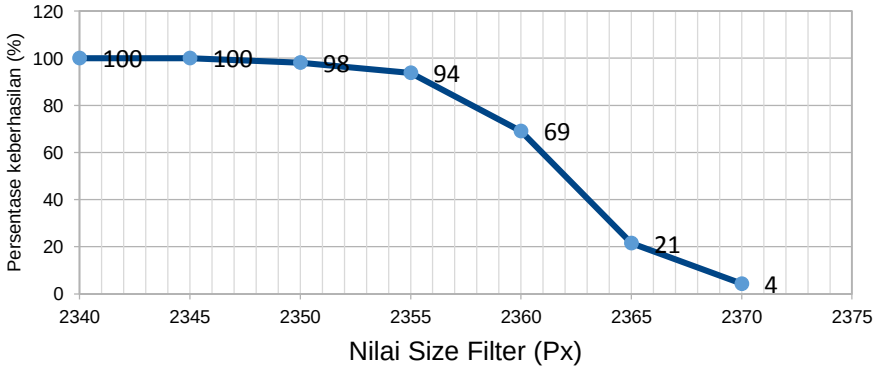
Pada setiap tahap pengambilan data akan dihasilkan data seperti pada Gambar 4.2. Data tersebut diambil pada jarak yang sama, yaitu 30 cm dari kamera. Sumbu y pada gambar tersebut adalah persentasi keberhasilan. Sumbu x pada gambar tersebut merupakan ukuran filter yang digunakan. Ukuran filter minimal dapat digunakan sebagain eliminator objek yang lebih kecil dan ukuran minimal lingkaran supaya dapat terdeteksi. Persentasi pendeteksian dihitung dengan berapa

kali lingkaran terdeteksi dibandingkan seluruh jumlah cuplikan pada video. Pada data tersebut nampak bahwa memperbesar nilai filter ukuran dapat menurunkan persentase pendeteksian. Pada Gambar 4.2 dapat disimpulkan bahwa 2350 *pixel* adalah sudah cukup untuk mendeteksi lingkaran yang di desain pada jarak 30 cm dengan persentasi pendeteksian 98%. Dengan mngulangi proses ini pada setiap jaraknya, maka diperoleh ukuran filter yang optimal pada setiap jaraknya. Pengertian dari potimal pada percobaan ini adalah dengan persentasi pendeteksian diatas 90%.

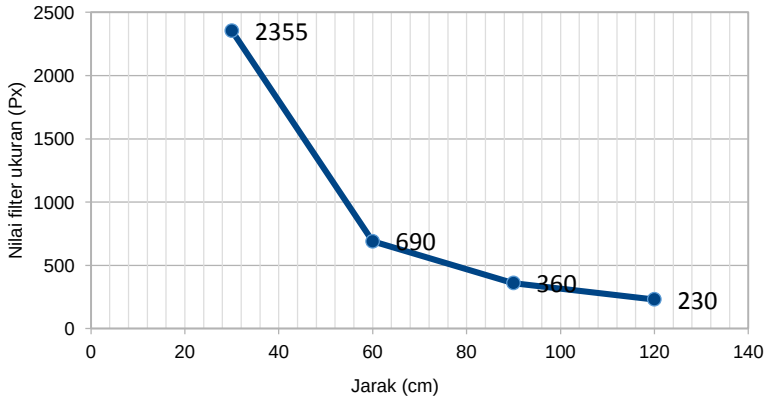
Percobaan nilai ukuran filter terus diulang hingga marka berjarak 120 cm dari kamera. Nilai filter ukuran yang optimal pada setiap jaraknya disajikan pada Gambar 4.3. Sumbu x pada gambar tersebt merepresentasikan jarak dari marka ke kamera dalam satuan cm. Sumbu y merepresentasikan nilai maksimum dari filter ukuran yang optimal untuk digunakan. Nilai yang lebih kecil dari data yang disajikan pada jarak tersebut akan mengurangi persentasi pendeteksian lingkaran sehingga marka tidak dapat terdeteksi. Nilai yang lebih besar dari data yang disajikan pada jarak tersebut akan memperbesar persentasi pendeteksian marka, namun akan memperbesar kemungkinan *noise* untuk masuk.



Gambar 4.1 Cuplikan video pengambian data. Sebelum algoritma diterapkan (kiri). Sesudah algoritma diterapkan (kanan)



Gambar 4.2 Persentase pendeteksian terhadap nilai filter ukuran pada jarak 30 cm



Gambar 4.3 Nilai filter optimal pada jarak tertentu.

Pada Gambar 4.3 hanya disajikan data hingga berjarak 120 cm. Hal ini dikarenakan persentase keberhasilan tidak pernah melebihi 90% diatas jarak tersebut. Oleh karena itu, jarak diatas 120 cm dikatakan tidak optimal dan 230 *pixel* adalah ukuran lingkaran terkecil yang paling optimal. Jarak pendeteksian dapat ditingkatkan dengan memperbesar lingkaran pada desain marka atau memperkecil nilai *circularity* pada program.

4.2 Percobaan penentuan nilai *circularity* yang optimal

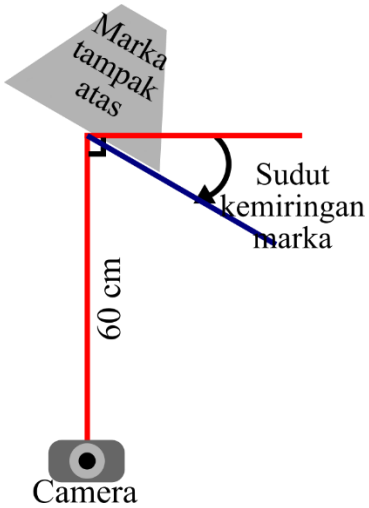
Pendeteksian marka juga bergantung dari sebelah mana marka dilihat. Marka dikenali berdasarkan bentuknya. Bentuk selain lingkaran, seperti oval, diabaikan dan dianggap sebagai *noise*. Akan tetapi lingkaran pada marka akan tampak oval apabila dilihat dari samping. Oleh karena itu, terdapat sudut maksimal marka dapat terdeteksi yang dipengaruhi oleh nilai *circularity*.

Tujuan dari percobaan ini adalah untuk menentukan nilai *circularity* yang optimal. Menentukan nilai *circularity* sangatlah penting untuk menjaga performa dari pendeteksian marka. Nilai circularity sangat dipengaruhi oleh sudut pandang. Karena marka yang didesain adalah marka 2D, maka lingkaran pada marka akan nampak seperti oval apabila dilihat dari samping. Oleh karena itu, pada percobaan kali ini akan disajikan nilai *circularity* yang optimal berdsarkan sudut pandang tertentu.

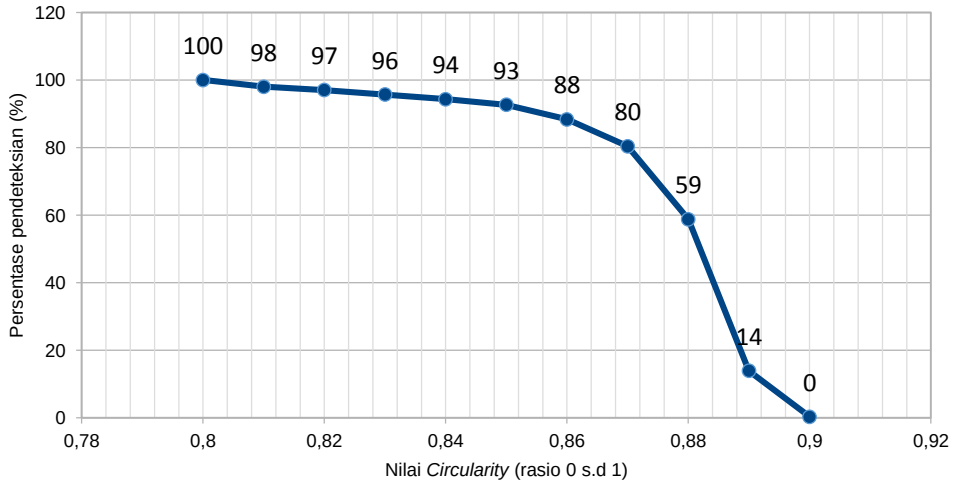
Perangkat keras yang digunakan pada percobaan ini sama dengan percobaan penentuan nilai filter ukuran. Pada percobaan ini kamera diletakkan

dengan jarak tetap yaitu 60 cm dari marka. Nilai filter ukuran juga telah ditetapkan menjadi 100 *pixel*. Setiap data diambil dengan terlebih dahulu memiringkan marka. Ilustrasi skenario pengambilan data dapat dilihat pada Gambar 4.4. Dengan menggunakan scenario ini marka direkam selama beberapa detik dengan sudut yang sama. Filter ukuran dan *circularity* kemudian diaplikasikan pada setiap frame dari video tersebut. Pada setiap cuplikan video terkadang terdeteksi lingkaran marka dan terkadang tidak bergantung pada kemiringan marka. Perbandingan jumlah cuplikan dimana lingkaran marka terdeteksi dengan jumlah keseluruhan cuplikan dalam video disebut sebagai persentase keberhasilan pendeteksian.

Setiap sudut yang sama akan diuji terhadap berbagai nilai *circularity*. Data persentase pendeteksian pada sudut 0^o ditampilkan pada Gambar 4.5. Sumbu x pada gambar tersebut merepresentasikan nilai filter circularity yang digunakan. Sumbu y merepresentasikan persentase keberhasilan berdasarkan nilai *circularity* yang digunakan. Pada data tersebut dapat dibuktikan bahwa memerkecil nilai *circularity* dapat meningkatkan persentase keberhasilan pendeteksian lingkaran. Pada penelitian ini dipilih 90% sebagai persentase pendeteksian yang optimal. Pada sudut ercobaan ini (0^o) dapat disimpulkan bahwa circularity tidak boleh lebih besar dari 0.85 untuk mendapatkan persentase lebih besar 92.6%. Data nilai circularity optimal pada sudut lainnya disajikan pada Gambar 4.6.

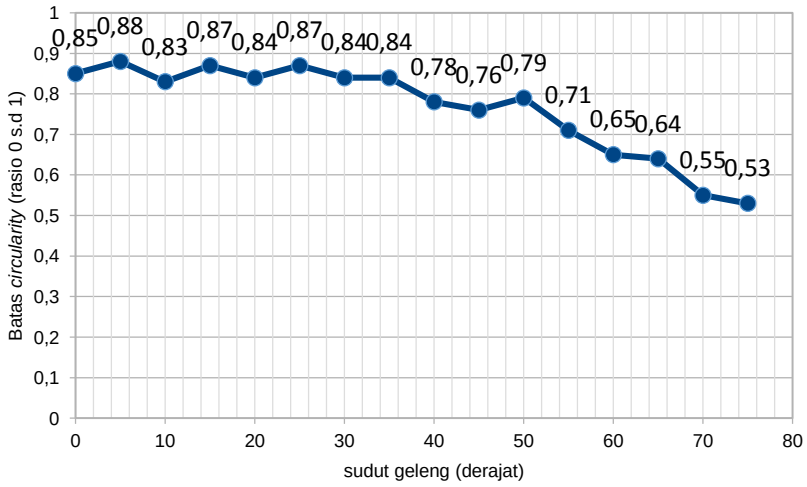


Gambar 4.4 Skenario pengambilan data nilai *circularity* yang optimal terhadap sudut kemiringan marka

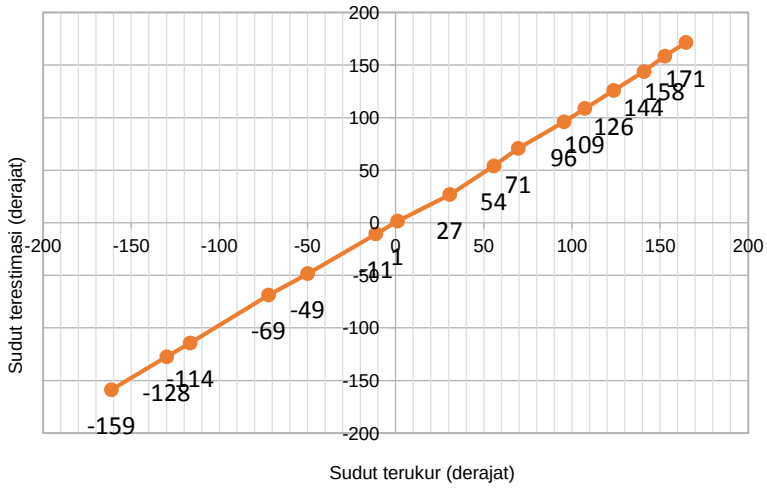


Gambar 4.5 Persentase keberhasilan pada sudut kemiringan marka 0^{o} dengan berbagai batas *circularity*

Untuk meningkatkan presentase pendeteksian, batas *circularity* harus dibuat lebih kecil. Fenomena ini dapat dilihat pada Gambar 4.6. Sumbu x pada gambar tersebut mewakili sudut kemiringan marka. Sumbu y mewakili nilai *circularity* yang optimal berdasarkan sudut kemiringan. Dari data dapat disimpulkan bahwa 0.8 adalah nilai yang cukup baik sebagai batas *circularity* sampai marka dimiringkan 40^{o}. Diatas 40^{o} batas *circularity* harus diperkecil. Pada gambar hanya disajikan data kemiringan sampai dengan 75^{o} karena marka tidak dapat terdeteksi apabila lebih miring. Diatas 75^{o}, memperkecil batas circularity tidak meningkatkan persentase pendeteksian. Hal ini dikarenakan, selain menjadi oval, luas dari lingkaran juga menjadi lebih kecil dibawah 100 *pixel*. Oleh karena itu, supaya marka dapat dikenali pada kemiringan yang lebar, batas *circularity* tidak boleh lebih besar dari 0.52.



Gambar 4.6 Batas *circularity* yang optimal pada setiap sudut kemiringan marka.



Gambar 4.7 Sudut rotasi terukur dengan sudut rotasi terstimasi

4.3 Pengujian performa pengestimasi orientasi marka

Marka telah didesain supaya orientasi dapat diestimasi secara mudah. Pada bagian ini diuji akurasi dari pengestimasi orientasi. Untuk melakukan pengujian ini marka diputar dengan sudut tertentu kemudian sudut putar tersebut diukur dengan alat ukur terkalibrasi. Hasil dari alat ukur kemudian dibandingkan dengan hasil estimasi.

Pada pengujian ini kamera diletakkan 60 cm dari marka. Sesaat setelah sudut orientasi marka ditentukan dan diukur, video dari marka diambil. Algoritma pengestimasi sudut diaplikasikan pada setiap cuplikan video. Setiap cuplikan

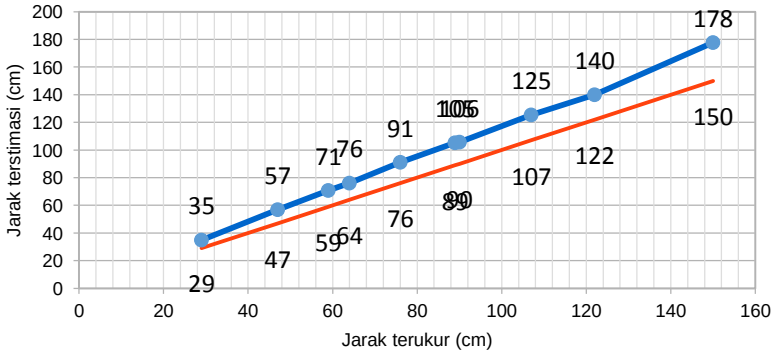
gambar dalam video menghasilkan satu sudut estimasi. Sudut estimasi dari keseluruhan cuplikan dalam video kemudian di rata-rata. Data dari rata-rata estimasi setiap sudut disajikan pada Gambar 4.7.

Pada Gambar 4.7 sumbu x mewakili nilai ukur orientasi marka dan sumbu y mewakili nilai estimasi. Pada data tersebut terdapat garis yang linear. Hal ini menunjukkan hubungan linear antara nilai orientasi terukur dan orientasi terestimasi. Rata-rata error dari estimasi adalah 1.75^{o}. Standar deviasi dari sudut estimasi adalah 0.502 yang berarti pada setiap pengukuran tidak terjadi fluktuasi data yang besar. Berdasarkan data tersebut dapat disimpulkan bahwa pengetimasi sudut cukup digunakan untuk membantu pendermagaan otomatis.

4.4 Pengujian performa pengestimasi jarak

Marka yang telah di desain dapat digunakan untuk mengestimasi jarak dari marka ke kamera. Pengujian ini bertujuan untuk mengetahui performa dari pengestimasi jarak. Untuk melakukan engujian terdapat beberapa parameter yang dibuat tetap. Seperti pengujian sebelumnya nilai *size filter* dipilih yang paling optimal, yaitu 100 *pixel* dan *circularity* 0.8.

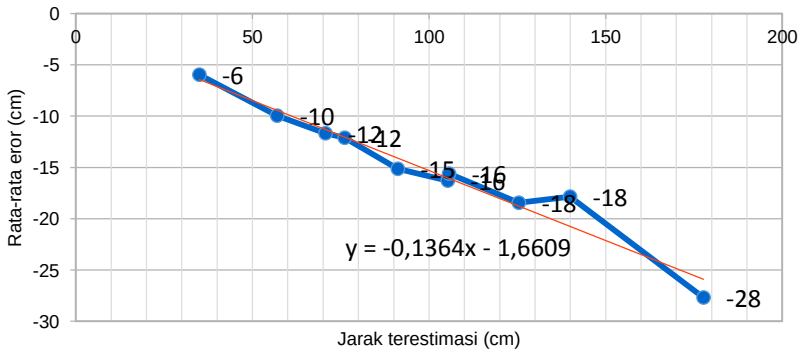
Pada pengujian ini kamera diletakkan dengan jarak yang terukur dengan alat ukur terkalibrasi. Pada jarak tersebut di rekam video dari marka. Algoritma pengestimasi jarak dilakukan pada jarak tersebut. Terdapat banyak nilai estimasi sesuai dengan jumlah cuplikan pada video. Rata-rata pengestimasian pada jarak tersebut kemudian diambil. Data rata-rata pengestimasian pada setiap jaraknya dapat dilihat pada Gambar 4.8



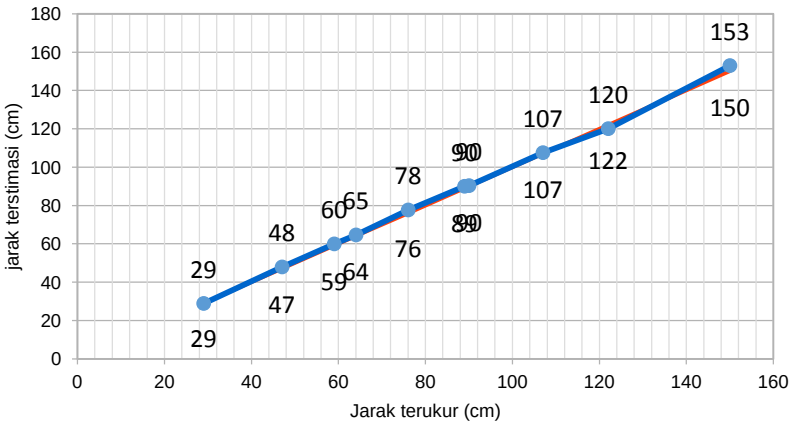
Gambar 4.8 Rata-rata pengestimasian jarak disbanding jarak sebenarnya.

Terdapat dua garis data pada Gambar 4.8. Garis oranye menunjukkan hasil pengukuran sebenarnya dibandingkan dirinya sendiri. Oleh karena itu garis oranye terlihat lurus sebagai referensi. Garis biru menunjukkan hasil estimasi dibandingkan dengan pengukuran sebenarnya. Pada gambar tersebut garis biru terlihat lurus yang berarti estimasi jarak terhadap jarak sebenarnya bersifat linier. Pada gambar juga nampak bahwa garis biru terlihat semakin menjauh dari garis oranye yang berarti terdapat selisih atau eror pengukuran. Selisih pengukuran bertambah seiring dengan bertambahnya jarak.

Pada Gambar 4.9 disajikan data eror (sumbu y) dibandingkan jarak yang diestimasi (sumbu x). Pada gambar tersebut nampak bahwa eror menjadi semakin besar kearah negatif. Dengan bantuan dari data tersebut dapat ditemukan relasi eror terhadap hasil estimasi. Relasi tersebut digunakan untuk membuat formula sebagai kalibrasi dari estimasi yang ditunjukkan oleh garis oranye. Hasil data estimasi jarak setelah dikalibrasi disajikan pada Gambar 4.10



Gambar 4.9 Rata-rata eror dibandingkan jarak estimasi



Gambar 4.10 Estimasi jarak setelah dikalibrasi

Pada Gambar 4.10. sekarang terlihat bahwa data estimasi (biru) berhimpitan dengan data jarak sebenarnya (oranye). Rata-rata eror dari estimasi setelah dikalibrasi adalah 0.62 cm. Standar deviasi dari estimasi adalah 2.38 yang juga berarti nilai fluktuasi pengukuran. Meskipun nilai estimasi berfluktuasi namun pengestimasi jarak marka cukup digunakan untuk membantu pendermagaan otomatis.

4.5 Pengujian performa metode filter ukuran dan bentuk dibandingkan transformasi morfologi

Pada penelitian ini marka dikenali dengan menggunakan filter ukuran dan filter bentuk. Terdapat cara yang lebih simpel daripada cara yang dapat digunakan pada penelitian ini yaitu menggunakan tranformasi morfologi. Transformasi morfologi mengeliminasi *noise* dengan cara mengurangi ukuran gumpalan gumpalan kecil atau menggabungkan gumpalan besar.

Pada percobaan ini marka diletakkan 60 cm dari kamera. Dari jarak tersebut diambil video dari marka. Algoritma pengenalan marka menggunakan *size filter* dan *shape filter* juga diaplikasikan pada video tersebut. Setelah itu algoritma diuji dengan mengganti *shape filter* dan *size filter* dengan algorima transformasi morfologi untuk mengurangi *noise*. Dari jarak tersebut dihitung persentase keberhasilan pendeteksian. Marka kemudian dimiringkan ke samping kemudian persentase keberhasilan dihitung kembali. Data dari pengujian ini ditampilkan pada Tabel 4.1

Tabel 4.1 Respon konstanta kontrol PID terhadap kontrol posisi pada sumbu x

Y angle (derajat)	Presentase keberhasilan (%)	
	Size filter	Morfoloogy transformation
0	100.00	14.62
10	100.00	0.00
20	97.48	21.01
30	100.00	7.27
40	94.64	1.79
50	0.00	9.38
60	0.00	0.00

Pada data tersebut nampak bahwa kedua metode gagal dalam mengenali marka ketika dimiringkan melebihi 50^{o}. Dari data tersebut dapat dilihat bahwa

metode yang dibuat tidak begitu unggul apabila dimiringkan ke samping namun sangat unggul apabila dimiringkan ke atas. Berdasarkan data tersebut dapat disimpulan bahwa metode filter ukuran dan bentuk lebih unggul daripada metode konvensional.

4.6 Percobaan penentuan nilai konstanta kontrol PID

Penjejakan marka pada penelitian ini dilakukan dengan menggunakan metode kontrol PID. Kontrol PID mendapatkan umpan balik berdasarkan fitur pada citra yang ditangkap. Konstanta parameter PID yang digunakan ditentukan berdasarkan eksperimen. Percobaan penentuan konstanta PID perlu dilakukan untuk memperoleh hasil kontrol yang baik.

Terdapat tiga kontrol PID yang digunakan pada penelitian ini (kontrol PID posisi x, posisi y, dan orientasi). Pengujian dilakukan pada akuarium dengan kondisi arus dan pencahayaan terkontrol. Pengujian dilakukan dengan menghilangkan kemampuan ROV dalam mendekati marka, sehingga gerakan ROV dalam mendekati marka tidak mempengaruhi pengukuran. Perbandingan performa berbagai konstanta kontrol untuk masing-masing kontrol ditunjukkan pada bagian ini.

Tabel 4.2 Respon konstanta kontrol PID terhadap kontrol posisi pada sumbu x

Kp	Ki	Kd	Overshoot (%)	Rise Time (detik)	Settling time (detik)	Eror steady state (%)
0,4	0	0	13,9	16,6	-	-
0,6	0	0	17,7	7,1	-	-
0,7	0	0	15,1	5,4	-	-
0,8	0	0	14,1	6,5	-	-
0,9	0	0	12,3	5,9	-	-
1,0	0	0	23,5	5,9	-	-
1,1	0	0	22,8	6,2	-	-
1,2	0	0	27,5	6,2	-	-
0,7	0,007	0	61,8	6,0	14,7	24,3
0,7	0,009	0	63,8	7,4	17,1	27,4
0,7	0,011	0	61,6	6,5	15,3	22,9
0,7	0,013	0	65,0	6,7	16,7	12,6
0,7	0,015	0	65,7	6,6	26,1	11,8
0,7	0,017	0	65,0	7,4	27,5	9,2
0,7	0,019	0	62,6	6,0	26,3	8,1
0,7	0,021	0	62,2	7,3	23,0	-1,3
0,7	0,025	0	64,9	7,3	22,7	-5,2
0,7	0,030	0	60,9	7,4	22,4	2,5
0,7	0,035	0	62,5	6,2	21,8	2,2
0,7	0,035	0,050	65,4	8,2	24,4	-7,2
0,7	0,035	0,100	61,6	8,1	28,7	1,5
0,7	0,035	0,150	60,9	6,7	21,2	5,7
0,7	0,035	0,200	60,0	6,1	17,8	2,3
0,7	0,035	0,250	63,0	6,4	14,3	-8,1
0,7	0,035	0,300	62,9	7,3	15,9	-3,7

Pada Tabel 4.2 ditunjukkan respon kontrol posisi x terhadap berbagai konstanta PID. Konstanta yang lebih dahulu dipilih adalah proporsional (*Kp*) Pada tabel tersebut dapat dilihat bahwa naiknya nilai konstanta proporsional meningkatkan nilai *overshoot*. Konstanta *Kp* tidak memberikan pengaruh yang begitu signifikan terhadap *rise time*. Data *settling time* saat hanya menggunakan konstanta *Kp* tidak disebutkan karena terlalu lama atau *steady state error* terlalu besar dan tak menentu.

Berlandaskan *rise time* tercepat dipilih konstanta *Kp* terbaik, yaitu 0,7. Untuk memperbaiki *steady state error*, ditambahkan konstanta integral (*Ki*) pada kontrol posisi x. Pada Tabel 4.1 dapat dilihat perbandingan performa saat kedua konstanta digunakan. Pada penelitian ini dipilih konstanta *Ki* dengan pengaruh terhadap *steady state error* terbaik. Berdasarkan data, konstanta *Ki* 0.021 memberikan hasil terbaik. Eror akan turun seiring dengan naiknya nilai konstanta hingga konstanta bernilai 0.035. setelah nilai konstanta tersebut eror kembali naik, oleh karena itu dipilih *Ki* 0.035 pada penelitian ini.

Berdasarkan data nilai overshoot menjadi sangatlah besar setelah konstanta *Kp* dan Ki diterapkan. Oleh karena itu diperlukan nilai konstanta diferensial (*Kd*) untuk mengurangi overshoot. Namun penambahan nilai *Kd* pada kontrol tidak memberikan perbaikan yang signifikan terhadap overshoot. Penambahan nilai *Kd* dapat berguna apabila terjadi gangguan dari arus didalam air, sehingga dipilih *Kd* dengan nilai 0.2 pada penelitian ini.

Tabel 4.3 Respon konstanta kontrol PID terhadap kontrol posisi pada sumbu y

Kp	Ki	Kd	Overshoot (%)	Rise time (detik)	Settling time time (detik)	Steady state error (%)
0,4	0	0	0	-	-	151,3
0,5	0	0	8,5	4,9	23,6	65,7
0,6	0	0	15,8	3,8	13,6	11,8
0,7	0	0	16,7	1,9	12,6	21,9
0,8	0	0	19,6	2,9	21,6	-3,2
0,9	0	0	20,1	4,5	21,6	0,9
1	0	0	25,9	2,0	16,8	-2,0
1,2	0	0	28,0	1,8	29,0	4,2
1,4	0	0	30,3	1,8	20,5	10,0
1,2	0	0,2	22,8	1,3	18,7	6,6
1,2	0	0,4	20,1	1,5	19,0	2,4
1,2	0	0,6	18,4	1,3	16,0	15,1
1,2	0	0,8	16,4	1,3	17,4	4,3
1,2	0	1	14,9	1,2	10,9	31,7

Respon kontrol posisi y terhadap berbagai konstanta kontrol PID dapat dilihat pada Tabel 4.3. Pemilihan konstanta dilakukan dengan terlebih dahulu dengan memilih nilai *Kp*. Pada tabel dapat dilihat bahwa nilai *Kp* sangat mempengaruhi *overshoot*. Semakin besar nilai *Kp* maka semakin besar pula overshoot. Saat *Kp* bernilai 0.4 tidak terjadi *overshoot* namun sistem tidak pernah mencapai *setpoint*. Dengan nilai *Kp*=0.5 sistem sempat mencapai *setpoint* dengan adanya *overshoot* namun kemudian menyimpang kembali dengan nilai *steady state* error yang lebih besar. Fenomena ini menunjukkan bahwa overshoot terjadi karena ROV tetap terpropagasi sesaat setelah motor berhenti. *Kp* yang dipilih pada penelitian ini adalah 1,2. Nilai tersebut dipilih karena memiliki nilai rise time terendah, atau paling cepat mencapai *setpoint*, sekaligus memiliki nilai *steady state* error yang kecil.

Nilai konstanta *Kp* yang dipilih tidak memberikan nilai *error steady state* yang besar sehingga konstanta yang selanjutnya harus ditentukan adalah *Kd* untuk mengurangi *overshoot*. Nilai *overshoot* berkurang dengan naiknya nilai *Kd*. Pada penelitian ini dipilih nilai *overshoot* yang kecil dengan nilai *steady state error* yang kecil. Nilai *steady state error* sangat mempengaruhi akurasi dari pendermagaan. Nilai *Kd* 0.4 dan 0.8 memiliki nilai *steady state error* yang kecil, namun nilai *Kd* 0.8 memiliki *overshoot* yang lebih kecil. Karena nilai *overshoot* yang besar dapat menyebabkan marka hilang dari medan pandang kamera.

Konstanta *Ki* dapat mengurangi *steady state error* namun meningkatkan *overshoot*. Konstanta *K*i pada kontrol posisi terhadap sumbu y dapat dibiarkan nol atau diberikan nilai yang sangat kecil. Hal ini dilakukan karena pada posisi y tidak terjadi *error steady state* yang begitu berarti.

Tabel 4.4 Respon konstanta kontrol PID terhadap kontrol orientasi

Kp	Ki	Kd	Overshoot (%)	Rise time (detik)	Settling time time (detik)	Steady state error (%)
1	0	0	13.6	7.1	8.4	3.7
1.2	0	0	0.0	7.3	10.2	4.8
1.4	0	0	20.6	6.4	20.1	19.4
1.6	0	0	38.8	5.9	18.0	7.4
2	0	0	30.5	5.7	20.9	12.1
2.5	0	0	46.3	5.6	19.4	51.2
3	0	0	13.9	4.6	14.9	13.1
3.5	0	0	27.5	3.2	16.1	15.6
4	0	0	43.4	3.0	16.9	29.8
5	0	0	53.4	3.8	18.0	39.5

Respon kontrol orientasi terhadap berbagai konstanta kontrol PID dapat dilihat pada Tabel 4.4. Pemilihan konstanta dilakukan dengan terlebih dahulu dengan memilih nilai Kp. Naiknya nilai *Kp* memberikan kecenderungan naiknya nilai *steady state error* dan menurunnya waktu *rise time*. Pada penelitian ini diprioritaskan *steady state error* yang minimum untuk menjaga akurasi meskipun membutuhkan waktu yang lama. Oleh karena itu, pada penelitian ini dipilih 1 sebagai nilai *Kp*. Nilai *Kp* tersebut memberikan error steady state 3.7% dengan *rise time* 7.1 detik.

4.7 Pengujian performa kontrol posisi dan orientasi

Metode yang digunakan pada penelitian ini diimplementasikan pada Raspberry Pi untuk mengolah gambar dan RaspiCam untuk mengangkap gambar. Dengan menggunakan Raspberry Pi, rata-rata waktu yang dibutuhkan untuk mendeteksi marka pada setiap cuplikan adalah 0.26 detik. Penggerak pada penelitian ini dikontrol menggunakan Arduino Mega. Dengan menggunakan perangkat tersebut, performa kontrol dari metode yang digunakan pada penelitian ini diuji. Performa kontrol diuji untuk mengetahi kualitas dari metode.

Pada pengujian ini marka diletakkan 65 cm diatas ROV. Algoritma pendermagaan otomatis kemudian dijalankan ketika marka telah nampak melalui kamera. Pengujian dilakukan pada kolam renang dengan kondisi arus dan gelombang tidak terkontrol sehingga mungkin terjadi gangguan pada data yang didapatkan. Beberapa cuplikan gambar marka pada saat pengambilan data dapat dilihat pada Gambar 4.11.

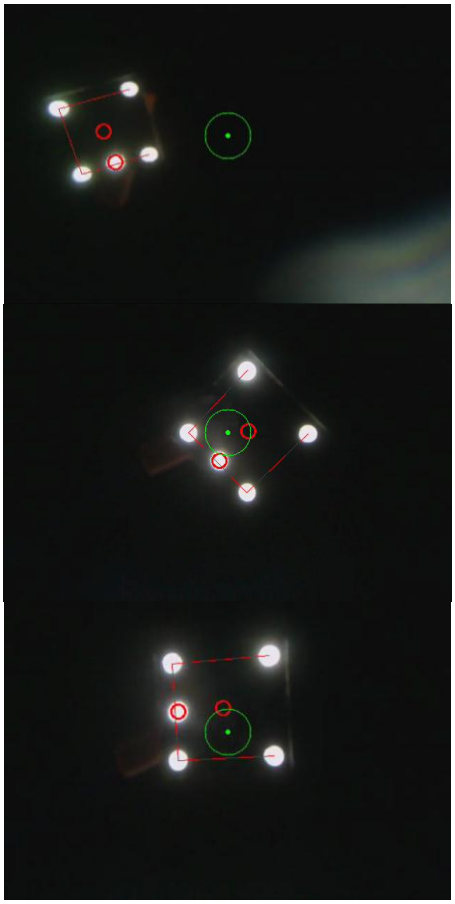
Pada Gambar 4.11 nampak gambar marka saat pendermagaan otomatis. Pada gambar tersebut marka diletakkan 65 cm diatas ROV. Marka nampak diluar radius aman, kemudian berangsur-angsur nampak bergerak ketengah. Setelah marka berada di radius aman ROV bermanuver kembali sehingga marka dan ROV berada pada orientasi yang tepat. Kemudian ROV bergerak mendekati marka. Ketika ROV berjarak 30 cm dari marka, pendermagaan otomatis dihentikan dan hasil data yang terekam dievaluasi.

Data eror posisi marka terhadap sumbu x dan y bidang gambar disajikan dalam Gambar 4.12. Pada gambar tersebut terdapat dua grafik. Grafik pertama adalah data eror posisi terhadap sumbu x dan grafik kedua adalah posisi terhadap sumbu y. Kedua data eror posisi disajikan dalam domain waktu.

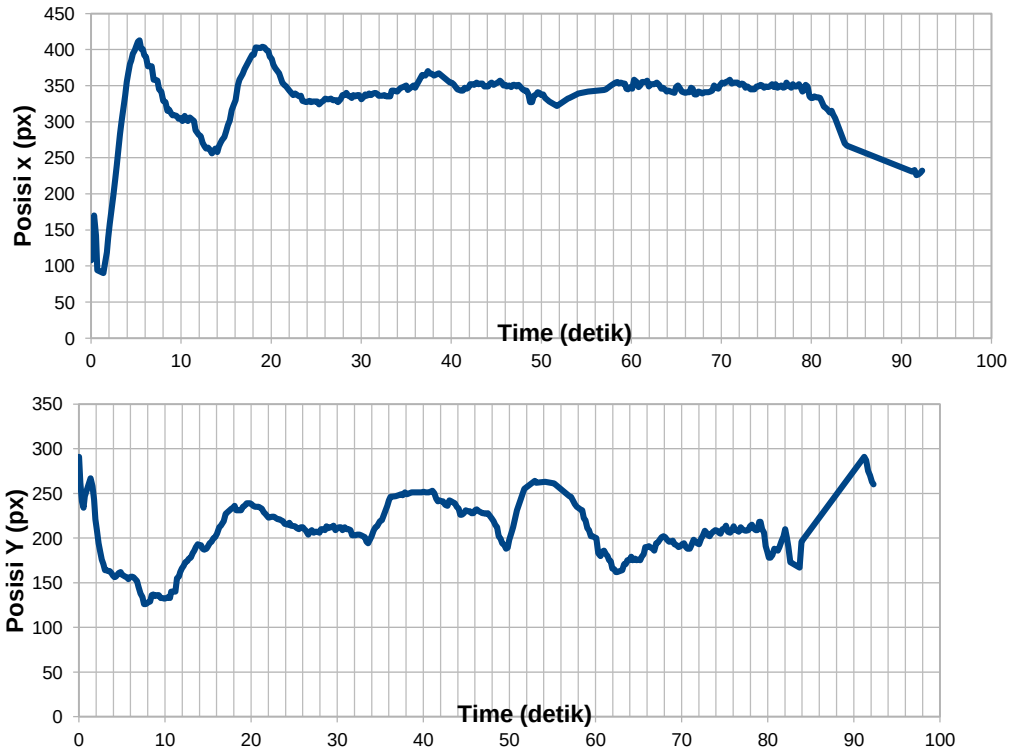
Pada grafik pertama Gambar 4.12 dapat diketahui bahwa *rise time* dari kontrol terhadap sumbu x adalah 2.53 detik dengan overshoot 41%. Dari grafik juga dapat diamati bahwa terdapat *eror steady state* dengan rata-rata 20 *pixel* dengan standar deviasi 21.6 *pixel*. *Eror steady state* menunjukkan bahwa marka selalu berjarak 20 *pixel* terhadap tengah bidang gambar. *Settling time* dari kontrol posisi x adalah 18 detik.

Data dari kontrol posisi terhadap sumbu y terlihat berosilasi. Hal tersebut dikarenakan penggerak untuk mengurangi eror posisi y juga digunakan untuk mengurangi eror orientasi setelah marka berada pada radius aman. *Rise time* dari kontrol posisi terhadap sumbu y adalah 9.07 detik dengan *overshoot* sebasar 57,1%.

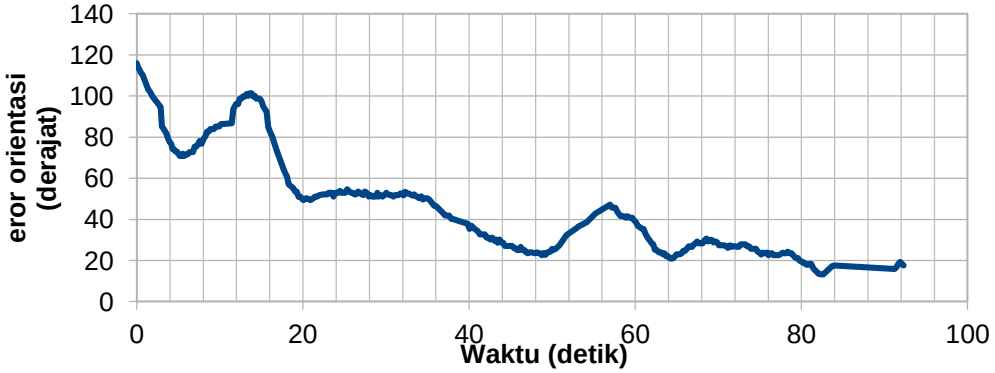
Pengambilan data orientasi dilakukan bersama dengan proses pendermagaan otomatis. Data orientasi ditunjukkan pada Gambar 4.13. Pada gambar tersebut terlihat bahwa eror orientasi perlahan berkurang. Berdasarkan grafik dapat diperkirakan bahwa *rise time* dari kontrol orientasi adalah 73 detik. Hal tersebut dikarenakan sistem di desain dengan lebih memperioritaskan kontrol posisi.



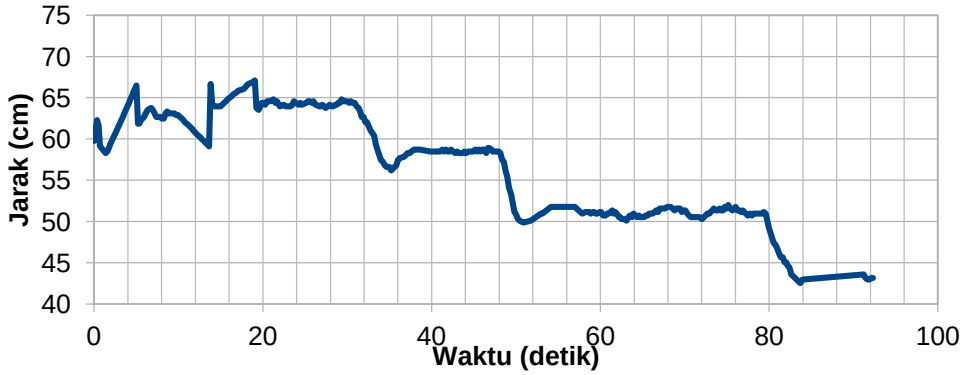
Gambar 4.11 Cuplikan gambar marka yang terlihat oleh ROV saat pendermagaan otomatis.



Gambar 4.12 Posisi terhadap sumbu x (atas) dan sumbu y (bawah) pada domain waktu



Gambar 4.13 Eror orientasi pada domain waktu



Gambar 4.14 Estimasi jarak kamera ke marka saat ROV mendekati marka

Terdapat pula data pengestimasian jarak yang disajikan pada Gambar 4.14. Pada gambar tersebut disajikan data pengestimasian jarak terhadap waktu Pada gambar terlihat bahwa jarak marka terhadap kamera perlahan turun dengan pola seperti anak tangga. Hal tersebut dikarenakan robot hanya memiliki dua opsi dalam mendekati marka, yaitu diam atau bergerak dengan kecepatan konstan. Robot diam ketika orientasi dan posisi diluar kriteria dan bergerak ketika kriteria terpenuhi. Terlihat pada grafik bahwa seluruh proses berjalan selama 80 detik dan robot berhenti pada jarak 45cm. Semakin dekat marka, semakin besar pula marka yang terlihat hingga akhirnya tidak terlihat. Dengan data tersebut juga dapat disimpulkan bahwa proses pendermagaan otomatis dilakukukan dengan kecepatan 0.25cm/detik.

4.8 Pengujian pengaruh jarak terhadap tingkat keberhasilan pendermagaan dan waktu yang dibutuhkan

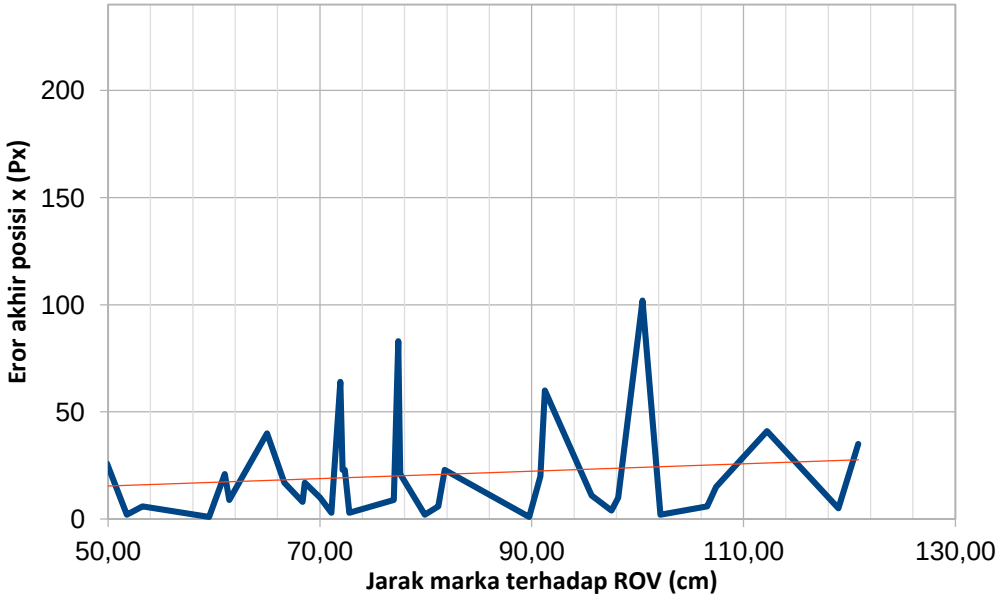
Tujuan dari pendermagaan otomatis adalah memposisikan ROV terhadap dermaga seakurat mungkin. Keberhasilan pendermagaan diukur dari posisi akhir ROV terhadap dermaga. Posisi akhir yang akurat lebih diprioritaskan daripada kecepatan pendermagaan. Apabila ROV berhasil memposisikan dirinya lebih dekat dengan setpoin maka semakin akurat pula perdermagannya. Oleh karena itu sangatlah penting untuk mengukur akurasi pendermagaan.

Parameter akurasi diukur dari kedekatan posisi x, posisi y, dan orientasi terhadap setpoint pada akhir pendermagaan. Pada bagian ini dilakukan pengujian untuk mengetahui tingkat akurasi pendermagaan dibandingkan dengan jarak. Pengujian dilakukan dengan cara meletakkan ROV dengan jarak tertentu dari dermaga, kemudian parameter akurasi diukur. Jarak terjauh dari percobaan ini adalah 120 cm karena marka hanya dapat dideteksi dengan jarak maksimal 120cm didalam air.

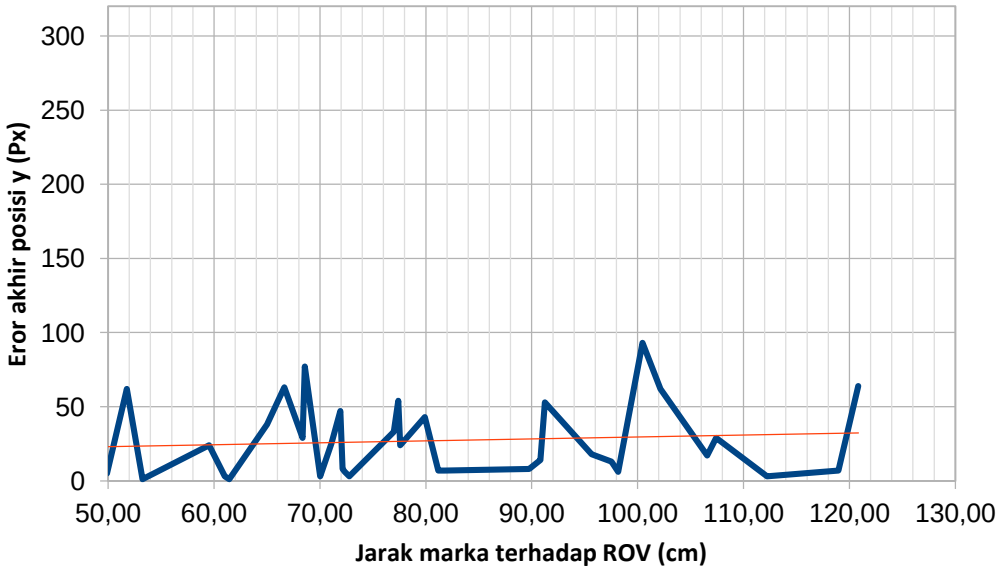
Pengaruh dari akhir posisi x ROV terhadap jarak dapat dilihat pada Gambar 4.15. Pada gambar tersebut hasil eror akhir posisi x terlihat fluktuatif (garis biru) dengan tren yang sedikit naik (garis oranye). Rata-rata eror akhir posisi x adalah 20.8 *pixel* dengan standar deviasi 23.8 *pixel*. Eror terbesar yang mungkin

terjadi pada posisi x adalah 240 *pixel*. Rata-rata eror akhir posisi x adalah 8.67% dibandingkan dengan eror terbesar.

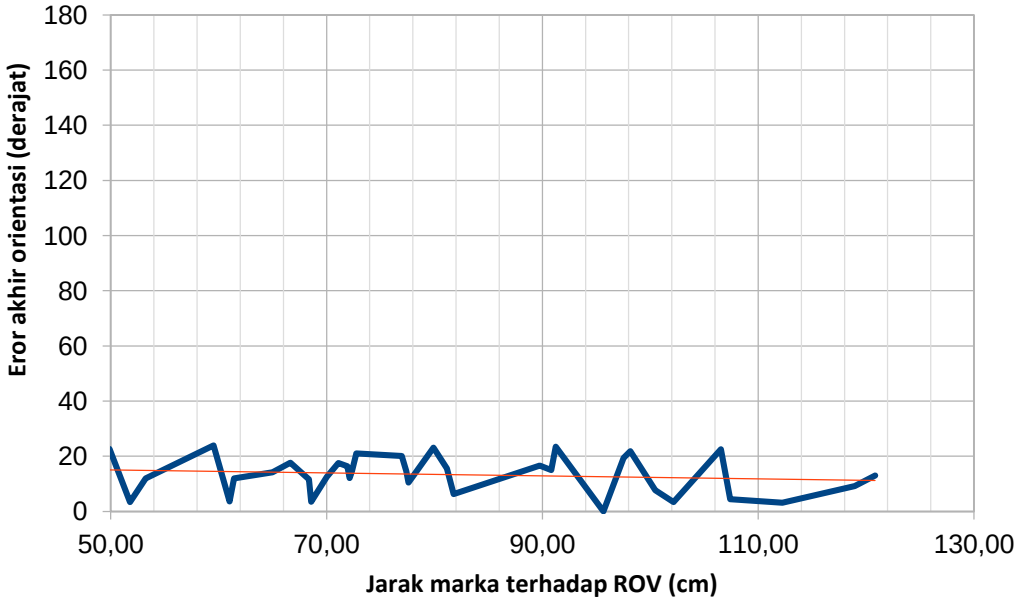
Pengaruh dari akhir posisi y ROV terhadap jarak dapat dilihat pada Gambar 4.16. Sama seperti akhir eror posisi x, akhir error posisi y terlihat fluktuatif (garis biru) dengan tren yang sedikit naik (garis oranye). Kenaikan tren akhir eror posisi x dan posisi y sangat kecil sehingga dapat disimpulkan bahwa jarak tidak begitu mempengaruhi akhir posisi x dan y ROV.



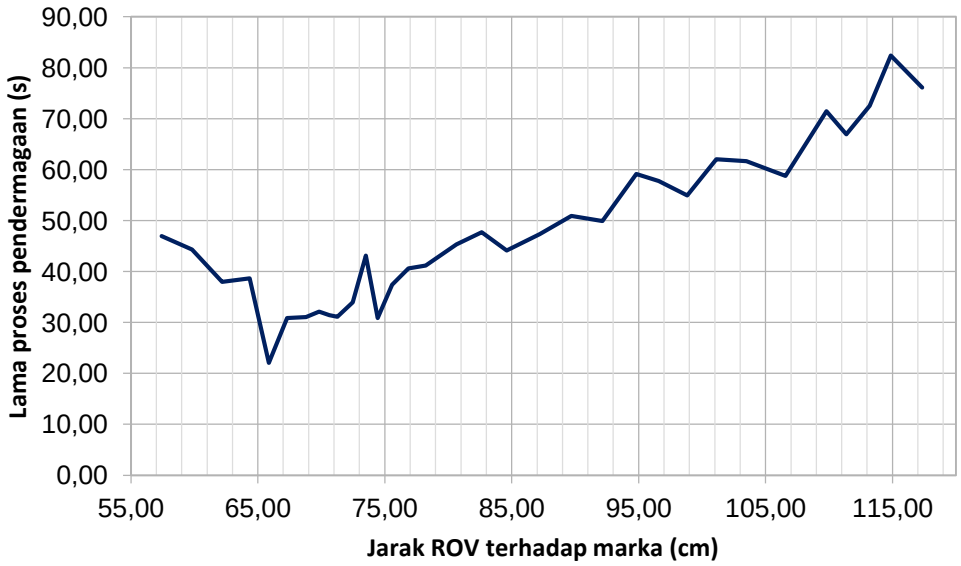
Gambar 4.15 Jarak terhadap akhir eror posisi x



Gambar 4.16 Jarak terhadap akhir eror posisi y



Gambar 4.17 Jarak terhadap akhir eror orientasi



Gambar 4.18 Jarak terhadap lama proses pendermagaan

Rata-rata eror akhir posisi y adalah 27.14 *pixel* dengan standar deviasi 25.26 *pixel*. Eror terbesar yang mungkin terjadi pada posisi y adalah 320 *pixel*. Rata-rata eror akhir posisi y adalah 8.48% dibandingkan dengan eror terbesar.

Pengaruh dari akhir posisi y ROV terhadap jarak dapat dilihat pada Gambar 4.17. Sama seperti pada eror akhir posisi x dan y, eror akhir orientasi nampak fluktuatif dengan. Namun, pada eror akhir orientasi terjadi tren turun (garis

oranye). Hal ini dikarenakan semakin jauh jarak ROV terhadap dermaga, marka nampak semakin kecil. Marka yang nampak kecil menambahkan ruang gerak untuk mengoreksi eror sudut orientasi.

Rata-rata eror akhir orientasi adalah 13.3^{o} dengan standar deviasi 6.93^{o}. Eror terbesar yang dapat terjadi pada kontrol orientasi adalah 180^{o}. Rata-rata eror akhir orientasi adalah 7.43% dibandingkan eror maksimal.

Pada Gambar 4.18 disajikan data pengaruh jarak terhadap kecepatan waktu yang dibutuhkan dalam pendermagaan. Pada grafik terlihat bahwa waktu tercepat dalam pendermagaan adalah pada saat ROV berjarak 66 cm dari marka. Pada jarak jarak yang lebih besar dari 66 cm waktu yang dibutuhkan proses pendermagaan bertambah. Pada jarak lebih kecil dari 66 cm juga terjadi peningkatan waktu yang dibutuhkan pendermagaan. Hal ini terjadi karena marka nampak lebih besar dari dekat. Ketika marka berada di dekat kamera, sisi atau pojokan marka dapat dengan mudah keluar dari medan pandang kamera.

4.9 Pengujian pengaruh kekeruahn terhadap keberhasilan pendeteksian

Tujuan dari pengujian ini adalah untuk melihat performa pengenalan marka terhadap kekeruhan air. Pengujian dilakukan pada akuarium dengan volum tetap yaitu 90x90x90cm^3 air. Perlahan-lahan 5 sendok semen ditambahkan pada akuarium untuk menambah gangguan visibilitas. Kekeruhan diukur dengan turbidimeter. Pada setiap tingkat kekeruhan, tingkat pendeteksian marka dihitung. Persentase keberhasilan masing-masing tingkat kekeruhan dapat dilihat pada Tabel 4.5

Tabel 4.5 Respon konstanta kontrol PID terhadap kontrol orientasi

Kekeruhan	Turbidity (NTU)	Persentase keberhasilan pendeteksian
1	2.79	100
2	5.58	96.1
3	8.37	100
4	11.16	100
5	13.95	97.9
6	16.75	98.7
7	19.54	0.0

Pada kekeruhan tingkat ketujuh marka menjadi tidak dapat terdeteksi sama sekali. Banyaknya partikel pengganggu dalam air menyebabkan berkurangnya cahaya yang diterima oleh kamera. Cuplikan gambar yang ditangkap kamera saat pengujian dapat dilihat pada Gambar 4.19. Pada gambar tersebut terdapat 3 tingkat kekeruhan. Pada tingkat yang ketiga (gambar paling bawah) marka tidak terdeteksi karena lingkaran marka nampak terlalu kecil oleh filter ukuran dan dianggap sebagai *noise*. Linlgkaran marka nampak lebih kecil karena cahayanya lebih terdistribusi. Hal ini menyebabkan berkurangnya cahaya yang tertangkap kamera

Gambar 4.19 Tiga tingkat kekeruhan dan pengaruhnya terhadap pendeteksian.

4.10 Pengujian pengaruh kemiringan marka terhadap keberhasilan pendermagaan

Tujuan dari pengujian ini adalah untuk melihat performa pengenalan pendermagaan terhadap marka yang dimiringkan. Kemiringan marka mempengaruhi bentuk dan ukuran lingkaran pada marka. Lingkaran penanda dari marka akan nampak oval apabila dilihat dari samping.

Tabel 4.6 Tingkat keberhasilan pendermagaan terhadap kemiringan marka

Kemiringan marka (derajat)	Percobaan Ke- (0 gagal, 1 berhasil)										Keberhasilan pendermagaan (%)
	1	2	3	4	5	6	7	8	9	10	
10	1	1	1	1	1	1	1	1	1	1	100
20	1	1	1	1	1	1	1	1	1	1	100
30	1	1	1	1	0	0	0	0	1	1	60
35	1	1	0	0	1	0	0	0	1	0	40
40	1	0	0	0	0	0	0	0	0	0	10

Percobaan dilakukan dengan melakukan sepuluh kali pendermagaan otomatis dalam satu kemiringan marka. Pada sepuluh kali percobaan dihitung seberapa banyak ROV berhasil mendekati marka. Data dari percobaan ini dapat dilihat pada Tabel 4.6. Pada tabel dapat dilihat bahwa hanya dengan memiringkan marka 30^o maka keberhasilan pendermakan turun menjadi 60%.

BAB 5
KESIMPULAN

Marka yang di desain telah berhasil diterapkan untuk membantu proses pendermagaan otomatis. Marka dapat dideteksi sampai dengan 120 cm dari kamera dengan persentasi keberhasilan diatas 90%. Marka dapat dideteksi meskipun telah dimiringkan 50^o dengan tingkat persentase keberhasilan diatas 80%. Marka dapat digunakan untuk mengestimasi jarak dengan rata-rata eror 0.62 cm. Pengestimasi jarak juga telah dibuktikan stabil dengan standar devasi 2.83 cm. Pengestimasi sudut memiliki eror rata-rata 1.75^o dan terbukti stabil dengan standar deviasi 0.5^o. Algoritma yang dibuat dapat diterapkan untuk pendermagaan otomatis dengan kecepatan 0.25cm/s dengan waktu pendeteksian marka 0.26 detik setiap cuplikan. Kontrol posisi dari metode memberikan fluktuasi eror 31 *pixel*. Kontrol posisi menggunakan metode kontrol PID berhasil dilakukan dengan *rise time* 9.07 detik. Pendermagaan dengan kamera sebagai umpan balik posisi dan orientasi berhasil dilakukan dengan kesalahan posisi 8.67% dan kesalahan orientasi 7.43% dari kesalahan maksimal. Pendermagaan berlangsung selama 80 detik dengan jarak mula ROV ke dermaga 65 cm.

Halaman ini sengaja dikosongkan

DAFTAR PUSTAKA

[1] John F Bruno, Elizabeth R Selig, and Kenneth S Casey, "Thermal Stress and Coral Cover as Drivers of Coral Disease Outbreaks," *PLOS*, May 2007.

[2] O. Hoegh-Guldberg, P. J. Mumby, and A. J. Hooten, "Coral Reefs Under Rapid Climate Change and Ocean Acidification," *Science*, vol. 318, no. 5857, pp. 1737–1742, Dec. 2007.

[3] H ERMAN S. J. C ESAR, "Coral Reefs: Their Functions, Threats and Economic Value," in *Collected Essays on the Economics of Coral Reefs*, Sweden: CORDIO, 2000, pp. 14–39.

[4] Andreas H. Muljadi and Marthen Welly, "Protokol Biofisik Monitoring Kesehatan Karang KKP Nusa Penida," Coral Triangle Center, Bali.

[5] Richard D. Vann, Petar J. Denoble, and Joel A. Dovenbarger, "Report on Decompression Illness, Diving Fatalities and Project Dive Exploration," Divers Alert Network, 0-9673066-7–1, 2004.

[6] Robert D. Christ and Robert L. Wernli, "The ROV Manual," in *The ROV Manual*, Second Edition., Waltham: Elsevier, 2014, pp. 3–5.

[7] Ottar L. Osen, Rolf-Inge Sandvik, Jørgen Berge Trygstad, Vegard Rogne, and Houxiang Zhang, "A Novel Low Cost ROV for Aquaculture," in *OCEANS – Anchorage, 2017*, Anchorage, AK, USA, USA, 2017.

[8] Zainah Md. Zain, Maziyah Mat Noh, and Khairul Ashraff Ab Rahim, "Design and development of an X4-ROV," in *Underwater System Technology: Theory and Applications (USYS)*, Penang, Malaysia, 2016.

[9] Hyeung-sik Choi, Hanil Park, and Sanki Chung, "Design and control of a convertible ROV," presented at the OCEANS, 2012 - Yeosu, Yeosu, South Korea, 2012.

[10] Andrea Arienti, Marcello Calisti, and Francesco Giorgio-Serchi, "PoseiDRONE: Design of a soft-bodied ROV with crawling, swimming and manipulation ability," presented at the Oceans, San Diego, 2013.

[11] Angelo Odetti, Giorgio Bruzzone, and Massimo Caccia, "P2-ROV a portable/polar ROV," presented at the OCEANS 2017, Aberdeen, 2017.

[12] Jinwoo Choi, Yeongjun Lee, and Taejin Kim, "Development of a ROV for visual inspection of harbor structures," presented at the 2017 IEEE Underwater Technology (UT), Busan, 2017.

[13] Killian Poore, Christopher Kitts, and Geoffrey Wheat, “A small scale ROV for shallow-water science operations,” presented at the OCEANS 2016, Monterey, 2016.

[14] Norimitsu Sakagami, Mizuho Shibata, and Hideki Hashizume, “Development of a Human-Sized ROV with Dual-Arm,” in *OCEANS’10 IEEE SYDNEY*, Sydney, 2010.

[15] P. Sotiropoulos, D. Grosset, and G. Giannopoulos, “AUV docking system for existing underwater control panel,” in *OCEANS 2009 - EUROPE*, Bremen, 2009.

[16] Du Jun, Meng Lingshuai, and Gu Haitao, “Optimal design of clamping mechanism for AUV underwater docking device based on Kriging model,” in *2017 2nd Asia-Pacific Conference on Intelligent Robot Systems (ACIRS)*, 2017.

[17] P.B. Sujit, A.J. Healey, and J.B. Sousa, “AUV docking on a moving submarine using a K-R navigation function,” in *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*, San Francisco, 2011.

[18] Nuno A. Cruz, Anibal C. Matos, and Rui M. Almeida, “A lightweight docking station for a hovering AUV,” in *2017 IEEE Underwater Technology (UT)*, Busan, 2017.

[19] Minsung Sung and Soncheol Yu, “Mini ROV based anchoring AUV system TreeBot AUV,” in *2016 IEEE/OES Autonomous Underwater Vehicles (AUV)*, Tokyo, 2016.

[20] Liam Paull, Sajad Saeedi, and Mae Seto, “AUV Navigation and Localization: A Review,” *IEEE Journal of Oceanic Engineering*, vol. 39, no. 1, pp. 131–149, Dec. 2013.

[21] Ken Teo, Benjamin Goh, and Oh Kwee Chai, “Fuzzy Docking Guidance Using Augmented Navigation System on an AUV,” *IEEE Journal of Oceanic Engineering*, vol. 40, no. 2, pp. 349–361, May 2014.

[22] Kyungmin Kwak, Daegil Park, and Wan Kyun Chung, “Manta ROV docking sequence using 3-D Omni-directional antenna’s signal attenuation,” in *Ubiquitous Robots and Ambient Intelligence (URAI)*, Jeju, South Korea, 2017.

[23] A. V. Inzartsev, A. M. Pavin, and N. I. Rylov, “Development of the AUV automatic docking methods based on echosounder and video data,” presented at the 2017 24th Saint Petersburg International Conference on Integrated Navigation Systems (ICINS), St. Petersburg, 2017.

[24] Albert Sans-Muntadas, Kristin Y. Pettersen, and Edmund Brekke, “Learning an AUV docking maneuver with a convolutional neural network,” in *OCEANS - Anchorage, 2017*, Anchorage, 2017.

[25] Myo Myint, Kenta Yonemori, and Akira Yanou, "Dual-eyes visual-based sea docking for sea bottom battery recharging," in *OCEANS 2016 MTS/IEEE Monterey*, Monterey, CA, USA, 2016.

[26] Zhuoyuan Song and Kamran Mohseni, "Automated AUV docking control with light-field imaging," in *OCEANS - Anchorage*, Anchorage, 2017.

[27] A.A. Kushnerik, A.V. Vorontsov, and A.Ph. Scherbatyuk, "Small AUV docking algorithms near dock unit based on visual data," in *OCEANS 2009, MTS/IEEE Biloxi - Marine Technology for Our Future: Global and Local Challenges*, Biloxi, 2009.

[28] M. F. Yahya and M. R. Arshad, "Tracking of multiple markas based on color for visual servo control in underwater docking," in *Control System, Computing and Engineering (ICCSCE)*, George Town, Malaysia, 2016.

[29] Jin-YeongPark, Bong-huanJun, Pan-mook Lee, and Junho Oh, "Experiments on vision guided docking of an autonomous underwater vehicle using one camera," *Ocean Engineering*, vol. 36, no. 1, pp. 48–61, Jan. 2009.

[30] Yoshiki Sato, Toshihiro Maki, and Kotohiro Masuda, "Autonomous docking of hovering type AUV to seafloor charging station based on acoustic and visual sensing," in *2017 IEEE Underwater Technology (UT)*, Busan, 2017.

[31] Gary Bradski and Adrian Kaehler, *Learning OpenCV*, First Edition. O'Reilly Media, 2008.

[32] Leonid Sigal, "Lecture 3 (Marka-based) Motion Capture," 21-Sep-2012.

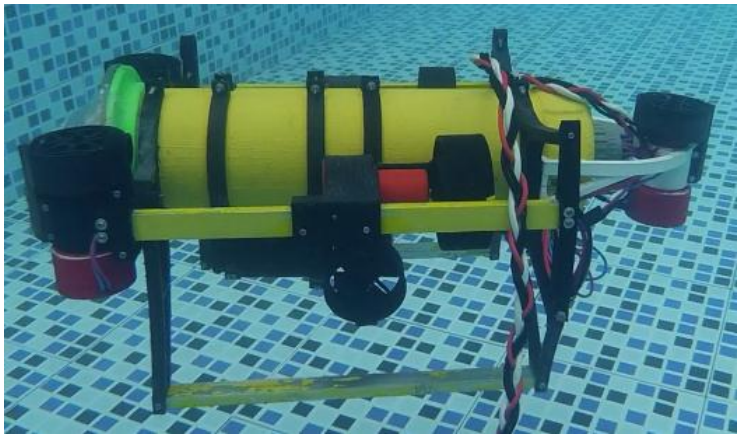
[33] Sanni Siltanen, *Theory and applicationsof marka-basedaugmented reality*. Espoo: Julkaisija utgivare publisher, 2012.

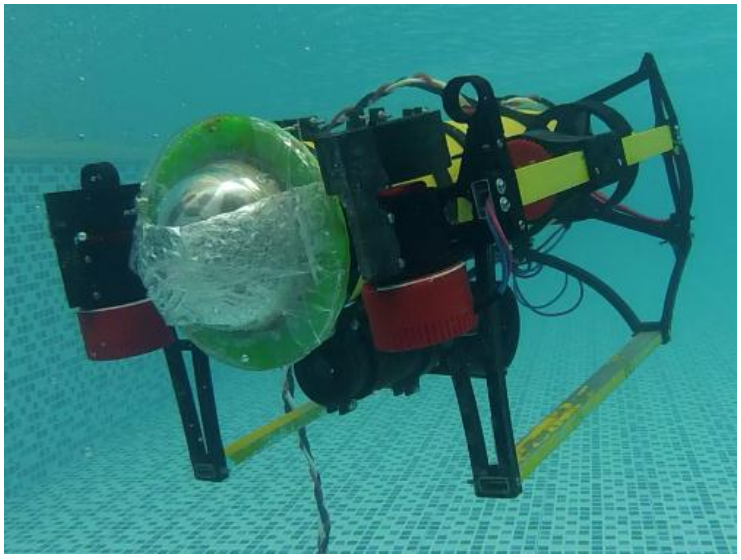
[34] Djoko Purwanto, "Control Design Principles," Surabaya, 23-Feb-2017.

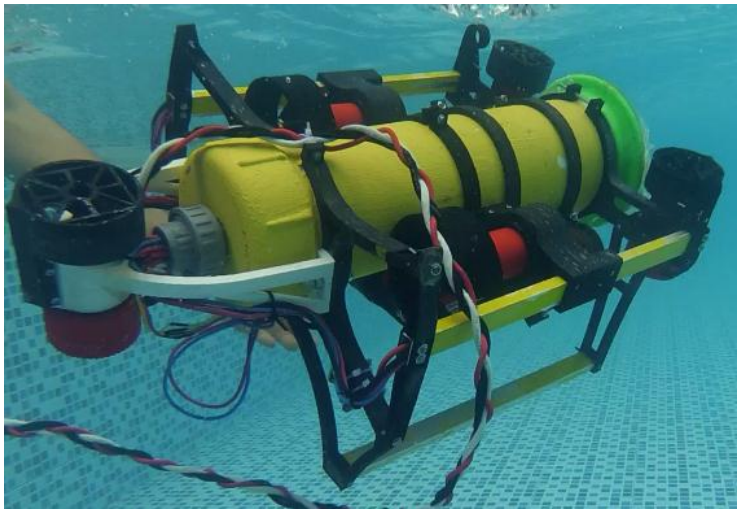
Halaman ini sengaja dikosongkan

LAMPIRAN

Foto-foto ROV







Program pada Arduino

```
#include <MS5803_14.h>
#include <Wire.h>
#define gyro_address 0x68
#define ms5803_address 0x76

MS_5803 sensor = MS_5803(512);

float depth_input;
int cal_int;
int temperature;
int acc_axis[3], gyro_axis[3];
long acc_x, acc_y, acc_z, acc_total_vector;
double gyro_pitch, gyro_roll, gyro_yaw;
float gyro_roll_input, gyro_pitch_input, gyro_yaw_input;
double axis_cal[3];
bool cal_is_done=0;
float roll_level_adjust, pitch_level_adjust;
float angleAcc[3], angle[3];

volatile int ch[12];
unsigned long t[12];
const int timeoutPulse = 3000;
const int pulseMin = 1000;
const int pulseMax = 2000;
byte pulse =1;
unsigned long motorOffTimer, trackOffTimer;
int timeout = 1000;
int pwm[6];
float setPoint[8] = {0};
float theFirstPressureRead=0;
float processVar[8] = {0};
float error[8], errorOld[8];
float gainP[8], gainI[8], gainD[8];
float sigmai[8], deltad[8];
float SUMpid[8];
float OUTpid[8];
float runtimeMicro = micros();
float runtimeMicroOld, autoModeRuntimeOld;
 String command="";
 byte CAMERA_MODE = 0;
 bool isAutoYaw=0;
 bool isSelectedFirstTime=1;
 bool parsingSucceed=0;
 unsigned long int noInputTimeout;
 float pitchThrotleOffset=0;
 bool overideAutoYaw=0;
```

```
void setup() {
  Serial.begin(57600);
  for (int i=8; i<14; i++){
    pinMode(i, KELUARAN);
  }
  for (int i=0; i<5; i++){
     pinMode(22+i*2,KELUARAN);
     pinMode(23+i*2,KELUARAN);
  }
  pinMode(32,KELUARAN);
  digitalWrite(32,1);
  pinMode(33,KELUARAN);
  digitalWrite(33,1);
  pinMode(34,KELUARAN);
  pinMode(35,KELUARAN);
  pinMode(44,KELUARAN);
  digitalWrite(33,1);
  pinMode(7,KELUARAN);
  digitalWrite(7,1);
  pinMode(5,KELUARAN);
  pinMode(6,KELUARAN);
  Wire.begin();
  TWBR = 12;
  digitalWrite(13,HIGH);

  sensor.initializeMS_5803(false);
  mpu6050_register();
  for (cal_int = 0; cal_int < 1250 ; cal_int ++){
    motor_stop();
  }
  for (cal_int = 0; cal_int < 1000 ; cal_int ++){
    if(cal_int % 15 == 0)digitalWrite(13, !digitalRead(13));
    mpu6050_signal();
    raw_to_angle();
    axis_cal[0] += angle[0];
    axis_cal[1] += angle[1];
    axis_cal[2] += angle[2];
  }
  axis_cal[0] /= 1000;
  axis_cal[1] /= 1000;
  axis_cal[2] /= 1000;
  cal_is_done=1;
  motor_stop();
  digitalWrite(13,LOW);
  pinMode(3, INPUT_PULLUP);
  attachInterrupt(digitalPinToInterrupt(3), readPPM, RISING);
```

```
  t[0]=micros();
            gainP[3]=13;
            gainD[3]=0.05;
            gainI[3]=0.03;
            gainP[2]=150;
            gainD[2]=0.1;
            gainI[2]=0.2;
            gainP[0]=5;
            gainD[0]=0.5;
            gainI[0]=0.1;
            gainP[1]=7;
            gainD[1]=0.5;
            gainI[1]=0.1;
  for (int i=0; i<3; i++) setPoint[i]=0;
  sensor.readSensor();
  setPoint[3] = sensor.pressure();
  theFirstPressureRead = setPoint[3];
}

void loop() {
  sensor.readSensor();
  float depth_input_now = sensor.pressure();
  depth_input = depth_input_now;
  mpu6050_signal();
  raw_to_angle();
  pidGet();
  setpointSet();
  motorPIDOut();
 modeSelect ();
}

float mapping(float x, float in_min, float in_max, float out_min, float out_max){
  return (x - in_min) * (out_max - out_min) / (in_max - in_min) + out_min;
}

void readPPM() {
    t[pulse] = micros();
    switch (pulse) {
      case 0:
        pulse++;
        break;

      case 1:
        ch[1] = t[1] - t[0];
        pulse++;
        if (ch[1] > timeoutPulse) {
          pulse = 1;
```

```
    t[0] = t[1];
    }
  break;

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

case 3:
  ch[3] = t[3] - t[2];
  pulse++;
  if (ch[3] >timeoutPulse ) {
    pulse = 1;
    t[0] = t[3];
    }
  break;

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

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

case 6:
  ch[6] = t[6] - t[5];
  pulse++;
  if (ch[6] > timeoutPulse) {
    pulse = 1;
    t[0] = t[6];
    }
```

```
  break;

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

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

case 9:
  ch[9] = t[9] - t[8];
  pulse++;
  if (ch[9] > timeoutPulse) {
    pulse = 1;
    t[0] = t[9];
    }
  break;

case 10:
  ch[10] = t[10] - t[9];
  pulse++;
  if (ch[10] > timeoutPulse) {
    pulse = 1;
    t[0] = t[10];
    }
  break;

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

default:
  pulse=1;
  break;
}
```

```
  }
void motor_stop(){
  for (int i = 0; i<6; i++) motorWrite(i, 0);
}

void mpu6050_signal(){
  Wire.beginTransmission(gyro_address);
  Wire.write(0x3B);
  Wire.endTransmission();
  Wire.requestFrom(gyro_address,14);

  while(Wire.available()<14);
  acc_axis[0] = Wire.read()<<8|Wire.read();
  acc_axis[1] = Wire.read()<<8|Wire.read();
  acc_axis[2] = Wire.read()<<8|Wire.read();
  temperature = Wire.read()<<8|Wire.read();
  gyro_axis[0] = Wire.read()<<8|Wire.read();
  gyro_axis[1] = Wire.read()<<8|Wire.read();
  gyro_axis[2] = Wire.read()<<8|Wire.read();

  gyro_roll = gyro_axis[0];
  gyro_pitch = gyro_axis[1];
  gyro_yaw = gyro_axis[2];

  acc_x = acc_axis[0];
  acc_y = acc_axis[1];
  acc_z = acc_axis[2];
}

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

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

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

  Wire.beginTransmission(gyro_address);
  Wire.write(0x1B);
```

```
  Wire.endTransmission();
  Wire.requestFrom(gyro_address, 1);

  while(Wire.available() < 1);

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

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

void raw_to_angle(){
  gyro_roll_input = (gyro_roll_input * 0.7) + ((gyro_roll / 65.5) * 0.3);
  gyro_pitch_input = (gyro_pitch_input * 0.7) + ((gyro_pitch / 65.5) * 0.3);
  gyro_yaw_input = (gyro_yaw_input * 0.7) + ((gyro_yaw / 65.5) * 0.3);

  angle[1] += gyro_pitch * 0.0000611;
  angle[0] += gyro_roll * 0.0000611;
  angle[2] += gyro_yaw * 0.00000611+0.000233;

  angle[1] -= angle[0] * sin(gyro_yaw * 0.000001066);
  angle[0] += angle[1] * sin(gyro_yaw * 0.000001066);
  acc_total_vector = sqrt((acc_x*acc_x)+(acc_y*acc_y)+(acc_z*acc_z));

  if(abs(acc_y) < acc_total_vector){
    angleAcc[1] = asin((float)acc_y/acc_total_vector)* 57.296;
    }
  if(abs(acc_x) < acc_total_vector){
    angleAcc[0] = asin((float)acc_x/acc_total_vector)* -57.296;
    }

  angleAcc[1] -= 0.0;
  angleAcc[0] -= 0.0;

  angle[1] = angle[1] * 0.9 + angleAcc[1] * 0.1;
  angle[0] = angle[0] * 0.9 + angleAcc[0] * 0.1;

  pitch_level_adjust = angle[1] * 15;
  roll_level_adjust = angle[0] * 15;
}

void motorWrite(int index, float speeds){
```

```
    bool dirrection=0;
    if (speeds<0) dirrection=!dirrection;
    if (index<5){
       index=index+8;
       digitalWrite(-2*index+46,dirrection);
       digitalWrite(-2*index+47,!dirrection);
       analogWrite(index, (int)abs(speeds));
      }
    else{
       digitalWrite(34,dirrection);
       digitalWrite(35,!dirrection);
       analogWrite(44, (int)abs(speeds));
      }
  }

void pidGet(){
  runtimeMicro = millis();
  float getPIDinterval= (runtimeMicro-runtimeMicroOld)/1000;
  runtimeMicroOld = runtimeMicro;
  processVar[0] = angle[0]-axis_cal[0];
  processVar[1] = angle[1]-axis_cal[1];
  processVar[2] = angle[2]-axis_cal[2];
  processVar[3] = depth_input;
  for (int i=0; i<4; i++) error[i]=setPoint[i]-processVar[i];
  for (int i=0; i<4; i++) sigmai[i] = sigmai[i] + gainI[i]*error[i]*getPIDinterval;
  for (int i=0; i<4; i++) deltad[i]=(gainD[i]*error[i] -
gainD[i]*errorOld[i])/getPIDinterval;
  for (int i=0; i<4; i++) errorOld[i] = error[i];
  for (int i=0; i<4; i++) SUMpid[i] = error[i] + sigmai[i] + deltad[i];
  for (int i=0; i<4; i++) OUTpid[i] = gainP[i] * SUMpid[i];
 }

void motorPIDOut(){
  float throInput, slideRL;
  throInput = constrain (ch[3], pulseMin, pulseMax);
  throInput = mapping(throInput, pulseMin, pulseMax, 255, -255);
  if ((ch[3]<1650)&&(ch[3]>1350))throInput=0;

  slideRL = constrain (ch[1], pulseMin, pulseMax);
  slideRL = mapping(slideRL, (float)pulseMin, (float)pulseMax, -255, 255);
  if ((ch[1]<1600)&&(ch[1]>1400))slideRL=0;
  if (!isAutoYaw && !overideAutoYaw) {
    OUTpid[2] = constrain (ch[4], pulseMin, pulseMax);
    OUTpid[2] = mapping(OUTpid[2], pulseMin, pulseMax, 255, -255);
    axis_cal[2] = angle[2];
    setPoint[2] = 0;
    sigmai[2]=0;
```

```
    }
  if (ch[9]<1050) {
    if (millis()-motorOffTimer>timeout){
    digitalWrite(32,0);
    digitalWrite(33,0);
    axis_cal[2] = angle[2];
    setPoint[2] = 0;
    setPoint[3] = depth_input;
    for (int i = 0; i<4; i++){
      sigmai[i]=0;
      }
    }
  }
  else if (ch[9]>1900){
       motorOffTimer=millis();
       digitalWrite(32,1);
       digitalWrite(33,1);
    }

  float m[6];
  m[0] =  OUTpid[3] - OUTpid[0] + OUTpid[5];
  m[1] = -OUTpid[3] - OUTpid[0] - OUTpid[1] - OUTpid[5];
  m[2] =  OUTpid[3] + OUTpid[0] - OUTpid[1] + OUTpid[5];
  m[3] = -throInput - OUTpid[2] - OUTpid[7] + OUTpid[4];
  m[4] =  throInput - OUTpid[2] + OUTpid[7] + OUTpid[4];

  if (ch[6]<1500){
    m[0]=0; m[1]=0; m[2]=0;
    }


  for (int i=0; i<6; i++){
    m[i]=constrain(m[i], -255, 255);
    motorWrite(i,m[i]);
    }
}

void setpointSet(){
  float yawInput, depthInputRemote, slideRL;
  if (ch[2]<pulseMin) ch[2]= 1500;
  depthInputRemote = constrain (ch[2], pulseMin, pulseMax);
  depthInputRemote = mapping(depthInputRemote, (float)pulseMin,
(float)pulseMax, -0.1, 0.1);
  if ((ch[2]<1600)&&(ch[2]>1400))depthInputRemote=0;
  setPoint[3]=setPoint[3]+depthInputRemote;
  if (isAutoYaw){
    yawInput = constrain (ch[4], pulseMin, pulseMax);
```

```
    yawInput = mapping(yawInput, (float)pulseMin, (float)pulseMax, 0.01, -0.01);
    if ((ch[4]<1600)&&(ch[4]>1400))yawInput=0;
    setPoint[2]=setPoint[2]+yawInput;
  }
}

void dataParsing(){
  parsingSucceed=0;
  while (Serial.available()>0){
    char aChar = Serial.read();
    if (aChar=='\n'){
      String distance, orientation, xPos, yPos;
      orientation   = command.substring(command.indexOf('$')+1,
command.indexOf('@'));
      distance      = command.substring(command.indexOf('@')+1,
command.indexOf('~'));
      xPos          = command.substring(command.indexOf('~')+1,
command.indexOf('!'));
      yPos          = command.substring(command.indexOf('!')+1,
command.indexOf('#'));

            processVar[4] = constrain (orientation.toFloat(),-180, 180);
      processVar[5] = constrain (distance.toFloat(), 15, 300);
      processVar[5] = processVar[5]>299 ? 0 : processVar[5];
      processVar[6] = constrain (xPos.toInt(), 0, 640);
      processVar[7] = constrain (yPos.toInt(), 0, 480);
      parsingSucceed=1;
      command = "";
      }
    else command = command + aChar;
    }
  }
void noDataParsing(){
      processVar[4] = 0;
      processVar[5] = 0;
      processVar[6] = 0;
      processVar[7] = 0;
  }

float expFilter(float input, float smoothBefore, float alpha){
  return smoothBefore+alpha*(input-smoothBefore);
  }

void modeSelect (){
  if (ch[8]<1180){
    digitalWrite(7,1);
    digitalWrite(5,0);
```

```
    CAMERA_MODE=0;
    noDataParsing();
    }
  else if ((ch[8]>1400)&&(ch[8]<1600)){
    if (millis()-trackOffTimer>timeout){
      digitalWrite(7,0);
      digitalWrite(5,0);
      CAMERA_MODE=1;
      overideAutoYaw = 0;
      noDataParsing();
      isSelectedFirstTime=1;
      for (int i=4; i<8; i++){
        sigmai[i]=0;
        OUTpid[i]=0;
        }
    }
  }
  else if (ch[8]>1700){
    trackOffTimer = millis();
    digitalWrite(7,0);
    digitalWrite(5,1);
    CAMERA_MODE=2;
    if (isSelectedFirstTime){
      autoModeRuntimeOld=millis();
      isSelectedFirstTime=0;
    }
    gainP[4]=0.6; gainI[4]=0.005; gainD[4]=0.0;
    gainP[6]=1; gainI[6]=0.018; gainD[6]=0.65;
    gainP[7]=0.6; gainI[7]=0.052; gainD[7]=0.6;
    float autoModeRuntime = millis();
    setPoint[4]=0;
    setPoint[5]=17;
    setPoint[6]=640/2;
    setPoint[7]=480/2;
    float autoModeRuntimeInterval = (autoModeRuntime-
autoModeRuntimeOld)/1000;
    autoModeRuntimeOld = autoModeRuntime;
    dataParsing();
    if (parsingSucceed){
      for (int i=4; i<8; i++) error[i]=setPoint[i]-processVar[i];
      noInputTimeout=millis();
      float radius = sqrt((320 - processVar[6])*(320 - processVar[6]) + (240 -
processVar[7])*(240 - processVar[7]));
      if (radius>33){
        error[5]=0;
        }
      else{
```

```
                    axis_cal[2] = angle[2];
        setPoint[2] = 0;
        sigmai[2]=0;

        if (abs(error[4])<25){
          setPoint[3] = setPoint[3]-3;
          }
       }

   }
   if ((!parsingSucceed)&&((millis()-noInputTimeout)>1500)) {
      for (int i=4; i<8; i++){
        error[i]=error[i]*0.8;
        sigmai[i]=sigmai[i]*0.8;
        }
   }

   for (int i=4; i<8; i++) sigmai[i] = sigmai[i] +
gainI[i]*error[i]*autoModeRuntimeInterval;
   for (int i=4; i<8; i++) deltad[i]=(gainD[i]*error[i] -
gainD[i]*errorOld[i])/autoModeRuntimeInterval;
   for (int i=4; i<8; i++) errorOld[i] = error[i];
   for (int i=4; i<8; i++) SUMpid[i] = error[i] + sigmai[i] + deltad[i];
   for (int i=4; i<8; i++) OUTpid[i] = gainP[i] * SUMpid[i];
  }
  if ((ch[10]<1500)||(ch[8]>1600)) isAutoYaw = 0;
  else if ((ch[10]>1500)&&(ch[8]<1600)) isAutoYaw = 1;
}
```

Program pada Raspberry Pi

```
#include <iostream>
#include <math.h>
#include <vector>
#include <opencv2/imgproc/imgproc.hpp>
#include <opencv2/videoio/videoio.hpp>
#include <opencv2/highgui/highgui.hpp>
#include <opencv2/features2d.hpp>
#include <wiringSerial.h>
#include <wiringPi.h>
#include <stdio.h>
#include <fstream>
#include <string>

float px2dist(float pxSize, float realSize, float focalLength, float cam2pixRatio){
   return (focalLength*realSize*cam2pixRatio)/pxSize;
}

float mapping (float input, float in_min, float in_max, float out_min, float
out_max){
   return (input - in_min)*(out_max-out_min)/(in_max-in_min)+out_min;
}

int main(int argc, char *argv[])
{
        int BYNARYTRESHOLD = 220;
        if (argc > 1){
                std::string temp (argv[1]);
                BYNARYTRESHOLD = std::stoi(temp);

        std::cout<<"BYNARYTRESHOLD="<<BYNARYTRESHOLD<<"\n";
        }
   cv::VideoCapture cap;
   cv::namedWindow("show",CV_WINDOW_NORMAL);
   cv::setWindowProperty("show", CV_WND_PROP_FULLSCREEN,
CV_WINDOW_FULLSCREEN);
   cap.open(0);
   if (!cap.isOpened()) {
      std::cout<<"cannot open camera"<<std::endl;
      return 1;
   }
   cv::Mat frameCap;
   cv::Mat inputImg;

   typedef struct arrayPointScalar{
```

```
    cv::Point pos;
    uchar val;
}arrayPointScalar;

typedef struct projectionAngle{
    float x_angle;
    float y_angle;
}projectionAngle;

float vertices = 11;
float focalLength = 0.0304;
float cam2pixRatio = 16627.939021282;
float deg2rad = 0.0174533;
float rad2deg = 57.3;
float FOV = 58;

wiringPiSetupPhys();
pinMode(11, INPUT);
pullUpDnControl (11, PUD_UP);
pinMode(13, INPUT);
pullUpDnControl (13, PUD_UP);
int fd =  serialOpen ("/dev/ttyUSB0", 57600);
if (fd < 0){
    std::cout << "cannot open serial port\n";
    return 1;
}

cv::VideoWriter keluaranVideo;
std::fstream logFile;
bool recordmodeIsOn=0;
bool isInitializing=1;

while(1){

    int historyNumber_int;
    if (digitalRead(11)==HIGH) recordmodeIsOn = 1;
    else recordmodeIsOn = 0;

    if (recordmodeIsOn && isInitializing){
        std::fstream historyFile;
        historyFile.open("historyFile.txt", std::fstream::in | std::fstream::out);
        historyFile.seekg(std::fstream::beg);
        char charArr[7];
        historyFile.getline(charArr,7);
        std::string historyNumber_str(charArr);
        historyNumber_int = std::stoi(historyNumber_str) + 1;
        historyFile.seekg(std::fstream::beg);
```

```
            historyFile<<historyNumber_int;
            historyFile.close();
            historyNumber_str = std::to_string(historyNumber_int);

            cv::Size videoSize = cv::Size((int)
cap.get(CV_CAP_PROP_FRAME_WIDTH),
                        (int) cap.get(CV_CAP_PROP_FRAME_HEIGHT));
            std::string videoFileName = "logVideo" + historyNumber_str + ".avi";

keluaranVideo.open(videoFileName,cv::VideoWriter::fourcc('H','2','6','4'),20,vide
oSize,true);
            if (!keluaranVideo.isOpened()){
               std::cout  << "\nCould not open the keluaran video for write\n";
               return -1;
            }
            std::string fileName = "logFile" + historyNumber_str + ".txt";
            logFile.open(fileName, std::fstream::in | std::fstream::out |
std::fstream::app);
            isInitializing = 0;
         }
         else if (!recordmodeIsOn && !isInitializing){
            isInitializing=1;
            logFile.close();
            keluaranVideo.release();
         }
         cap >> inputImg;
         inputImg.copyTo(frameCap);

         float cam_width = (float)frameCap.cols;
         float cam_height = (float)frameCap.rows;

         cv::Mat gray;
         cv::cvtColor(frameCap,gray,CV_BGR2GRAY);
         cv::cvtColor(frameCap,frameCap,CV_BGR2RGB);
         cv::threshold(gray, gray, BYNARYTRESHOLD, 255,
CV_THRESH_BINARY_INV);
         cv::SimpleBlobDetector::Params blobParams;
         blobParams.maxThreshold=128;
         blobParams.filterByArea=1;
         blobParams.minArea = 100;
         blobParams.maxArea = 1800;
         blobParams.filterByCircularity=1;
         blobParams.minCircularity=0.8;

         cv::Ptr<cv::SimpleBlobDetector> detector =
cv::SimpleBlobDetector::create(blobParams);
```

```
std::vector<cv::KeyPoint> blobKeyPoints;
detector->detect(gray,blobKeyPoints);

cv::Mat keluaran;
int kepointSize = blobKeyPoints.size();
if (kepointSize==5){
   std::vector<cv::Point> pointsShorted;
   pointsShorted.resize(6);
   pointsShorted[0] = blobKeyPoints[0].pt;
   pointsShorted[2] = blobKeyPoints[0].pt;
   pointsShorted[1] = blobKeyPoints[0].pt;
   pointsShorted[3] = blobKeyPoints[0].pt;
   pointsShorted[4] = blobKeyPoints[0].pt;
   pointsShorted[5] = blobKeyPoints[0].pt;

   int pivotIndex=0; int indexSum[4]={0};
   for (int i = 0; i<kepointSize-1; i++){
      cv::Point temp = blobKeyPoints[i+1].pt;
      if (temp.y < pointsShorted[0].y){
         pointsShorted[0]=temp;
         indexSum[0]=i+1;
      }
      if (temp.y > pointsShorted[2].y){
         pointsShorted[2]=temp;
         indexSum[2]=i+1;
      }
      if (temp.x < pointsShorted[1].x){
         pointsShorted[1]=temp;
         indexSum[1]=i+1;
      }
      if (temp.x > pointsShorted[3].x){
         pointsShorted[3]=temp;
         indexSum[3]=i+1;
      }
   }

   for (int i=0; i<4; i++){
      pivotIndex=pivotIndex+indexSum[i];
   }
   pivotIndex=10-pivotIndex;
   pointsShorted[4] = blobKeyPoints[pivotIndex].pt;

   int deltaX = abs(pointsShorted[1].x-pointsShorted[3].x);
   int deltaY = abs(pointsShorted[2].y-pointsShorted[0].y);
   pointsShorted[5].x=pointsShorted[1].x+deltaX/2;
   pointsShorted[5].y=pointsShorted[0].y+deltaY/2;
```

```
        float Ixy[4] = {0};
        for (int i=0; i<3; i++){
           Ixy[i]= sqrt(    (float)(std::pow((pointsShorted[i].x-
pointsShorted[i+1].x),2)
                 +std::pow((pointsShorted[i].y-pointsShorted[i+1].y),2))    );
        }
        Ixy[3]= sqrt(    (float)(std::pow((pointsShorted[0].x-pointsShorted[3].x),2)
              +std::pow((pointsShorted[0].y-pointsShorted[3].y),2))   );

        float orientation = atan2(pointsShorted[5].y-
pointsShorted[4].y,pointsShorted[5].x-pointsShorted[4].x) * 57.2958;

        float Cz[4] = {0};
        for (int i=0; i<4; i++){
           Cz[i]=px2dist(Ixy[i], vertices, focalLength, cam2pixRatio);
        }
        float overallDistance;
        for (int i=0; i<4; i++){
           overallDistance=abs(overallDistance+Cz[i]);
        }
        overallDistance=overallDistance/4;
        overallDistance = -0.1603*overallDistance+0.2303 + overallDistance+2;
        projectionAngle pointAngle[4];
        for (int i = 0; i<4; i++){
           pointAngle[i].x_angle = abs(mapping(pointsShorted[i].x, 0, cam_width,
FOV/-2, FOV/2));
           pointAngle[i].y_angle = abs(mapping(pointsShorted[i].y, 0,
cam_height, FOV/-2, FOV/2));
        }

        float u =
Cz[0]*sin(deg2rad*(pointAngle[1].x_angle+pointAngle[0].x_angle)/2);
        float v =
Cz[2]*sin(deg2rad*(pointAngle[3].x_angle+pointAngle[2].x_angle)/2);
        float alpha = rad2deg*acos(abs(u+v)/vertices);
        cv::Point pt1 = pointsShorted[0];
        for (int i = 0; i<3; i++){
           cv::Point pt2 = pointsShorted[i+1];
           cv::line(frameCap, pt1, pt2, cv::Scalar(0,0,255),1,cv::LINE_8,0);
           pt1=pt2;
        }

        cv::circle(frameCap,pointsShorted[4],10,cv::Scalar(0,0,255),2);

        cv::circle(frameCap,pointsShorted[5],10,cv::Scalar(0,0,255),2);

        char send2raspberry [100];
```

```
            std::sprintf(send2raspberry, "$%.2f@%.2f~%d!%d#\n",
                        orientation, overallDistance, pointsShorted[5].x,
pointsShorted[5].y);
            serialPuts (fd, send2raspberry) ;
               std::cout<<send2raspberry<<'\n';
            if (recordmodeIsOn) logFile << millis() << '\t' << orientation <<'\t' <<
overallDistance <<'\t'<< pointsShorted[5].x <<'\t'<< pointsShorted[5].y <<'\n';
        }

        cv::circle(frameCap,cv::Point2d(cam_width/2,
cam_height/2),2,cv::Scalar(0,255,0),2);
        cv::circle(frameCap,cv::Point2d(cam_width/2,
cam_height/2),33,cv::Scalar(0,255,0),1);

        if (recordmodeIsOn) keluaranVideo.write(frameCap);
        cv::imshow("show", frameCap);

        char c = cv::waitKey(10);
        if (c=='q')break;
    }
    cap.release();
    return 0;
}
```

Halaman ini sengaja dikosongkan

BIOGRAFI PENULIS

Penulis bernama Muhammad Qomaruzzaman. Penulis dilahirkan di desa Bungah, Kabupaten Gresik. Pada masa kecilnya, penulis disekolahkan pada pendidikan berbasis pesantren di Yayasan Pondok Pesantren Qomaruddin. Demi memperluas wawasan, orang tua penulis merelakan anaknya untuk sekolah di SMA Negeri 3 Malang. Jauh dari kampung halaman, disana penulis belajar mandiri dan menghargai perbedaan. Selama masa kecilnya, penulis tidak pernah berkecimpung di dunia elektronika maupun komputer. Penulis menemukan ketertarikan terhadap dunia teknologi setelah dikenalkan oleh gurunya, yaitu guru fisika. Pada akhirnya ketertarikan itu mendorongnya untuk melanjutkan pendidikan tinggi di Institut Teknologi Sepuluh Nopember. Bidang yang ditempuh oleh penulis adalah Elektronika. Setelah lulus dengan gelar sarjana penulis semakin tertarik dengan luasnya ilmu dan kembali menempuh pendidikan master pada bidang yang sama.