TUGAS AKHIR - TE 141599

SISTEM PEMANDU PENDARATAN PADA BALON UDARA BERBASIS PENGOLAHAN CITRA DAN KENDALI PID

AGUNG ANDRI KURNIAWAN
NRP 2212 100 010

Dosen Pembimbing
Dr. Muhammad Rivai, S.T., M.T.
Fajar Budiman, S.T.,M.Sc.

JURUSAN TEKNIK ELEKTRO
Fakultas Teknologi Industri
Institut Teknologi Sepuluh Nopember
Surabaya 2016

FINAL PROJECT - TE 141599

GUIDED LANDING SYSTEM OF ROBOTIC BLIMP BASED ON IMAGE PROCESSING AND PID CONTROL

AGUNG ANDRI KURNIAWAN
NRP 2212 100 010

Supervisor
Dr. Muhammad Rivai, S.T., M.T.
Fajar Budiman, S.T.,M.Sc.

DEPARTMENT OF ELECTRICAL ENGINEERING
Faculty of Industrial Technology
Sepuluh Nopember Institute of Technology
Surabaya 2016

SISTEM PEMANDU PENDARATAN PADA BALON UDARA BERBASIS PENGOLAHAN CITRA DAN KENDALI PID

TUGAS AKHIR

Diajukan Guna Memenuhi Sebagian Persyaratan
Untuk Memperoleh Gelar Sarjana Teknik
Pada
Bidang Studi Elektronika
Jurusan Teknik Elektro
Institut Teknologi Sepuluh Nopember

Menyetujui :

Dosen Pembimbing I,	Dosen Pembimbing II,
Dr. Muhammad Rivai, S.T., M.T.	Fajar Budiman, ST.,M.Sc.
NIP. 196904261994031003	NIP. 198607072014041001

SURABAYA
JUNI, 2016

SISTEM PEMANDU PENDARATAN PADA BALON UDARA BERBASIS PENGOLAHAN CITRA DAN KENDALI PID

Nama : Agung Andri Kurniawan
Dosen Pembimbing I : Dr. Muhammad Rivai, S.T., M.T.
Dosen Pembimbing II : Fajar Budiman, S.T., M.Sc.

Abstrak:

Riset dan pengembangan studi *Unmanned Aerial Vehicle* (UAV) atau pesawat tanpa awak tengah berkembang pesat. Banyak perusahaan besar memfokuskan dalam kegiatan ini, mulai dari aplikasi untuk pengantaran barang, teknologi bertani, hingga untuk penanganan bencana alam. UAV menggunakan sistem Navigasi global positioning system (GPS) untuk memandu menuju lokasi tujuan. Navigasi dengan GPS mempunyai kelemahan yaitu rawan terhadap error hingga mencapai puluhan meter. Sehingga dapat menyebabkan masalah pada saat mendarat di lokasi tujuan. Maka dari itu, dirancanglah sistem pemandu pendaratan pada UAV dalam hal ini balon udara untuk membantu navigasi GPS dalam meningkatkan keakuratan pendaratan. Dengan bantuan pengolahan citra, sistem akan membantu mengenali pola dari landasan dengan metode *Hu Moments contour matching* yang tidak berubah secara skala, rotasi, dan translasi. Dari hasil percobaan sistem dengan metode *Hu Moments contour matching*, didapatkan tingkat keakurasian sebesar 94% pada perubahan skala secara ketinggian, 94% pada perubahan rotasi .

Kata Kunci : *UAV , Contour Matching, Pengolahan Citra*

GUIDED LANDING SYSTEM OF A ROBOTIC BLIMP BASED ON IMAGE PROCESSING AND PID CONTROL

Name : Agung Andri Kurniawan
Supervisor : Dr. Muhammad Rivai, S.T., M.T.
Co-Supervisor : Fajar Budiman, S.T., M.Sc.

Abstract:

Research and development of Unmanned Aerial Vehicle (UAV) has become popular study. Many of big companies have focused in this, such as the applications on delivery of goods, on farming in spreading seeds, and on area monitoring for search and rescue. Most of UAV are using Global Positioning System (GPS) as a navigation toward a target destination. However, GPS sometimes can experience a bad signal and have low accuracy up to several meters. This situation results a risk when the UAV cannot land properly at the safe area. Therefore, a guided landing system of UAV, especially for blimps, is designed. The system is used to assist the GPS to increase it's accuracy as the solution for the problem that mention above. This system could reduce the error from the GPS with the helps of camera. *Hu Moments Contour Matching* method that is invariant to scale, rotation, dan translation is used in this research. From the experimental result, the accuracy of landing pad detection using this method is 94% for the scale changing dan 94% for the landing pad rotating.

Keyword : *UAV , Contour Matching, Image Processing*

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

BAB I
PENDAHULUAN

1.1 LATAR BELAKANG

Penggunaan teknologi Unmanned Aerial Vehicle (UAV) atau Pesawat Tanpa Awak dewasa ini banyak berkembang pada negara maju. UAV adalah mesin terbang yang dapat dikendalikan secara jarak jauh oleh pilot maupun secara autopilot. Contoh dari UAV sendiri dapat berupa quadcopter, helicopter, dan balon udara. UAV sendiri banyak digunakan pada bidang militer dan sipil seperti pengintaian musuh, pengawasan suatu wilayah yang sulit dijangkau, monitoring kualitas udara suatu daerah, maupun eksplorasi gas dan minyak bumi[1]. Banyak dari aplikasi UAV diatas memerlukan fitur autopilot untuk melakukan monitoring dan pengawasan secara berkelanjutan.

Fitur autopilot pada UAV sangat berguna untuk mengatur rute pengambilan data, keberangkatan, dan pendaratan secara otomatis. Autopilot pada UAV mengandalkan Global Positioning System (GPS) sebagai acuan utama untuk penentuan rute yang akan dilalui [2]. Kekurangan dari sistem ini adalah keakuratan dan ketelitian pada GPS yang dapat ber-orde meter. Sehingga pada saat pendaratan UAV beresiko mendarat pada titik yang berbahaya dan dapat menimbulkan kerusakan pada UAV.

Oleh karena itu dalam tugas akhir ini telah dirancang sebuah sistem yang dapat membantu meningkatkan ketelitian dan keakuratan GPS untuk mengenali titik pendaratan yang diinginkan. Salah satunya adalah dengan menggunakan bantuan kamera melalui pengolahan citra untuk mengoptimalkan pendaratan pada UAV.

1.2 PERUMUSAN MASALAH

Permasalahan yang dibahas pada tugas akhir ini adalah :

1. Bagaimana penerapan pengolahan citra untuk mengenali titik landasan pendaratan.

2. Bagaimana cara untuk menjejak titik landasan pendaratan pada balon udara.
3. Bagaimana penerapan kendali pada balon udara untuk mengoptimalkan pendaratan.

1.3 BATASAN MASALAH

Batasan masalah pada tugas akhir ini adalah :

1. Landasan untuk mendarat sudah berada pada titik penglihatan kamera dari sistem.
2. Lokasi pelaksanaan mempunyai pencahayaan yang cukup dan statis.
3. Kontrol dibagi menjadi 2 bagian pada kamera pada axis x dan y dan sensor ultrasonic pada axis z.
4. Pengetesan balon udara diasumsikan dalam kondisi tenang tanpa gangguan dari angin

1.4 TUJUAN

Tujuan dari pembuatan tugas akhir ini adalah

1. Sistem mampu untuk mengenali dan menjejak titik landasan dengan pengolahan citra.
2. Sistem mampu untuk menghitung jarak antara UAV dengan daratan menggunakan sensor *ultrasonic*.
3. Sistem mampu untuk menerapkan kendali motor untuk pendaratan pada balon udara.

1.5 METODOLOGI

Metodologi yang dikerjakan pada tugas akhir ini adalah sebagai berikut:

1. Studi Literatur

Pada tahap studi literatur ini dilakukan pengumpulan dasar teori yang menunjang dalam penulisan tugas akhir. Dasar teori tersebut diambil dari artikel,jurnal,dan paper.

2. Perancangan perangkat keras

Pada tahap ini akan dirancang perangkat keras yang akan dipasang pada balon udara berupa raspberry pi untuk pengolahan data citra , sensor *ultrasonic* beserta arm stm32 untuk mengakuisisi data ketinggian balon udara . Dengan komunikasi serial antara raspberry pi dan arm stm32 data-data yang diterima akan diproses guna mengontrol arah dan kecepatan motor pada balon udara.

3. Perancangan Lunak

Pada tahap ini akan dibahas mengenai algoritma yang digunakan untuk membandingkan dan menentukan titik landasan serta menjejak titik landasan tersebut dengan seiring dengan pergerakan objek. Dengan data titik landasan tersebut kontrol motor balon udara akan bergerak menjejak sehingga balon udara akan tepat berada diatas titik landasan lalu balon udara akan melakukan pendaratan .

4. Perancangan Sistem

Setelah melakukan riset dari referensi yang berkaitan dengan pekerjaan tugas akhir ini, langkah berikutnya adalah melaksanakan perancangan sistem yang akan digunakan dalam implementasi perangkat keras, yaitu penggabungan antara perangkat keras dan perangkat lunak yang telah dirancang sebelumnya.

5. Pengujian Sistem

Pengujian alat dilakukan untuk menentukan keandalan dari sistem yang telah dirancang. Pengujian dilakukan untuk melihat apakah perangkat lunak dan perangkat keras dapat bekerja secara baik. Proses pengujian sistem dilakukan dengan sistem yang telah dirancang sebelumnya guna memperoleh sistem untuk mengontrol pendaratan pada balon udara. Pengujian awal akan dilakukan terhadap kamera dengan pengolahan citra untuk menentukan titik landasan. Setelah didapatkan data tersebut mikrokontroller akan menyesuaikan posisi untuk berada tepat di atas titik pendaratan. Data hasil dari kendali lalu akan dicatat untuk dianalisa.

6. **Penulisan Laporan Tugas Akhir**

Tahap penulisan laporan tugas akhir adalah tahapan terakhir dari proses pengerjaan tugas akhir ini. Laporan tugas akhir berisi seluruh hal yang berkaitan dengan tugas akhir yang telah dikerjakan yaitu meliputi pendahuluan, tinjauan pustaka dan teori penunjang, perancangan sistem, pengujian, dan penutup.

1.6 SISTEMATIKA PENULISAN

Laporan tugas akhir ini terdiri dari lima bab dengan sistematika penulisan sebagai berikut :

- Bab 1 : Pendahuluan
  Bab ini meliputi latar belakang, perumusan masalah, tujuan penelitian, batasan masalah, metodologi, sistematika penulisan,dan relevansi.
- Bab 2 : Tinjauan pustaka dan teori penunjang
  Bab ini menjelaskan tentang teori penunjang dan literatur yang dibutuhkan dalam pengerjakan tugas akhir ini.
- Bab 3 : Perancangan Sistem
  Bab ini menjelaskan tentang perangkat keras dan perangkat lunak yang digunakan pada tugas akhir ini.
- Bab 4 : Pengujian dan pembahasan sistem
  Bab ini berisi tentang pembahasan sistem yang digunakan dan hasil dari pengujian pada tugas akhir ini.
- Bab 5 : Penutup
  Bab ini berisi tentang kesimpulan dari hasil pengujian dan saran untuk pengembangan tugas akhir ini.

1.7 RELEVANSI

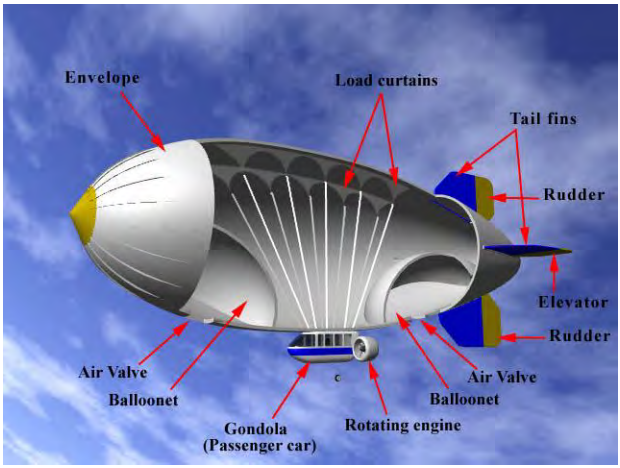
Tugas akhir ini memiliki relevansi dengan optimalisasi sistem autopilot pada *UAV*. Disamping itu hasil dari tugas akhir ini diharapkan juga dapat memberikan sumbangsih penelitian dalam bidang pengolahan untuk aplikasi *matching* dan *tracking*.

BAB II
TINJAUAN PUSTAKA DAN TEORI PENUNJANG

2.1 Balon Udara Blimps

Balon Udara Blimps adalah tipe kendaraan udara yang termasuk dalam Ligther-Than-Air (LTA)craft. Disebut Lighter-Than-Air dikarenakan blimps menggunakan komponen helium yang mempunyai massa jenis lebih ringan dari udara untuk membantu mengangkat pesawat [3]. Bagian bagian pada balon blimps dapat dilihat pada Gambar 2.1.

Balon Udara blimps menggunakan helium sebagai gaya angkat untuk terbang yang dikandung dalam envelope. Kontrol utama dari balon blimps pada dasarnya menggunakan motor yang berfungsi sebagai pendorong ke depan, elevate (mendorong keatas atau kebawah), dan rudder (untuk membelokkan kemudi)[4]. Penggunaan Balon Udara dalam UAV untuk aplikasi tertentu mempunyai beberapa kelebihan dimana tingkat keamanan, kemudahan penggunaan, dan kemampuan dalam terbang lebih baik daripada helikopter maupun quadcopter.

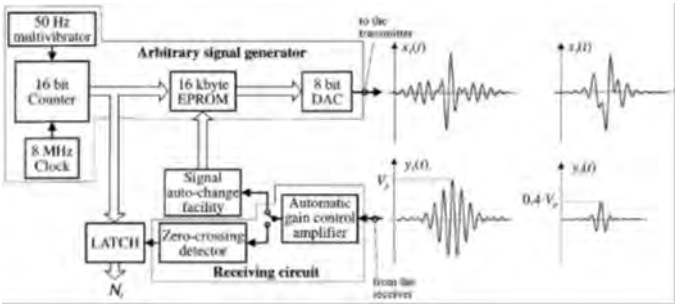


Gambar 2.1 Bagian-bagian pada balon blimps

2.2 Sensor *Ultrasonic*

Sensor *ultrasonic* adalah sensor yang bekerja dengan prinsip pantulan (echo) dengan menggunakan dua buah *piezoelectric* transduser, dengan satu transduser untuk memancarkan gelombang dan yang lainnya menerima gelombang hasil pantulan tersebut[5]. Sensor *ultrasonic* biasanya mempunyai 40 KHz tranduser resonansi *piezoelectric* yang digunakan untuk menciptakan gelombang *ultrasonic*.

Salah satu keuntungan dari pemakaian prinsip kerja sensor ultrasonic dengan metode time of flight atau waktu tempuh, yaitu waktu yang diperlukan dari transmitter menuju receiver kita dapat memperkirakan berapa jarak suatu benda terhadap sensor ultrasonic tersebut. Sehingga pada saat penerapannya sensor ultrasonic dapat membantu kamera untuk mengatur ketinggian balon pada tugas akhir ini.



Gambar 2.2 Sinyal Trigger dan Echo Ultrasonic

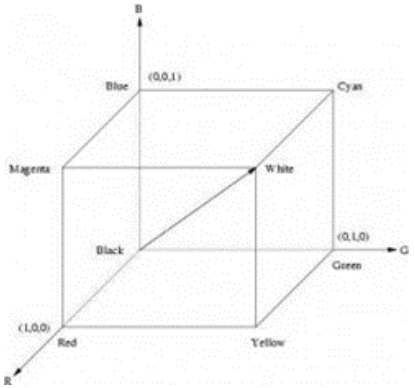
2.3 Citra Warna RGB

Citra dalam model Red-Green-Blue (RGB) terdiri dari tiga bidang citra yang saling lepas, masing-masing terdiri dari warna utama: merah, hijau dan biru. Suatu warna dispesifikasikan sebagai campuran sejumlah komponen warna utama.

RGB adalah suatu model warna yang terdiri dari merah, hijau, dan biru, digabungkan dalam membentuk suatu susunan warna yang luas. Setiap warna dasar, misalnya merah, dapat diberi rentang nilai. Untuk monitor komputer, nilai rentangnya paling kecil = 0 dan paling besar = 255. Pilihan skala 256 ini didasarkan pada cara mengungkap 8 digit

bilangan biner yang digunakan oleh mesin komputer. Dengan cara ini, akan diperoleh warna campuran sebanyak 256 x 256 x 256 = 1677726 jenis warna.

Sebuah jenis warna, dapat dibayangkan sebagai sebuah vektor di ruang dimensi 3 yang biasanya dipakai dalam matematika, koordinatnya dinyatakan dalam bentuk tiga bilangan, yaitu komponen-x, komponen-y dan komponen-z. Misalkan sebuah vektor dituliskan sebagai r = (x,y,z). Untuk warna, komponen-komponen tersebut digantikan oleh komponen Red, Green, Blue. Jadi, sebuah jenis warna dapat dituliskan sebagai berikut: warna = RGB(30, 75, 255). Putih = RGB (255,255,255), sedangkan untuk hitam= RGB(0,0,0).



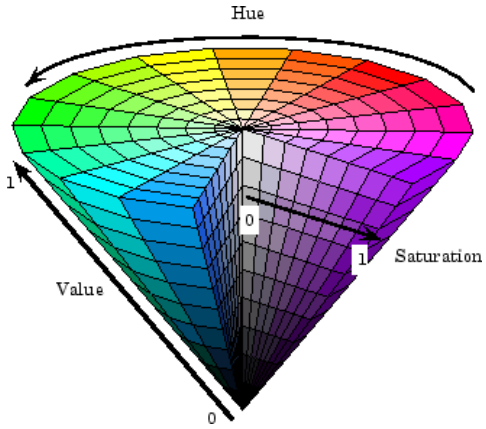
Gambar 2.3 Koordinat ruang warna pada RGB

2.4 Citra Warna HSV

Citra Warna HSV (Hue Saturation Value) menunjukkan ruang warna dalam bentuk tiga komponen utama, yaitu hue, saturation dan value (atau disebut juga brightness). Hue adalah sudut dari 0 sampai 360 derajat. Biasanya 0 adalah merah, 60 derajat adalah kuning, 120 derajat adalah hijau, 180 derajat adalah cyan, 240 derajat adalah biru dan 300 derajat adalah magenta.

Hue menunjukkan jenis warna (seperti merah, biru atau kuning) atau corak warna, yaitu tempat warna tersebut ditemukan dalam spektrum warna. Merah, kuning dan ungu adalah kata-kata yang menunjukkan hue.

Saturasi suatu warna adalah ukuran seberapa besar kemurnian dari warna tersebut. Sebagai contoh, suatu warna yang semuanya merah tanpa putih adalah saturasi penuh. Jika ditambahkan putih ke merah, hasilnya lebih berwarna-warni dan warna bergeser dari merah ke merah muda. Hue masih tetap merah tetapi nilai saturasinya berkurang. Saturasi biasanya bernilai 0 sampai 1 (atau 0% sampai 100%) dan menunjukkan nilai keabu-abuan warna dimana 0 menunjukkan abu-abu dan 1 menunjukkan warna primer murni. Komponen ketiga dari HSV adalah value atau disebut juga intensitas, yaitu ukuran seberapa besar kecerahan suatu warna atau seberapa besar cahaya datang dari suatu warna. Nilai value dari 0% sampai 100%.



Gambar 2.4 Model ruang warna pada HSV

2.5 Filter Warna HSV

Filter warna adalah metode yang digunakan untuk mengeliminasi warna lain selain dari warna yang diinginkan, misal jika kita ingin mengambil warna coklat saja dari berbagai warna dari suatu gambar, maka dengan metode Color Segmentation ini kita dapat mengatur agar hanya warna merah saja yang akan terdeteksi pada output.

Filter warna dengan citra HSV sendiri membutuhkan penkonversian gambar dari Red, Green, Blue (RGB) menjadi Hue, Saturation, Value (HSV), HSV sendiri adalah suatu ruang warna yang

direpresentasikan dari Hue (warna asli yang diekpresikan dari derajat 0 sampai 360), Saturation (warna asli yang dicampur dengan banyaknya kuantitas warna putih), dan Value (warna asli yang dicampur dengan banyaknya kuantitas warna hitam).

$R' = R/255$
$G' = G/255$
$B' = B/255$
$Cmax = \max(R', G', B')$
$Cmin = \min(R', G', B')$
$\Delta = Cmax - Cmin$

$$H = \begin{cases} 0° & \Delta = 0 \\ 60° \; x \left(\frac{G'-B'}{\Delta} mod 6\right), & Cmax = R' \\ 60° \; x \left(\frac{B'-R'}{\Delta} + 2\right), & Cmax = G' \\ 60° \; x \left(\frac{R'-G'}{\Delta} + 4\right), & Cmax = B' \end{cases} \quad (2.1)$$

$$S = \begin{cases} 0, Cmax = 0 \\ \frac{\Delta}{Cmax}, Cmax \neq 0 \end{cases} \quad (2.2)$$

$$V = Cmax \quad (2.3)$$

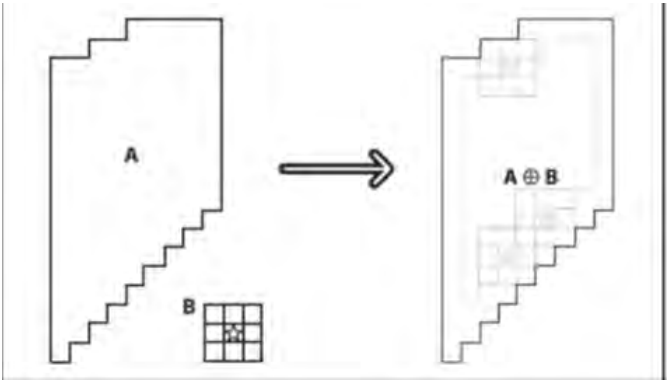
Filter HSV bekerja dengan cara mengatur nilai batas maksimum dan minimum pada Hue, Saturation, dan Value. Pengaturan nilai tersebut akan menfilter warna dari citra digital yang akan ditangkap.

Gambar 2.5 Contoh hasil dari filter warna menggunakan HSV

2.6 Morfologi Citra

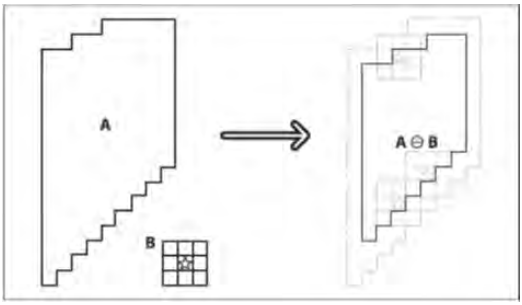
Morfologi citra (*Image Morphology*) adalah suatu metode untuk melakukan transformasi morfologi pada sebuah citra , metode paling dasar pada morfologi citra adalah dilation (dilasi), erotion (erosi), opening, dan closing. Pada penggunaannya morfologi citra digunakan untuk menghapus gangguan pada gambar (noise), mengisolasi suatu objek gambar, dan menggabung beberapa elemen yang terpisah pada gambar.

Dilasi adalah operasi morphologi yang akan menambahkan pixel pada batas antar objek dalam suatu citra digital. Atau secara rinci Dilasi merupakan suatu proses menambahkan piksel pada batasan dari objek dalam suatu image sehingga nantinya apabila dilakukan operasi ini maka image hasilnya lebih besar ukurannya dibandingkan dengan image aslinya.



Gambar 2.6 Proses operasi dilasi

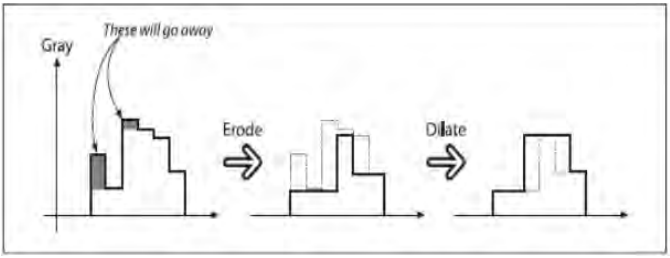
Erosi merupakan kebalikkan dari Dilasi. Proses ini akan membuat ukuran sebuah citra menjadi lebih kecil. Berbeda dengan dilatasi, apabila erosi dilakukan maka yang dikerjakan adalah memindahkan piksel pada batasan-batasan objek yang akan di erosi. Jumlah dari pikselyang ditambah atau dihilangkan bergantung pada ukuran dan bentuk dari structuring element yang digunakan untuk memproses image tersebut.



Gambar 2.7 Proses operasi erosi

Selain operasi dasar dilasi dan erosi dalam morfologi citra juga dikenal operasi kombinasi dari kedua operasi dasar dilasi dan erosi, yaitu operasi opening dan closing. Kombinasi dari kedua operasi dasar ini dapat membantu untuk menghilangkan noise dari citra digital dengan hasil yang lebih optimal.

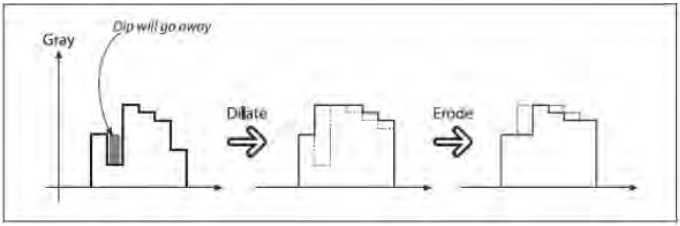
Opening merupakan kombinasi proses dimana suatu citra digital dikenai operasi erosi dilanjutkan dengan dilasi. Operasi opening pada citra mempunyai efek memperhalus batas-batas objek, memisahkan objek-objek yang sebelumnya bergandengan, dan menghilangkan objek-objek yang lebih kecil daripada ukuran structuring.



Gambar 2.8 Proses operasi opening

Closing merupakan kombinasi dimana suatu citra dikenai operasi dilasi dilanjutkan dengan erosi. Operasi closing juga cenderung akan memperhalus objek pada citra, namun dengan cara menyambung

pecahan-pecahan (fuses narrow breaks and thin gulf) dan menghilangkan lubang-lubang kecil pada objek.



Gambar 2.9 Proses operasi closing

2.7 Deteksi Kontur

Contour Detection (Deteksi Kontur) pada pendeteksian sebuah bentuk kontur luar maupun dalam dari input yang akan diproses yang berupa gambar biner, sebuah kontur terdiri dari beberapa titik koordinat (x,y) yang jumlahnya lebih dari 2 dan saling dihubungkan dengan suatu garis lurus antara satu titik dan titik lainnya[7].

Gambar 2.10 Kontur pada sebuah persegi

2.8 *Hu Invariant Moments*

Hu Invariant Moments pertama kali dipublikasikan oleh Hu pada tahun 1961 dan *Hu Invariant Moments* di reformula oleh Li . Nilai-nilai yang dihasilkan dari *Hu Invariant Moments* ini terdiri dari tujuh nilai yang menidentifikasikan ciri dari sebuah objek citra digital. Nilai-nilai tersebut bersifat independen terhadap translasi, rotasi dan penskalaan .Momen

yang mentransformasikan fungsi citra f(i,j) pada sistem diskrit dinyatakan dengan persamaan :

$$m_{pq} = \sum_{X=0}^{H-1} \sum_{y=0}^{w-1} x^p y^q f(x,y) \quad (2.4)$$

Dimana H dan W masing-masing merupakan tinggi dan lebar citra dan p = 0,1,2,… dan q = 0,1,2,… adalah integer. Selanjutnya, momen pusat untuk suatu citra dinyatakan dengan :

$$\mu_{pq} = \sum_{X=0}^{H-1} \sum_{y=0}^{w-1} (x - \bar{x})^p (y - \bar{y})^q f(x,y) \quad (2.5)$$

Dimana :

$$\bar{x} = \frac{m10}{m00} \; dan \; \bar{y} = \frac{m01}{m00} \quad (2.6)$$

Kemudian normalized central moments, didefinisikan sebagai berikut :

$$\eta_{pq} = \frac{\mu_{pq}}{\mu_{pq}{}^{y}} \; , y = \frac{p+q}{2} + 1 \quad (2.7)$$

Sehingga ketujuh nilai invariant moments dapat diturunkan dari moment kedua dan ketiga sebagai berikut.

$$\phi 1 = \eta_{20} + \eta_{02} \quad (2.8)$$

$$\phi 2 = (\eta_{20} - \eta_{02})^2 + 4\eta_{11}^2 \quad (2.9)$$

$$\phi 3 = (\eta_{30} - 3\eta_{12})^2 + (3\eta_{21} + \eta_{03})^2 \quad (2.10)$$

$$\phi 4 = (\eta_{30} + \eta_{12})^2 + (\eta_{21} + \eta_{03})^2 \quad (2.11)$$

$$\phi 5 = (\eta_{30} - 3\eta_{12})(\eta_{30} + \eta_{12})[(\eta_{30} + \eta_{12})^2 - (3\eta_{21} + \eta_{03})^2] + (3\eta_{21} - \eta_{03})(\eta_{21} + \eta_{03})[(3\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2] \quad (2.12)$$

$$\phi 6 = (\eta_{20} - 3\eta_{02})[(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2] + 4\eta_{11}(\eta_{30} - \eta_{12})(\eta_{21} + \eta_{03}) \quad (2.13)$$

$$\phi 7 = (3\eta_{21} - \eta_{03})(\eta_{30} + \eta_{12})[(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2] +$$

$$(\eta_{30} + \eta_{12})(\eta_{21} + \eta_{03})[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2] \quad (2.14)$$

2.9 Contour Matching

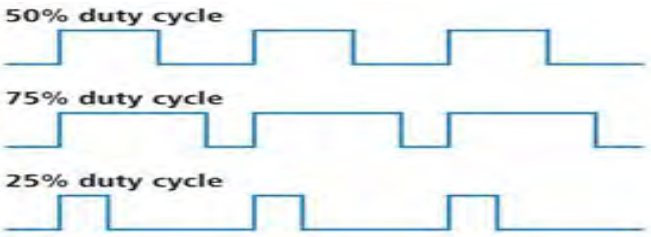
Metode contour matching dengan Hu Moments mampu mendeteksi kontur yang berbeda secara skala, rotasi, dan translasi dengan cara menjumlahkan semua nilai hasil pengurangan dari moments target dan moments yang akan dideteksi[8]. Dengan mengurangkan ketujuh nilai hu moments pada dua gambar yang identik nilai dari moments tersebut akan bernilai mendekati nol.

$$I_1(A, B) = \sum_{i=1\ldots7} \left| \frac{1}{m_i^A} - \frac{1}{m_i^B} \right| \quad (2.15)$$

2.10 *Pulse Width Modulation*

Pulse Width Modulation (PWM) adalah suatu sinyal yang dikirim dengan frekuensi tetap namun dapat memiliki panjang pulsa yang berbeda-beda dalam setiap periodenya. Perbedaan ini biasanya disebut dengan *Duty Cycle (*DC*)*, yaitu perbandingan lama pulsa dengan keseluruhan periode sinyal. *Pulse Width Modulation* (PWM) adalah istilah yang biasanya disebut sebagai sinyal keluaran (output) analog . Pada tugas akhir ini sinyal *Pulse Width Modulation* (PWM) akan digunakan untuk mengatur kecepatan motor pada balon udara.

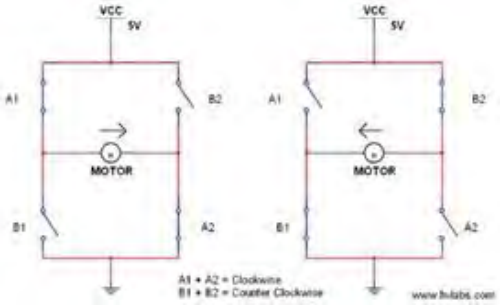
Pengaturan kecepatan motor dapat dilakukan dengan cara mengubah besaran nilai lebar pulsa *duty cycle* (DC). Pulsa yang yang berubah ubah *duty cycle*-nya inilah yang akan menentukan kecepatan motor. Besarnya amplitudo dan frekuensi pulsa adalah konstan, sedangkan besarnya *duty cycle* berubah-ubah sesuai dengan kecepatan yang diinginkan, semakin besar *duty cycle* yang dihasilkan maka semakin cepat pula kecepatan motor yang terjadi, dan sebaliknya semakin kecil *duty cycle* yang dihasilkan maka semakin pelan pula kecepatan motor yang terjadi.



Gambar 2.11 *Duty cycle* pada PWM

2.11 Driver Motor

Driver motor yang biasa digunakan adalah konfigurasi H-Bridge dimana arah pergerakan motor dapat dikendalikan sesuai dengan arah arum jam atau berlawanan dengan jarum jam. Pada prinsipnya, rangkaian ini terdiri dari empat buah saklar yang disusun sedemikian rupa sehingga motor DC dapat dialiri arus dengan arah yang berkebalikan.. Berikut gambar untuk menjelaskan prinsip tersebut:



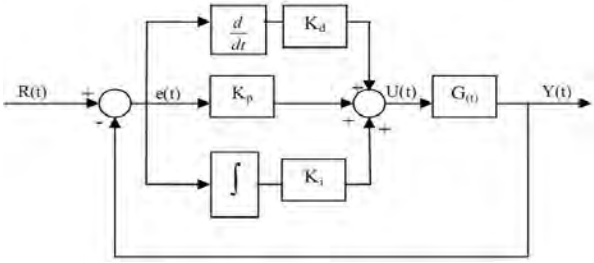
Gambar 2.12 Konfigurasi H-Bridge

Prinsip kerja dari rangkaian H-bridge adalah mengatur arah alir arus pada motor dengan cara membuka atau menutup switch yang ada sehingga motor dapat diatur untuk bergerak searah jarum jam atau berlawanan dengan jarum jam. Ketika A1 dan A2 tertutup sedangkan B2 dan B1 terbuka maka sisi kiri dari motor akan tersambung dengan VCC dan sisi sebelah kanan motor akan terhubung dengan ground sehingga

kondisi ini akan membuat arah pergerakan motor kekanan atau searah dengan jarum jam. Begitupun juga dengan kondisi sebaliknya, pada saat B2 dan B1 tertutup sedangkan A1 dan A2 terbuka maka arah alir arus akan bergerak dari sisi kanan motor menuju ke sisi kiri motor karena sisi kanan motor terhubung dengan VCC sedangkan sisi kiri motor terhubung dengan ground sehingga motor akan bergerak kearah kiri atau berlawanan dengan jarum jam. Pada penerapannya, switch ini biasanya diganti dengan saklar otomatis, salah satunya adalah dengan komponen transistor dimana untuk mengalirkan arus dari sisi kolektor kesisi emitor (short), kita hanya perlu memberikan arus forward pada basis, sehingga fungsi transistor tersebut akan berperan sebagai saklar.

2.12 Kendali PID

Penggunaan kendali PID pada industri penerbangan sudah banyak digunakan dengan berbagai macam jenis mulai dari algoritma yang simpel, mempunyai presisi tinggi, dengan struktur yang kuat, dan mempunyai tingkat keandalan yang baik[9]. Kendali PID merupakan kontroler untuk menentukan presisi suatu sistem instrumentasi dengan karakteristik adanya umpan balik pada sistem tesebut. Komponen kontrol PID ini terdiri dari tiga jenis yaitu Proportional, Integratif dan Derivatif. Ketiganya dapat dipakai bersamaan maupun sendiri-sendiri tergantung dari respon yang kita inginkan terhadap suatu plant. PID Blok Diagram dapat dilihat pada gambar dibawah :



Gambar 2.13 Diagram Blok PID

persmaan pengontol PID datas dapat juga dituliskan sebagai berikut :

$$y(t) = K_P\, e(t) + K_i \int_0^t e(t)dt + K_d \left(\frac{de(t)}{dt}\right) \quad (2.16)$$

dengan :

$$K_i = \frac{1}{T_i} \text{ x } K_p \text{ dan } K_d = K_p \text{ x } T_d \quad (2.17)$$

Komponen kontrol PID ini terdiri dari tiga jenis yaitu Proportional, Integratif dan Derivatif. Ketiganya dapat dipakai bersamaan maupun sendiri-sendiri tergantung dari respon yang kita inginkan terhadap suatu plant. Untuk lebih memaksimalkan kerja pengontrol diperlukan nilai batas minimum dan maksimum yang akan membatasi nilai *Manipulated Variable* yang dihasilkan.

1. Kontrol Proporsional

Kontrol P jika G(s) = kp, dengan k adalah konstanta. Jika u = G(s) • e maka u = Kp • e dengan Kp adalah Konstanta Proporsional. Kp berlaku sebagai Gain (penguat) saja tanpa memberikan efek dinamik kepada kinerja kontroler. Penggunaan kontrol P memiliki berbagai keterbatasan karena sifat kontrol yang tidak dinamik ini. Walaupun demikian dalam aplikasi-aplikasi dasar yang sederhana kontrol P ini cukup mampu untuk memperbaiki respon transien khususnya rise time dan settling time. Pengontrol proporsional memiliki keluaran yang sebanding/proporsional dengan besarnya sinyal kesalahan (selisih antara besaran yang diinginkan dengan harga aktualnya).

Ciri-ciri pengontrol proporsional :

1. Jika nilai Kp kecil, pengontrol proporsional hanya mampu melakukan koreksi kesalahan yang kecil, sehingga akan menghasilkan respon sistem yang lambat (menambah rise time).
2. Jika nilai Kp dinaikkan, respon/tanggapan sistem akan semakin cepat mencapai keadaan mantapnya (mengurangi rise time).

3. Namun jika nilai Kp diperbesar sehingga mencapai harga yang berlebihan, akan mengakibatkan sistem bekerja tidak stabil atau respon sistem akan berosilasi.
4. Nilai Kp dapat diset sedemikian sehingga mengurangi steady state error, tetapi tidak menghilangkannya.

2. Kontrol Integratif

Pengontrol Integral berfungsi menghasilkan respon sistem yang memiliki kesalahan keadaan mantap nol (Error Steady State = 0). Jika sebuah pengontrol tidak memiliki unsur integrator, pengontrol proporsional tidak mampu menjamin keluaran sistem dengan kesalahan keadaan mantapnya nol.

Jika G(s) adalah kontrol I maka u dapat dinyatakan sebagai u(t)=[integral e(t)dT]Ki dengan Ki adalah konstanta Integral, dan dari persamaan di atas, G(s) dapat dinyatakan sebagai u=Kd.[delta e/delta t] Jika e(T) mendekati konstan (bukan nol) maka u(t) akan menjadi sangat besar sehingga diharapkan dapat memperbaiki error. Jika e(T) mendekati nol maka efek kontrol I ini semakin kecil. Kontrol I dapat memperbaiki sekaligus menghilangkan respon steady-state, namun pemilihan Ki yang tidak tepat dapat menyebabkan respon transien yang tinggi sehingga dapat menyebabkan ketidakstabilan sistem. Pemilihan Ki yang sangat tinggi justru dapat menyebabkan output berosilasi karena menambah orde system

Keluaran pengontrol ini merupakan hasil penjumlahan yang terus menerus dari perubahan masukannya. Jika sinyal kesalahan tidak mengalami perubahan, maka keluaran akan menjaga keadaan seperti sebelum terjadinya perubahan masukan. Sinyal keluaran pengontrol integral merupakan luas bidang yang dibentuk oleh kurva kesalahan / error.

Ciri-ciri pengontrol integral :

1. Keluaran pengontrol integral membutuhkan selang waktu tertentu, sehingga pengontrol integral cenderung memperlambat respon.

2. Ketika sinyal kesalahan berharga nol, keluaran pengontrol akan bertahan pada nilai sebelumnya.
3. Jika sinyal kesalahan tidak berharga nol, keluaran akan menunjukkan kenaikan atau penurunan yang dipengaruhi oleh besarnya sinyal kesalahan dan nilai Ki.
4. Konstanta integral Ki yang berharga besar akan mempercepat hilangnya offset. Tetapi semakin besar nilai konstanta Ki akan mengakibatkan peningkatan osilasi dari sinyal keluaran pengontrol

3. Kontrol Derivatif

Keluaran pengontrol diferensial memiliki sifat seperti halnya suatu operasi derivatif. Perubahan yang mendadak pada masukan pengontrol akan mengakibatkan perubahan yang sangat besar dan cepat. Ketika masukannya tidak mengalami perubahan, keluaran pengontrol juga tidak mengalami perubahan, sedangkan apabila sinyal masukan berubah mendadak dan menaik (berbentuk fungsi *step*), keluaran menghasilkan sinyal berbentuk impuls. Jika sinyal masukan berubah naik secara perlahan (fungsi *ramp*), keluarannya justru merupakan fungsi step yang besar magnitudenya sangat dipengaruhi oleh kecepatan naik dari fungsi *ramp* dan factor konstanta Kd.

Sinyal kontrol u yang dihasilkan oleh kontrol D dapat dinyatakan sebagai G(s)=s.Kd Dari persamaan di atas, nampak bahwa sifat dari kontrol D ini dalam konteks “kecepatan” atau rate dari error. Dengan sifat ini ia dapat digunakan untuk memperbaiki respon transien dengan memprediksi error yang akan terjadi. Kontrol Derivative hanya berubah saat ada perubahan error sehingga saat error statis kontrol ini tidak akan bereaksi, hal ini pula yang menyebabkan kontroler Derivative tidak dapat dipakai sendiri

Ciri-ciri pengontrol derivatif :

1. Pengontrol tidak dapat menghasilkan keluaran jika tidak ada perubahan pada masukannya (berupa perubahan sinyal kesalahan)

2. Jika sinyal kesalahan berubah terhadap waktu, maka keluaran yang dihasilkan pengontrol tergantung pada nilai Kd dan laju perubahan sinyal kesalahan.
3. Pengontrol diferensial mempunyai suatu karakter untuk mendahului, sehingga pengontrol ini dapat menghasilkan koreksi yang signifikan sebelum pembangkit kesalahan menjadi sangat besar. Jadi pengontrol diferensial dapat mengantisipasi pembangkit kesalahan, memberikan aksi yang bersifat korektif dan cenderung meningkatkan stabilitas sistem.
4. Dengan meningkatkan nilai Kd, dapat meningkatkan stabilitas sistem dan mengurangi *overshoot*.

Berdasarkan karakteristik pengontrol ini, pengontrol diferensial umumnya dipakai untuk mempercepat respon awal suatu sistem, tetapi tidak memperkecil kesalahan pada keadaan tunaknya. Kerja pengontrol diferensial hanyalah efektif pada lingkup yang sempit, yaitu pada periode peralihan. Oleh sebab itu pengontrol diferensial tidak pernah digunakan tanpa ada kontroler lainnya.

2.13 Metode Tuning PID *Ziegler-nichols*

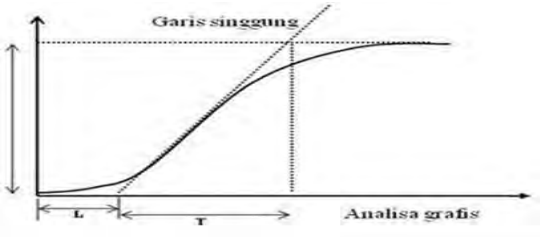
Penentuan parameter kontroler PID adalah hal yang sangat penting untuk menentukan bagaimana performa dari kontrol plant tersebut pada suatu sistem , Hal ini disebut juga dengan tuning kontroler. Terkadang pemodelan matematis suatu plant susah untuk dilakukan. Jika hal ini terjadi maka perancangan kontroler PID secara analitis tidak mungkin dilakukan sehingga perancangan kontroler PID harus dilakukan secara eksperimental.

2.13.1 Manual tuning

Manual tuning dapat dilakukan dengan mengeset Ki an Kd menjadi nol. Kemudian tingkatkan nilai Kp perlahan sampai mencapai osilasi lalu set kembali sekitar menjadi setengahnya. Tingkatkan Ki sampai mencapai hasil yang lebih baik. Akan tetapi, nilai Ki yang terlalu besar akan menyebabkan sistem tidak stabil. Terakhir, tingkatkan nilai Kd. Akan tetapi terlalu besar nilai Kd akan menyebabkan respon yang berlebihan dan overshoot.

2.13.2 Ziegler-nichols open loop tuning method

Aturan perpotongan garis lurus terjadi pada kondisi linier dari kurva S repon sistem. Ketepatan dalam pengambilan perpotongan ini sangatlah penting karena menentukan parameter T dan L yang menjadi acuan dari kontroler.



Gambar 2.14 Analisa grafis pada metode open loop

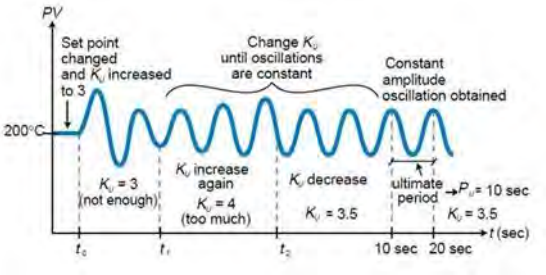
Dengan mengetahui grafis pada respon open loop kita dapat menentukan nilai L dan T dan dengan menggunakan formula yang telah dirumuskan oleh ziegler-nichols kita dapat menentukan perkiraan nilai Kp, Ki, dan Kd yang optimal untuk plant yang akan kita kontrol.

Tabel 2.1 Formula ziegler nichols pada open loop

Tipe Pengendali	Kp	Ki	Kd
P	$\frac{T}{L}$	∞	0
PI	$\frac{0.9T}{L}$	$0.27\frac{T}{L^2}$	0
PID	$\frac{1.2\,T}{L}$	$0.6\frac{T}{L^2}$	0.6T

2.13.3 Ziegler-nichols closed loop tuning method

Metode kedua yang dirumuskan oleh ziegler nichols adalah metode untuk plant yang menggunakan kontrol PID closed loop. Metode ini bekerja dengan menaikkan harga Ku hingga plant berosilasi secara penuh dan konstan pada grafis respon kontrol.



Gambar 2.15 Analisa grafis pada metode closed loop

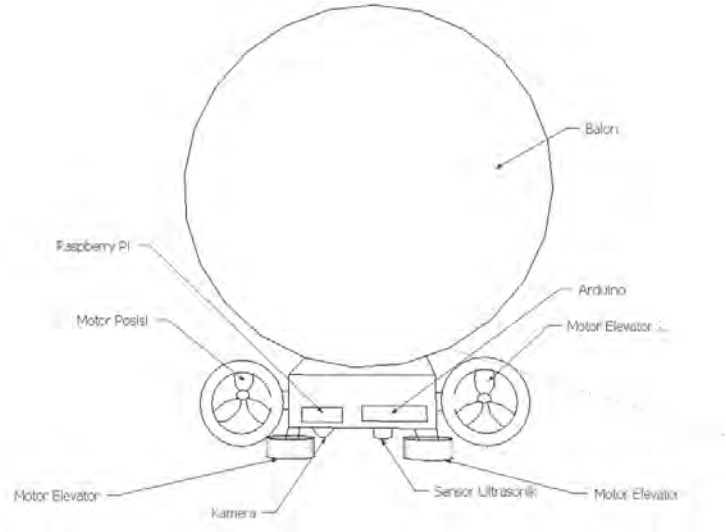
Setelah nilai Ku diketahui nilai konstanta Kp, Ki, dan Kd dapat diketahui melalui tabel yang telah dirumuskan oleh ziegler-nichols seperti berikut:

Tabel 2.2 Formula ziegler nichols pada closed loop

	K_c	T_I	T_D
P	$K_u/2$		
PI	$K_u/2.2$	$P_u/1.2$	
PID	$K_u/1.7$	$P_u/2$	$P_u/8$

BAB III
PERANCANGAN SISTEM

Pada bab ini akan dibahas perancangan keseluruhan sistem dimulai dari penjelasan bagaimana pengenalan target landasan sampai pada perancangan perangkat lunak dan perangkat keras dari sistem pemandu pendaratan berbasis pada balon udara ini. Pada tugas akhir ini memanfaatkan teknologi pengolah citra untuk melakukan pengenalan target landasan dan penjejakan target landasan. Sistem akan mencari target landasan berdasarkan citra yang diambil dari kamera dan mengolah citra tersebut sehingga didapatkan data untuk perencanaan pergerakan balon udara menuju target landasan.



Gambar 3.1 Rancangan hardware pada balon udara

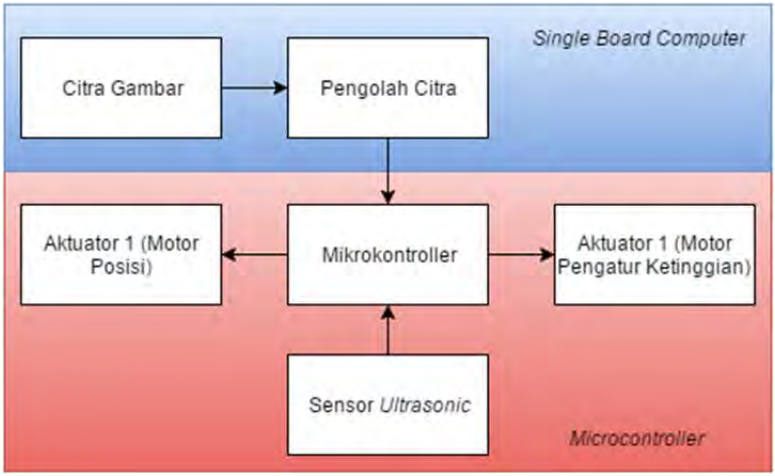
Balon udara yang sebelumnya sudah diatur menuju tempat untuk pendaratan melalui GPS akan mempertahankan ketinggian dengan kontrol PID ketinggian untuk mendapatkan penglihatan ke daratan yang luas. Setelah balon udara menemukan target landasan, motor akan

bergerak untuk mengatur posisi balon udara supaya simetris dengan landasan. Jika posisi balon sudah tepat berada diatas titik landasan balon udara akan melakukan pendaratan.

3.1 Diagram Blok dan Flowchart Sistem

Perangkat keras untuk pemrosesan data pada sistem ini adalah raspberry pi dan arduino, citra gambar yang ditangkap oleh kamera akan diproses dengan *contour matching* pada raspberry pi dan output dari pemrosesan citra gambar berupa koordinat lokasi landasan. Setelah koordinat ditemukan akan ditentukan pergerakan dan kecepatan motor yang sesuai untuk menuju koordinat titik pendaratan tersebut.

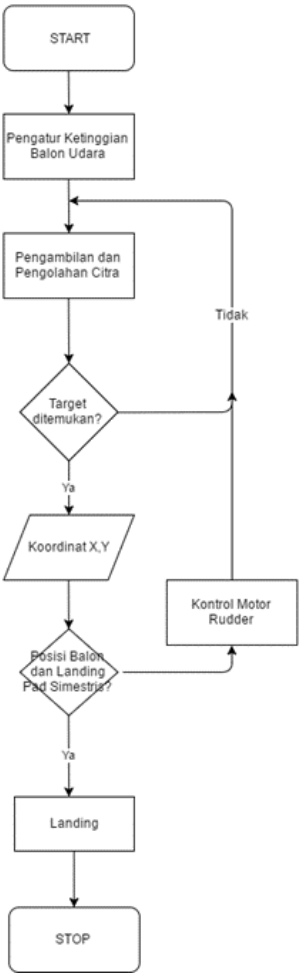
Mikrokontroller yang digunakan untuk mengolah kecepatan, arah, dan mengukur ketinggian balon udara adalah arduino. Keluaran sinyal PWM pada arduino digunakan untuk mengatur kecepatan motor dan sensor *ultrasonic* berfungsi untuk mengetahui ketinggian balon udara. Berikut adalah diagram blok sistem yang digunakan.



Gambar 3.2 Diagram blok koneksi hardware pada balon udara

Berikut adalah flowchart sistem yang menjelaskan bagaimana sistem akan berjalan secara keseluruhan mulai dari pengaturan ketinggian

untuk mendapatkan sudut pandang yang luas dari atas landasan, pergerakan motor hingga kontrol untuk mengemudikan balon menuju tepat diatas landasan.



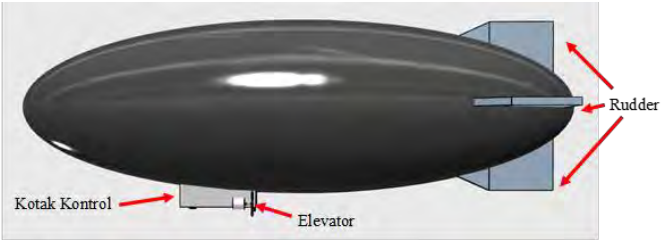
Gambar 3.3 Flowchart pergerakan balon udara untuk pendaratan

3.2 Perancangan Perangkat Keras

Pada bagian ini dibahas tentang perangkat keras yang digunakan dalam tugas akhir ini. Perangkat keras yang digunakan meliputi : balon blimps, power supply, sensor *ultrasonic*, raspberry pi, modul kamera, *h-bridge*, dan Arduino pro mini.

3.2.1 Balon Blimps

Balon udara dirancang untuk memiliki 4 buah motor, yaitu 2 motor rudder untuk mengatur posisi dan 2 motor elevator untuk mengatur ketinggian.



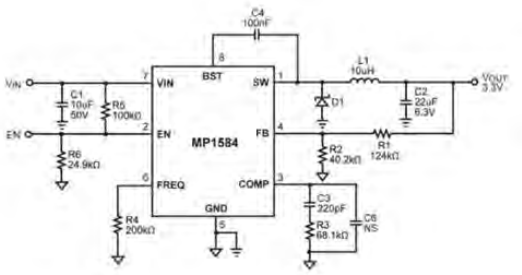
Gambar 3.4 Peletakan posisi kendali pada balon udara

Posisi peletakan kamera berada di bagian bawah dari titik masa balon udara, hal ini dilakukan untuk mempermudah proses kalkulasi dari penjejakan pipa. Dan posisi peletakan sensor ultrasonic berada disebelah kamera. Posisi sensor Ultrasonic harus sejajar untuk mengetahui ketinggian dari balon udara.

3.2.2 Power Supply

Power Supply adalah perangkat elektronika yang mensuplai sumber listrik ke perangkat elektronika lainnya. Dalam suatu power supply terdapat sebuah regulator tegangan dimana digunakan untuk menurunkan tegangan dari satu level tertentu ke level yang diinginkan. Dalam tugas akhir ini menggunakan sebuah MP1584 yang merupakan sebuah buck converter yang meregulaasi tegangan dari rentang 4.5-28Vinput menjadi 0,8-18Voutput dan mampu menyuplai suatu beban sampai batas arus 3A. Pada Tugas Akhir ini diperlukan duah buah besaran

tegangan yang dibutuhkan ialah 3,3V dan 5V. Untuk mikrokontroller dan sensor.



Gambar 3.5 MP1528 Buck Converter

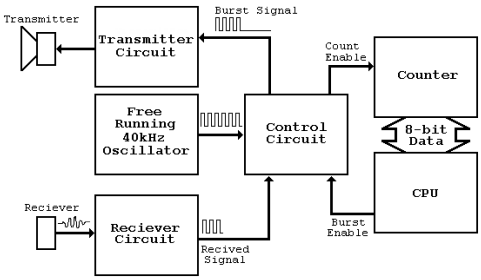
Untuk mendapatkan keluaran tegangan yang dinginkan menggunakan perbandingan besar R yang berbeda-beda. Berikut perhitungan rumus untuk menentukan besar perbandingan niali resistor:

$$Vout = \frac{Vfb(R1+R2)}{R2} \quad (3.1)$$

Pada beberapa aplikasi nilai R2 dijadikan 40,2 KOhm sehingga yang perlu dicari adalah besar nilai R1. Untuk nilai tegangan 3,3V nilai R1 adalah 124 KOhm da nilai R2 adalah 40,2 KOhm.

3.2.3 Sensor *Ultrasonic*

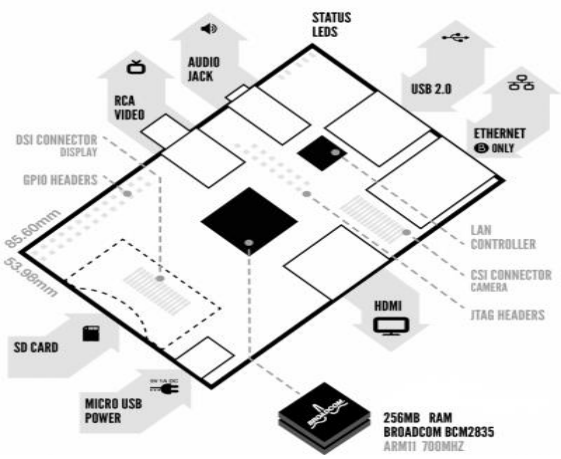
Sensor *ultrasonic* terdiri dari dua buah transduser piezoelektrik yang mampu untuk memancarkan dan menerima sinyal *ultrasonic*. Sensor *ultrasonic* sendiri digunakan untuk mengetahui ketinggian balon udara dengan cara transduser pada sensor *ultrasonic* berupa piezoelektrik yang bekerja pada frekuensi 40 KHz memancarkan sinyal *ultrasonic* sesaat dan menghasilkan pulsa output yang sesuai dengan waktu pantul sinyal *ultrasonic* sesaat kembali menuju sensor. Dengan mengukur lebar pulsa pantulan tersebut jarak target didepan sensor dapat diketahui.



Gambar 3.6 Blok diagram sensor *ultrasonic*

3.2.4 Raspberry Pi

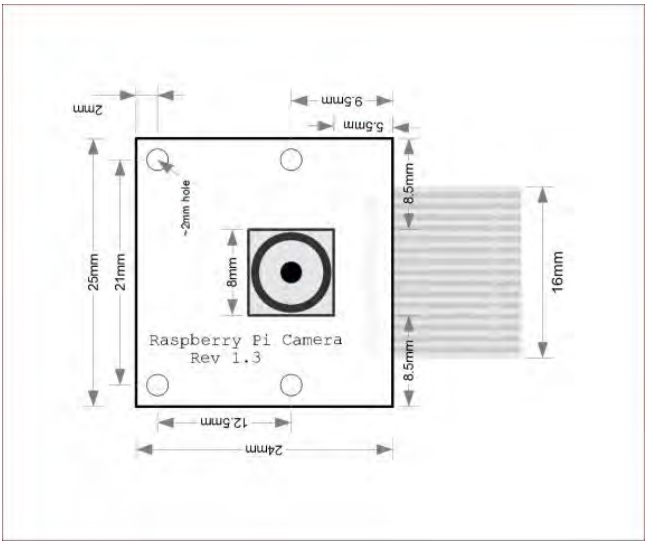
Raspberry Pi digunakan sebagai pengolah citra utama, pada konektor CSI Raspberry akan dipasang sebuah kamera khusus untuk Raspberry Pi. Raspberry Pi memiliki input tegangan 5V 2A. Agar dapat memenuhi suplai tegangan, maka sebuah power bank 4400 MAh dipasang pada balon udara.



Gambar 3.7 Blok diagram hardware pada Raspberry Pi

3.2.5 Modul Kamera Raspberry Pi

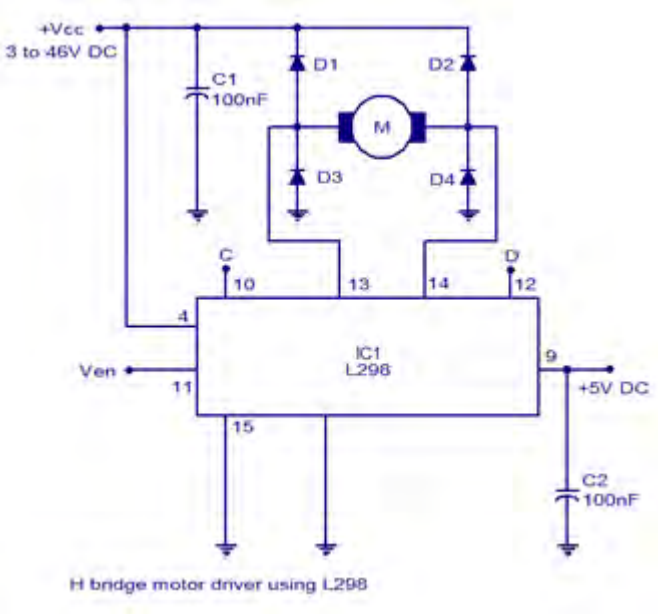
Modul kamera yang digunakan merupaka modul yang khusus dibuat untuk Raspberry Pi, konektor yang digunakan adalah CSI-2 (Camera Serial Interface Gen-2). Modul ini dipasang di bagian bawah titik masa balon untuk mempermudah kalkulasi pergerakan.



Gambar 3.8 Modul kamera Raspberry Pi

3.2.6 H-Bridge L298

H-Bridge L298 adalah *Integrated Circuit* (IC) yang dapat digunakan sebagai driver motor DC. IC ini menggunakan prinsip kerja H-Bridge. Tiap H-Bridge dikontrol menggunakan level tegangan *Transistor-transistor-logic* (TTL) yang berasal dari output mikrokontroler. L298 dapat mengontrol 2 buah motor DC. Tegangan yang dapat digunakan untuk mengendalikan robot bisa mencapai tegangan 46 VDC dan arus 2 A untuk setiap kanalnya. Bentuk IC L298 yang digunakan sebagai motor driver dapat dilihat pada gambar 3.9.

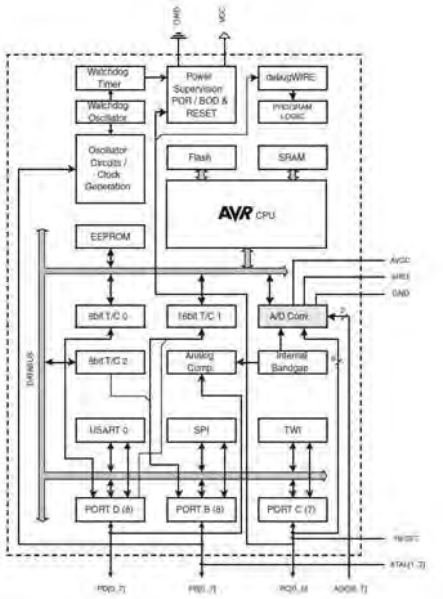


Gambar 3.9 Dual H-Bridge L298

3.2.7 Arduino Pro Mini

Arduino Pro Mini merupakan sebuah board mikrokontroler berbasis ATMega328. Modul ini memiliki 14 digital input/output dmana 6 digunakan untuk PWM output dan 6 digunakan sebagai analog input, 1 port serial, 8 MHz osilator Kristal, koneksi USB, power jack, ICISP Header, dan tombol reset. Memiliki flash memory sebesar 32KB sangat cukup untuk menampung program yang banyak.

Arduino Pro Mini memerlukan flash program external karena di dalam chip mikrokontroler Arduino tidak mempunyai bootloader untuk proses upload. Untuk memprogram Arduino Pro Mini diperlukan FTDI chip.



Gambar 3.10 Blok diagram arduino

3.3 Perancangan Perangkat Lunak

Pada bagian ini dibahas tentang perangkat lunak yang digunakan dalam tugas akhir ini. Perangkat keras yang digunakan meliputi: pengakuisisian data *ultrasonic*, proses pengolahan citra, dan kendali PID.

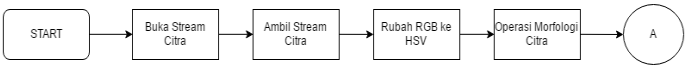
3.3.1 Akuisisi Data Sensor Ultrasonic

Proses akuisisi data sensor ultrasonic dilakukan dengan menghubungkan pin trigger dan echo pada kaki digital pada arduino, pada pin trigger akan dikeluarkan sinyal pulse high dan pada pin trigger akan mengeluarkan sinyal yang nantinya akan berguna untuk mengukur ketinggian. Setelah sinyal keluar dan menabrak sebuah objek, sinyal akan memantul kembali menuju sensor ultrasonic dan pada pin echo akan terdeteksi sinyal high yang berasal dari pantulan. Penghitungan ketinggian dilakukan dengan menghitung waktu jeda antara pengiriman dan penerimaan sensor. Dengan acuan kecepatan suara 340m/s

ketinggian dapat diperoleh dengan mengalikan kecepatan suara dengan waktu jeda lalu dibagi menjadi dua. Dikarenakan jarak yang ditempuh sinyal tersebut merupakan dua kali jarak ketinggian saat dipancarkan dan saat memantul.

3.3.2 Proses Preprocessing Citra

Proses preprocessing citra berfungsi untuk mendapatkan hasil yang maksimal saat dilakukannya pengolahan citra, dengan preprocessing mengolah data citra yang mentah menjadi data citra yang siap untuk diproses selanjutnya dengan melewati proses pengubahan ruang warna dari RGB ke HSV hingga proses morfologi citra. Kedua proses tersebut sangat penting untuk menfilter warna dan mengurangi noise pada citra yang akan diproses, sehingga dapat meningkatkan keakurasian pada proses selanjutnya. Proses pertama yang dilakukan pada preprocessing ini adalah pengambilan citra. Pengambilan citra dilakukan oleh kamera, lalu citra digital tersebut akan diolah pada Raspberry Pi 3, bahasa yang dipakai adalah bahasa python dengan versi 3.4. Proses pengambilan citra dan filter warna dibantu dengan library OpenCV 3.0. Untuk mengambil citra langkah pertama yang harus dilakukan adalah membuka stream citra pada prosesnya secara threading agar tidak memblok sistem saat dilakukan akuisisi citra. Setelah stream dibuka data citra dapat diakuisisi untuk diambil dan diproses oleh sistem.



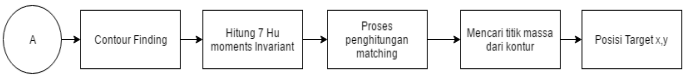
Gambar 3.11 Diagram Blok Proses Preprocessing Citra

Filter warna dilakukan dengan mengubah terlebih dahulu ruang warna RGB menjadi ruang warna HSV, dengan bantuan library pada OpenCV nilai Hue, Saturation, dan Value pada ruang warna HSV dapat dibatasi dengan perintah cv2.inrange .Perubahan dan pengaturan batasan bawah dan batasan atas ini berfungsi untuk membatasi warna yang masuk pada pengambilan citra, sehingga warna diluar target tidak akan tertangkap pada pengolahan citra.

Setelah hasil citra dari proses pengambilan citra dan filter warna didapatkan, data citra akan dimorfologikan terlebih dahulu. Proses morfologi berfungsi untuk mengurangi gangguan pada piksel yang tidak sengaja tertangkap kamera dan menutup lubang dari hasil filter warna yang tidak sempurna. Proses morfologi yang dilakukan adalah 3 kali operasi opening , 1 operasi closing, dan dilanjutkan dengan 1 kali operasi opening lagi.

3.3.3 Proses Pengenalan Landasan

Setelah data citra yang didapatkan melewati proses preprocessing, selanjutnya data yang telah didapatkan akan diproses untuk dapat mengenali landasan. Metode pengenalan landasan yang dilakukan adalah dengan menemukan kontur dari target dan kontur *template* yang digunakan untuk mengenali target. Setelah kontur luar dari tiap data citra didapatkan, ketujuh nilai Hu Moments dari kedua target tersebut akan dihitung menggunakan library OpenCV, ketujuh nilai moments tersebut akan terus dihitung dan dibandingkan. Untuk metode perbandingan yang dilakukan dengan Hu Moments ini adalah dengan terus mengurangi ketujuh nilai Hu Moments dari kedua gambar ini secara terus menerus. Semakin kecil nilai hasil yang didapatkan pada pengurangan nilai ini semakin besar tingkat kemiripan pada kedua gambar tersebut.



Gambar 3.12 Diagram Blok Sistem Pengenalan Landasan

Setelah didapatkan matching, dicari posisi nilai tengah dari kontur yang ditemukan dengan mengubah kontur menjadi *moments* agar dapat diakses secara *array*. Nilai x dan y terletak pada kombinasi posisi array m10, m01 dan m00. Untuk mendapatkan nilai x dan y yang benar digunakan rumus:

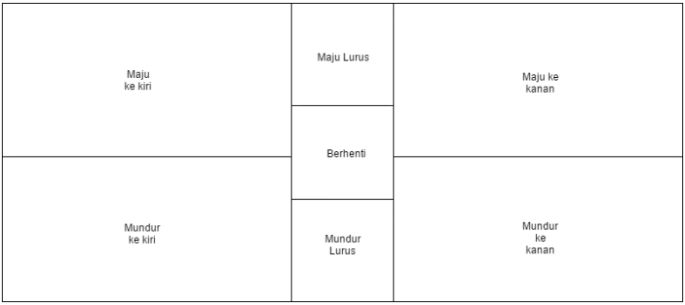
$$x = \frac{m10}{m00} dan\ y = \frac{m01}{m00} \tag{3.2}$$

Setelah didapatkan nilai x dan y pada setiap kotak ditemukan, maka sudut dari landasan yang ingin dijejak dapat dikalkulasikan dengan rumus:

$$\deg = \tan^{-1} \frac{y1-y2}{x1-x2} * \left(\frac{180}{\pi}\right) \quad (3.3)$$

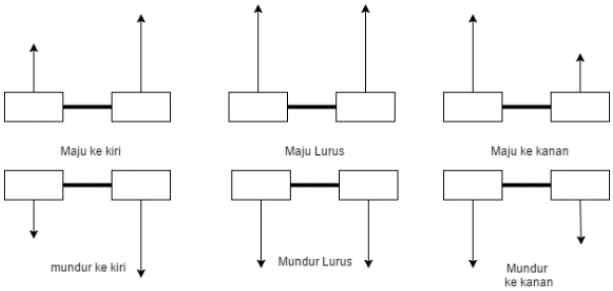
3.3.4 Perencanaan Pergerakan

Dalam pergerakan balon udara untuk menyesuaikan posisi agar berhenti tepat diatas landasan, pergerakan untuk menuju titik landasan pada balon udara dibagi menjadi tujuh region dalam frame tangkapan kamera. Balon udara akan bergerak sesuai dengan letak titik landasan yang terdeteksi pada frame tangkapan kamera. Kecepatan motor akan bergerak sesuai dengan error yang akan dimasukkan ke dalam kendali PID. Balon udara akan terus bergerak hingga posisi letak landasan berada pada region berhenti. Setelah region telah sampai pada berhenti maka kendali PID untuk motor pengatur ketinggian akan berhenti dan balon udara akan melakukan pendaratan.



Gambar 3.13 tujuh region gerak pada frame kamera.

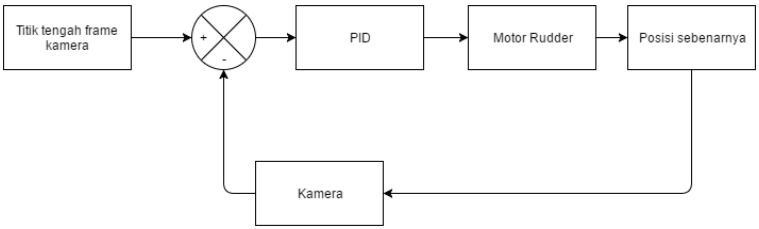
Pergerakan balon udara sendiri menggunakan motor yang berfungsi sebagai rudder , untuk membelokkan kemudi balon udara dan untuk pergerakan maju dan mundur. Berikut adalah gambar rencana pergerakan motor rudder untuk bergerak.



Gambar 3.14 Pergerakan motor rudder

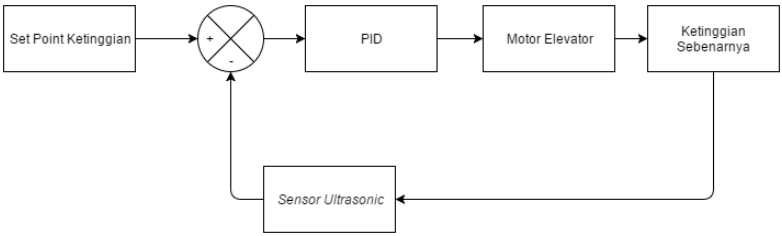
3.3.5 Kendali PID

Untuk mengatur kecepatan dan arah motor dc yang digunakan, penggerak kanan dan kiri pada sistem pengatur posisi balon udara diproses dengan kontrol PID terlebih dahulu. Dalam metode pelacakan, selisih dari posisi piksel sasaran terhadap posisi posisi tengah frame (set point) dianggap sebagai error untuk masukan E pada perhitungan kontrol PID oleh processing unit. Kontrol error akan dimasukkan ke dalam algoritma PID dan sinyal output akan digunakan untuk mengatur PWM pada kecepatan motor.



Gambar 3.15 Blok Diagram PID posisi

Sedangkan pada kontrol ketinggian, error akan diambil dari set poin ketinggian yang akan diinput dari software dikurangi dari feedback dari sensor ultrasonic, semakin jauh titik dari set poin maka semakin cepat putaran motor elevator yang diperlukan untuk menaikkan balon udara.



Gambar 3.16 Blok Diagram PID Ketinggian

BAB IV
PENGUJIAN DAN PEMBAHASAN SISTEM

4.1 Realisasi dan Desain Balon Udara

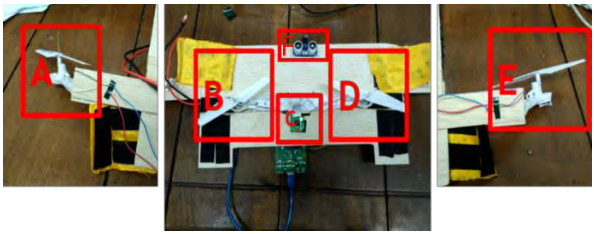
Pada tugas akhir ini digunakan dua balon udara air swimmer yang akan diisi dengan helium. Volume dari balon udara ini adalah 0.1274 m3. Dua balon udara air swimmer mempunyai volume untuk diisi helium sebesar 0.2548 m3. Gambar realisasi balon udara yang digunakan dapat dilihat pada Gambar 4.1.

Helium utamanya digunakan sebagai gas pengangkat pada kapal udara. Permintaan atas gas helium meningkat semasa Perang Dunia II. Spektrometer massa helium juga sangat vital dalam proyek bom atom Manhattan. Dengan isian 28316 liter helium balon udara mampu mengangkat besaran 29,85 Kg atau 1m3 helium dapat mengangkat 960 gram. Sehingga balon udara air swimmer dengan volume 0.2548 m3 dapat mengangkat beban sekitar 240 gram.

Gambar 4.1 Realisasi balon udara yang digunakan

Tabel 4.1 Tabel kemampuan lifting balon udara

	Weight of Lifting Gas (per 1,000 cu. ft.)	Weight of Air (per 1,000 cu. ft.)	Net Lift (per 1,000 cu. ft.)
Hydrogen	5.31 lbs	76.36 lbs	71.05 lbs
Helium	10.54 lbs	76.36 lbs	65.82 lbs



Gambar 4.2 Realisasi kotak kontrol dan lokasi motor

Keterangan:
A = Motor Samping Kanan
B = Motor Bawah Kanan
C = Kamera
D = Motor Bawah Kiri
E = Motor Samping Kiri
F = Sensor Ultrasonic

4.2 Pengujian Sensor Ultrasonic

Pengujian sensor ultrasonic ini dilakukan dengan membandingkan nilai dari sensor ultrasonic dan nilai dari alat pengukur berupa meteran. Sensor ultrasonic ini diuji dari nilai 1 cm sampai dengan 300 cm, dan akan dilihat berapa kisaran nilai errornya. Menurut datasheet sensor yang digunakan sensor mampu mendeteksi hingga 4 meter dengan dataran yang merata. Sehingga sinyal pantulan ultrasonic tidak terbias dan menimbulkan kesalahan terhadap sensor. Feedback pada kontrol PID ketinggian juga berasal dari sensor ultrasonic, sehingga ketinggian balon udara dapat diketahui secara realtime.

Tabel 4.2 Pengujian Sensor Ultrasonic

No.	Jarak sebenarnya (cm)	Pembacaan sensor ultrasonic (cm)	Error (cm)
1	1	2,8	1,8
2	10	10,3	0,3
3	20	20	0
4	30	30,3	0,3
5	40	42,1	0,1
6	50	48,9	1,1
7	60	59,8	0,2
8	70	70,4	0,4
9	80	79,5	0,5
10	90	90,4	0,4
11	100	99,6	0,4
12	110	109	1
13	120	119,2	0,8
14	130	129,7	0,3
15	140	140,1	0,1
16	150	150,3	0,3
17	160	160,2	0,2
18	170	170,6	0,6
19	180	180,3	0,3
20	190	190,2	0,2
21	200	200,3	0,3
22	210	209,5	0,5
23	220	220,3	0,3
24	230	230,1	0,1
25	240	240,1	0,1
26	250	250,7	0,7
27	260	260,2	0,2
28	270	269,1	0,9
29	280	281,1	1,1
30	290	289,1	0,9
31	300	302,2	2,2

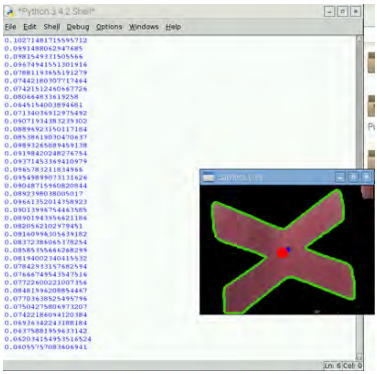
Berdasarkan data yang didapat, dapat ditarik kesimpulan bahwa sensor digunakan bekerja dengan baik dan mampu untuk mengukur jarak ketinggian hingga 300cm, dengan error rata-rata sebesar 7% . Error terbesar saat dilakukan pengukuran jarak rendah dibawah 4cm , dikarenakan sensor ultrasonic bekerja dibawah range kerjanya.

4.3 Pengujian Deteksi Landasan dengan Kamera

Pada pengujian ini akan diambil data pendeteksian landasan dengan kamera, metode yang digunakan dalam pendeteksian landasan ini adalah hu moments matching. Metode ini dapat mendeteksi perubahan target landasan berupa rotasi, skala, dan translasi yang dirasa akan bekerja dengan baik untuk pendeteksian landasan.

4.3.1 Pengujian Nilai Matching Terhadap Jarak

Pengujian kali ini dilakukan pengambilan data tentang hasil nilai matching berdasarkan jarak. Pengujian ini berguna untuk mengetahui rentang nilai hasil matching, sehingga pada program pengenalan landasan dapat menyesuaikan dengan nilai matching yang dapat dianggap identik / valid sebagai landasan. Rata-rata nilai matching dihitung berdasarkan 20 frame pendeteksian.



Gambar 4.3 Hasil nilai matching pada jarak 20cm

Tabel 4.3 Rata-rata nilai matching terhadap jarak

No.	Jarak (cm)	Rata-rata nilai Matching
1	20	0.073
2	40	0.043
3	60	0.051
4	80	0.057
5	100	0.076
6	120	0.112
7	140	0.154
8	160	0.129
9	180	0.172
10	200	0.191

Dari data penghitungan yang dilakukan oleh fungsi pada pustaka opencv, nilai matching dari gambar yang ditangkap oleh kamera berada dibawah 0.2. Sehingga batasan pada program opencv dapat disesuaikan, Data pengujian dan akurasi dari pendeteksi landasan dapat ditingkatkan dengan membatasi nilai matching.

4.3.2 Pengujian Tingkat Pendeteksi Landing Pad

Dengan diterapkannya batasan nilai matching akan dilakukan pengujian ulang untuk menghitung ketepatan sistem untuk mendeteksi landing pad. Pengujian ini dilakukan untuk membuktikan ketepatan metode *Hu moments* untuk *contour matching* dalam mendeteksi perubahan target landasan yang berubah secara skala, rotasi, dan translasi. Pengujian pertama ini dilakukan tanpa gangguan objek dengan warna yang sama dengan landing pad dan jarak dari kamera menuju landasan diubah untuk menguji tingkat keberhasilan metode *contour matching* dalam mendeteksi perubahan terhadap besarnya citra landasan yang tertangkap pada kamera. Tingkat keberhasilan pendeteksian dihitung berdasarkan banyaknya frame yang terdeteksi pada setiap 20 frame gambar kamera yang diambil.

Tabel 4.4 Pengujian tingkat keberhasilan pendeteksian berdasarkan jarak

No.	Jarak (cm)	Tingkat keberhasilan
1	20	100%
2	40	100%
3	60	100%
4	80	95%
5	100	100%
6	120	95%
7	140	85%
8	160	90%
9	180	90%
10	200	85%

Setelah diuji keakurasian dari pendeteksian landing pad berdasarkan jarak, pengujian selanjutnya akan dilakukan berdasarkan sudut pandang dari kamera. Pengujian ini dilakukan untuk menguji tingkat keberhasilan pendeteksian metode *Hu moments contour matching* yang citra gambar landasannya berubah secara sudut pandang kamera. pengujian dilakukan dengan merubah sudut kamera sebesar 15^o,30^o, dan 45^o terhadap landasan. Tingkat keberhasilan pendeteksian dihitung berdasarkan banyaknya frame yang terdeteksi pada setiap 20 frame gambar kamera yang diambil.

Tabel 4.5 Tingkat keberhasilan pendeteksian berdasarkan sudut kamera

No.	Sudut (^{o})	Tingkat keberhasilan
1	15	55%
2	30	20%
3	45	0%

Data selanjutnya yang akan diambil adalah bagaimana tingkat pendeteksian dari sistem apabila landing pad berotasi. Pengujian ini dilakukan untuk menguji tingkat keberhasilan pendeteksian metode Hu moments contour matching yang citra gambar landasannya berubah secara rotasi sebesar 0^o, 60^o, 120^o, 180^o, 240^o, dan 300^o. Pengujian dilakukan dengan jarak 100cm. Tingkat keberhasilan pendeteksian

dihitung berdasarkan banyaknya frame yang terdeteksi pada setiap 20 frame gambar kamera yang diambil.

Tabel 4.6 Pengujian tingkat keberhasilan pendeteksian berdasarkan rotasi

No.	Rotasi (°)	Tingkat keberhasilan
1	0	95%
2	60	90%
3	120	100%
4	180	100%
5	240	90%
6	300	90%

Dari serangkaian pengujian deteksi landing pad dengan metode contour matching dari Hu Moments didapatkan landing pad dapat terdeteksi secara rotasi dan skala namun apabila terjadi perubahan sudut pandang kamera, metode ini tidak begitu efektif .

4.4 Pengujian Kendali PID pada ketinggian balon udara

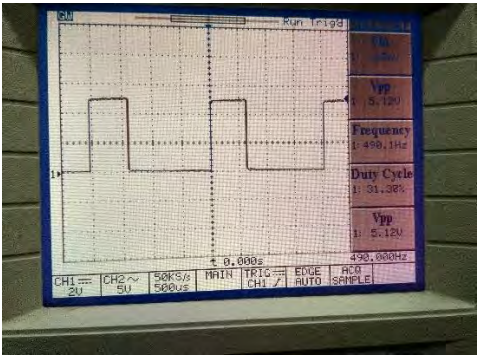
Pada pengujian ini akan diambil data uji respon pertama dari sinyal kontrol berupa *duty cycle* sinyal PWM terhadap ketinggian, Pengambilan data uji ini berguna untuk mengetahui seberapa besar *duty cycle* pertama dihasilkan PWM sehingga motor yang digunakan untuk mengangkat balon udara diharapkan dapat bekerja dengan baik. Pengujian ini lakukan dengan konstanta Kp = 1, Ki = 0, dan Kd=0 dengan set poin ketinggian 100 cm dan 200 cm. Pengujian data dilakukan dengan arduino mega, yang mempunyai sinyal PWM beresolusi 8 bit. Data yang diambil pada tabel dan gambar dibawah ini merupakan respon *duty cycle* pertama yang dihasilkan berdasarkan error awal.

4.4.1 Pengujian dengan setpoin 100 cm

Pengujian pengukuran *duty cycle* PWM pertama dilakukan dengan setpoin 100 cm, kemudian jarak ketinggian diubah secara bertahap sebanyak 20cm hingga ketinggian 100cm. Hasil pengujian jarak terhadap *duty cycle* dapat dilihat pada Tabel 4.7 dan hasil keluaran sinyal PWM pada ketinggian 20cm dapat dilihat pada Gambar 4.4.

Tabel 4.7 Pengujian jarak terhadap duty cycle

No.	ketinggian (cm)	Duty Cycle (%)
1	20	31.38
2	40	23.52
3	60	15.68
4	80	8.72
5	100	0

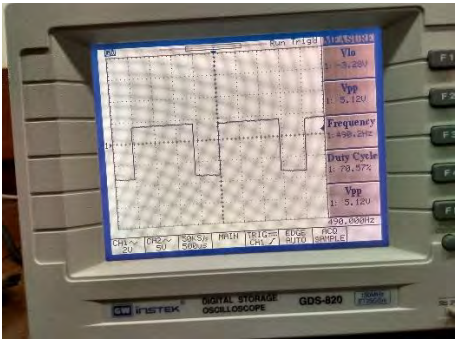


Gambar 4.4 Sinyal keluaran pwm pada ketinggian 20 cm

4.4.2 Pengujian dengan set poin 200 cm

Tabel 4.8 Pengujian jarak terhadap duty cycle 2

No.	ketinggian (cm)	Duty Cycle (%)
1	20	70.57
2	40	62.74
3	60	54.90
4	80	47.08
5	100	39.21
6	120	33.33
7	140	22.55
8	160	15.69
9	180	9.82
10	200	0



Gambar 4.5 Sinyal keluaran pwm pada jarak 20cm

Dari hasil data pertama yang didapatkan dapat ditarik kesimpulan bahwa kontrol PID dapat bekerja dengan baik, dimana semakin jauh jarak antara balon dan set poin yang telah ditentukan semakin besar pula duty cycle yang dihasilkan. Dengan duty cycle yang tinggi kecepatan motor juga akan berputar dengan kecepatan yang tinggi.

4.4.3 Pengujian Kendali PID Ketinggian pada Balon Udara

Setelah pengujian yang dilakukan dengan melihat duty cycle PWM pada kontrol PID balon udara, pengujian selanjutnya dilakukan pada balon udara yang sebelumnya ketinggiannya telah diatur dan digantung dengan tali. Pengujian dilakukan dengan konstanta Kp = 1, Ki = 0, dan Kd=0 serta setpoin ketinggian 100cm. Pengujian dibagi menjadi dua bagian, yaitu saat balon berada dibawah setpoin lalu dinaikkan melewati jarak diatas setpoin dan ketika balon berada diatas setpoin ketinggian lalu diturunkan menuju kebawah setpoin. Berbeda dari pengujian sebelumnya yang hanya menggunakan data error awal sebagai kontrol acuan, sinyal kontrol kali ini akan menggunakan algoritma PID dibawah ini :

$$y\,(t) = K_P\, e(t) + K_i \int_0^t e(t)dt + K_d \left(\frac{de(t)}{dt}\right) \qquad (4.1)$$

Dimana sinyal kontrol tidak hanya bergantung pada error sekarang , tetapi juga pada error sebelumnya. Pada pengujian pertama alat

diposisikan dibawah setpoin jarak setinggi 100 cm, lalu secara perlahan dinaikkan melewati setpoin jarak. Pengujian pertama yang telah dilakukan pada posisi dibawah jarak setpoin didapatkan motor langsung berputar dan kecepatannya terus bertambah dengan cepat, Setelah dinaikkan secara perlahan melewati setpoin kecepatan motor berkurang secara cepat dan mati hingga posisi kembali berada dibawah setpoin yang telah ditentukan.

Tabel 4.9 Uji kontrol PID ketinggian 1

No.	Posisi	Keadaan motor
1	Posisi awal dibawah setpoin	Motor berputar dan kecepatan motornya bertambah hingga maksimal.
2	Dinaikkan secara perlahan melewati setpoin	Kecepatan motor turun hingga terhenti.

Sedangkan pada pengujian kedua alat diposisikan diatas setpoin jarak setinggi 100 cm, lalu secara perlahan diturunkan melewati setpoin jarak. Pengujian kedua ini dilakukan pada posisi diatas jarak setpoin didapatkan motor tidak berputar. Ketika posisi alat diturunkan hingga posisi jarak dibawah setpoin, motor mulai berputar dan kecepatan putaran motornya bertambah.

Tabel 4.10 Uji kontrol PID ketinggian 2

No.	Posisi	Keadaan motor
1	Posisi awal diatas setpoin	Motor tidak berputar.
2	Diturunkan secara perlahan dibawah setpoin	Kecepatan motor bertambah hingga kecepatan maksimal.

Dari kedua percobaan diatas respon motor sudah sesuai dengan prinsip kontrol PID. Sinyal kontrol akan terus bertambah hingga melewati setpoin yang telah ditentukan, setelah setpoin terlewati sinyal kontrol akan berkurang sehingga hasil kontrol dan terus berulang sehingga sinyal kontrol akan mempertahankan besaran koreksi error disekitar setpoin.

4.5 Pengujian Motor pada kontrol PID Posisi

Dengan menggunakan metode yang telah dijelaskan pada bagian perancangan sistem, maka dilakukan pengujian terhadap sistem yang telah dibuat. Error dari kontrol PID posisi ini adalah merupakan jarak (piksel) antara posisi tengah frame (setpoin) dan posisi dari landasan yang terdeteksi. Error tersebut akan menghasilkan berupa sinyal kontrol PWM yang akan bertambah jika error membesar. Percobaan dilakukan dengan balon udara yang diikat dan dilakukan pengamatan pada respon motor pada balon udara yang mengatur posisi. Motor akan melakukan gerakan yang berfungsi untuk berbelok ke kiri depan, kanan depan, kiri belakang, dan kanan belakang. Papan landasan yang telah ditentukan akan digerakkan ke kanan depan, kiri depan, kanan belakang, kiri belakang, dan pada posisi tengah kamera akan dilihat pergerakan motor pada kontrol posisi. Percobaan kali ini dilakukan dengan Kp=1, Ki=0, dan Kd=0 dengan setpoin adalah posisi tengah frame pada kamera.

Tabel 4.11 Uji kontrol PID posisi

No.	error (piksel)	Posisi pola landasan	Duty Cycle motor kiri(%)	Duty Cycle motor kanan(%)
1	50	Kanan atas	18.7	4.7
2	55	Kanan bawah	4.8	20.3
3	57	Kiri atas	4.4	21.5
4	53	Kiri bawah	19.3	4.8
5	35	Maju	14.2	14.6
6	37	Mundur	14.8	14.3
7	12	Berhenti	0	0

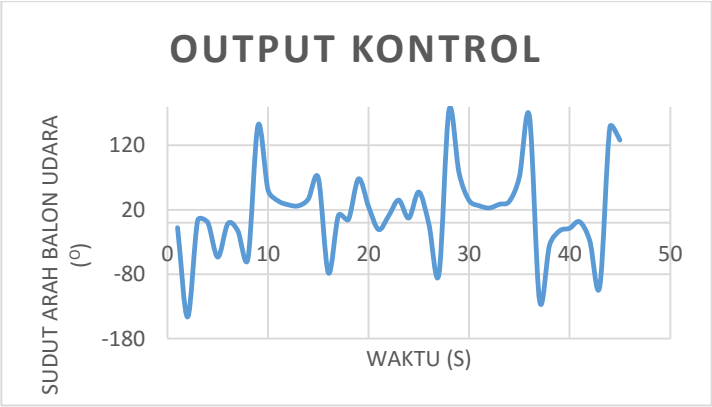
Dari percobaan yang telah dilakukan semakin jauh jarak landasan dari setpoin semakin cepat putaran motor, semakin dekat jarak landasan dari setpoin semakin rendah putaran motor. Ketika posisi landasan berada pada posisi tengah kamera (setpoin) putaran motor pengatur posisi berhenti. Pergerakan motor untuk mengatur posisi sudah sesuai dengan kontrol yang diinginkan.

4.6 Kendali PID pada arah gerak balon

Pengujian terakhir yang akan dilakukan adalah dengan menguji kendali PID pada arah gerak maju balon terhadap target, pengujian ini bertujuan untuk mengatur pergerakan balon udara dalam menuju titik landasan dengan tepat dan akurat. arah maju balon akan sejajar menuju landasan.

4.6.1 Pengujian dengan kendali P

Pengujian pertama dengan kendali tipe P dilakukan dengan mengatur Kp=1 , Ki=0 , dan Kd=0. Pengujian ini bertujuan untuk melihat output kontrol berupa sudut arah gerak balon udara. Berikut adalah respon kontrol hasil dari pengujian gerak arah balon dengan set poin depan balon udara yang terletak pada 0º pada sudut frame kamera.



Gambar 4.6 Grafik Respon kontrol sudut arah balon terhadap set poin

Dari data yang diperoleh dapat dilihat bahwa respon kontrol dari kendali PID pada arah gerak balon udara sudah mempunyai kemampuan untuk mengoreksi error untuk menuju pada titik setpoin 0^o dalam arah gerak balon, namun dikarenakan respon dari kontroler P yang kurang cepat sehingga offset dan error yang terjadi masih relatif besar.

4.6.2 Pengujian dengan kendali PID

Maka dari itu dengan metode tuning PID ziegler-nichols yang pertama akan ditentukan konstanta kp, ki, dan kd untuk mendapatkan respon yang plant yang lebih baik. Dengan memberi unit step pada balon udara kita dapat menentukan konstanta yang tepat respon kontrol yang terjadi pada balon udara. Berikut adalah perhitungan output dari plant yang telah diberikan, konstanta kp, ki, dan kd dapat ditentukan dengan tabel formula ziegler-nichols. Dari gambar respon output yang telah diperoleh nilai dari T dan L telah diketahui.

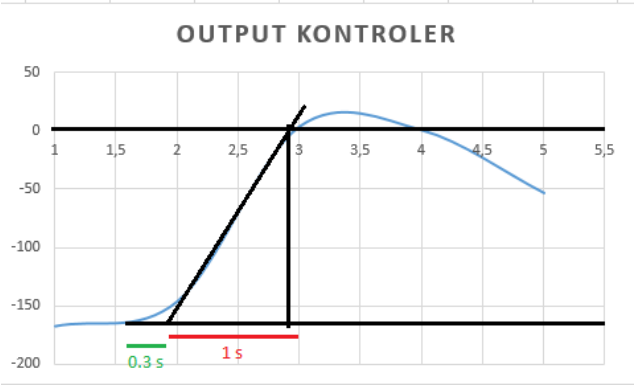
$$T = 1 \text{ s}$$

$$L = 0.3 \text{ s}$$

$$Kp = 1.2\frac{T}{L} = 4$$

$$Ki = 0.6\frac{T}{L^2} = 6.67$$

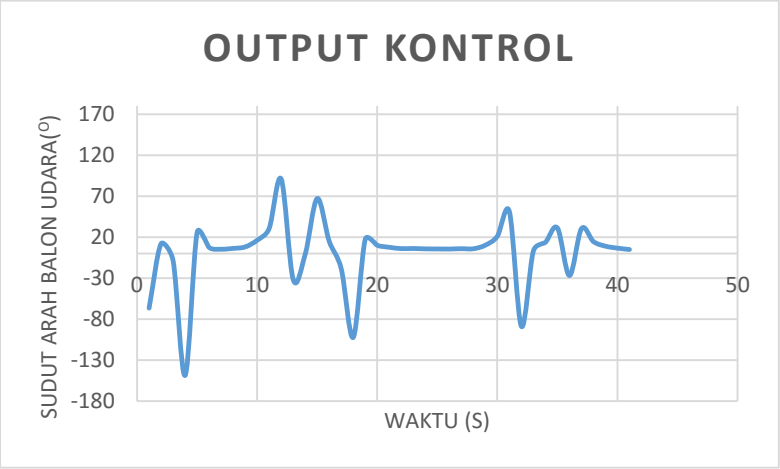
$$Kd = 0.6T = 0.6$$



Gambar 4.7 Output unit step

Setelah mengetahui konstanta Kp, Ki, dan Kd arah gerak balon udara akan diuji kembali, berikut adalah hasil pengujian dari kontrol balon udara setelah Kp, Ki, dan Kd telah diatur menggunakan metode tuning ziegler nichols.

.



Gambar 4.8 Grafik output setelah menggunakan metode ziegler-nichols

Terlihat dari respon kontrol dengan konstanta Kp, Ki, dan Kd yang telah diatur menggunakan metode ziegler nichols mempunyai respon yang lebih cepat dalam mengkoreksi kesalahan sudut arah gerak balon lebih cepat dari respon yang hanya menggunakan kontroler tipe P. Error osilasi pada gerak arah balon yang terjadi dengan menggunakan kontroler PID juga berkurang.

LAMPIRAN

Listing program pengenalan landasan:

```
class ExampleApp(QtGui.QMainWindow, coba1.Ui_MainWindow):
  def __init__(self):
    super(self.__class__, self).__init__()
    self.setupUi(self)
    #
    self.serialView_B.clicked.connect(self.saveSerialView)
    self.debugView_B.clicked.connect(self.saveDebug)
    self.Start_B.clicked.connect(self.startTracking)
    self.startTrack = False
    #
    if (useSerial==1):
      self.ser = serial.Serial(
        port = serialPort,
        baudrate = baud,
        parity = serial.PARITY_NONE,
        stopbits = serial.STOPBITS_ONE,
        bytesize = serial.EIGHTBITS,
        timeout = 1
      )
      sleep(1)
      self.thread =
threading.Thread(target=self.getSerialData(14),args=(self.ser,))
      self.thread.start()
    self.roiMean = np.zeros((row,column))
    self.setFPS(30)

    if(useLowRes==1):
      self.resolution = (320,240)
    else:
      self.resolution = (640,480)
    if(usePiCamera==1):
```

```
            self.stream = PiVideoStream(resolution =
self.resolution,framerate = self.fps)
            self.stream.start()
        elif(usePiCamera==0):
            self.cap = cv2.VideoCapture(0)
        self.initData(500)

self.serialView.append("hari;bulan;tahun;jam;menit;detik;ax;ay;az;gx;g
y;gz;mx;my;mz;pitch;yaw;roll;heading;height;")
        self.start()

    def setFPS(self, fps):
        self.fps = fps
    def startTracking(self):
        self.startTrack = not self.startTrack
        if (self.startTrack):
            self.Start_B.setText("STOP")
        else:
            self.Start_B.setText("START")
    def nextFrame(self):
        #ambil index tab saat ini
        self.indexTab = self.tabWidget.currentIndex()
        #-----------------------------------------------
        #---------- opencv code --------------------
        #-----------------------------------------------
        #load template landing
        self.imagetemplate = cv2.imread('testo.jpg',1)
        self.mask2 = cv2.cvtColor(self.imagetemplate,
cv2.COLOR_BGR2GRAY)
        ret,self.mask2 =
cv2.threshold(self.mask2,127,255,cv2.THRESH_BINARY)
        #penggunaan kamera
        if(usePiCamera==1):
            orimg = self.stream.read()
        elif(usePiCamera==0):
            ret, orimg = self.cap.read()
```

```
            if useLowRes==1:
                orimg =
cv2.resize(orimg,(self.resolution[0],self.resolution[1]),interpolation =
cv2.INTER_CUBIC)
        else:
            orimg = cv2.imread('4.jpg',1)
            if useLowRes==1:
                orimg =
cv2.resize(orimg,(self.resolution[0],self.resolution[1]),interpolation =
cv2.INTER_CUBIC)
        #flip
        orimg = cv2.flip(orimg,1)
self.cx,self.cy,self.kontur,self.kontur2=self.pengenalanland(self.hsv,self.
mask2)
            self.hsv = cv2.bitwise_and(orimg,orimg,mask = self.hsv)
            cv2.drawContours(self.mask2, self.kontur2, -1, (20,200,65), 3)
            cv2.imshow("template",self.mask2)
            if(self.cx!='none'):
                print(self.cx,self.cy)
                cv2.circle(self.hsv,(self.cx,self.cy), 5, (0,0,255), -1)
            Qimg2 = QtGui.QImage(self.hsv, self.hsv.shape[1],
self.hsv.shape[0], QtGui.QImage.Format_RGB888)
            Qimg2 = Qimg2.scaled(self.cvImage_2.size(),
Qt.KeepAspectRatio)
            pix2 = QtGui.QPixmap.fromImage(Qimg2)
            self.cvImage_2.setPixmap(pix2)

    def preprocessingImage(self, source):
        #dapatkan ukuran gambar
        if (usePiCamera==0 or usePiCamera==2):
            self.width  = source.shape[1]
            self.height = source.shape[0]
        else:
            self.width  = self.resolution[1]
            self.height = self.resolution[0]
        #konversi gambar BGR ke HSV
```

```
        img = cv2.cvtColor(source, cv2.COLOR_RGB2HSV)
        #ambil nilai dari slider bar
        Low = np.array([self.LowH_T.value(), self.LowS_T.value(),
self.LowV_T.value()])
        High = np.array([self.HighH_T.value(), self.HighS_T.value(),
self.HighV_T.value()])
        Low2 = np.array([self.LowH_T_2.value(), self.LowS_T_2.value(),
self.LowV_T_2.value()])
        High2 = np.array([self.HighH_T_2.value(),
self.HighS_T_2.value(), self.HighV_T_2.value()])
        # Low2 = np.array([50,0,0])
        # High2 = np.array([179,255,255])
        #filter untuk landing
        destination2 = cv2.inRange(img, Low2, High2)
        ret,destination2 = cv2.threshold(destination2,127,255,0)
        #filter untuk pipe tracking
        destination = cv2.inRange(img, Low, High)
        #deklarasi bentuk morfologi
        kernel = cv2.getStructuringElement(cv2.MORPH_ELLIPSE, (5,5))

        #morfologi untuk destination 3xOpen,1xClose,1xOpen
        destination = cv2.morphologyEx(destination, cv2.MORPH_OPEN,
kernel)
        destination = cv2.morphologyEx(destination, cv2.MORPH_OPEN,
kernel)
        destination = cv2.morphologyEx(destination, cv2.MORPH_OPEN,
kernel)
        destination = cv2.morphologyEx(destination,
cv2.MORPH_CLOSE, kernel)
        destination = cv2.morphologyEx(destination, cv2.MORPH_OPEN,
kernel)
        #morfologi untuk destination2
        destination2 = cv2.morphologyEx(destination2,
cv2.MORPH_OPEN, kernel)
        destination2 = cv2.morphologyEx(destination2,
cv2.MORPH_OPEN, kernel)
```

```
        destination2 = cv2.morphologyEx(destination2,
cv2.MORPH_OPEN, kernel)
        destination2 = cv2.morphologyEx(destination2,
cv2.MORPH_CLOSE, kernel)
        destination2 = cv2.morphologyEx(destination2,
cv2.MORPH_OPEN, kernel)
        #kembalikan frame
        return destination , destination2

    def pengenalanland(self, mask, mask2):
        cx = 'none'
        cy = 'none'
        nilaimatch=20
        contours = cv2.findContours(mask,2,1)
        imagez2,contours2,hierarchy2 = cv2.findContours(mask2, 2, 1)
        cnt2 = contours2[-1]
        contours = contours[0] if imutils.is_cv2() else contours[1]
        Index = 0
        if len(contours) > 0:
            for cnt in contours:
                ret = cv2.matchShapes(cnt,cnt2,1,0.0)
                if(nilaimatch > ret and ret>0 and ret<0.4):
                    nilaimatch=ret
                    Index=cnt
            M = cv2.moments(Index)
            if (int(M['m00'])!=0):
                cx = int(M['m10']/M['m00'])
                cy = int(M['m01']/M['m00'])
                ex = self.resolution[0]/2 - cx
                ey = self.resolution[1]/2 - cy
                error = pow(abs(pow(ex,2)-pow(ey,2)),0.5)
                deg = cv2.fastAtan2(int(ey),int(ex))
                print(error, deg)
        return cx,cy,Index,cnt2

    def start(self):
```

```
        self.timer = QtCore.QTimer()
        self.timer.timeout.connect(self.nextFrame)
        self.timer.start(1000./self.fps)

    def stop(self):
        self.timer.stop()

    def deleteLater(self):
        self.cap.release()
        super(QtGui.QWidget, self).deleteLater()
def main():
    app = QtGui.QApplication(sys.argv)
    form = ExampleApp()
    form.show()
    app.exec_()


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

listing program arduino:

```
#define IN1 2
#define IN2 3
#define IN3 4
#define IN4 5
int ENA=9;
int ENB=10;
String serialDataIn;
int kp = ;
int ki = ;
int kd = ;
int sinyalkontrol;
int setpoin = 100;
int e0;
int e1;
```

```
String inbyte;
void setup(){
 Serial.begin(115200);
 Serial.setTimeout(10);
 pinMode(ENA, OUTPUT);
 pinMode(ENB, OUTPUT);
 pinMode(IN1, OUTPUT);
 pinMode(IN2, OUTPUT);
 pinMode(IN3, OUTPUT);
 pinMode(IN4, OUTPUT);
}
void loop(){
   if(Serial.available()){
  inbyte = Serial.readString();}
  else{
  int dir_maju = getValue(inbyte,';',0).toInt();
  int speed_maju = getValue(inbyte,';',1).toInt();
  int dir_Ki = getValue(inbyte,';',2).toInt();
  int speed_Ki = getValue(inbyte,';',3).toInt();
  int dir_Ka = getValue(inbyte,';',4).toInt();
  int speed_Ka = getValue(inbyte,';',5).toInt();
  int flag = getValue(inbyte,';',6).toInt();
  int controlMode = getValue(inbyte,';',7).toInt();
  if (controlMode==1){
  digitalWrite(IN1, HIGH);
  digitalWrite(IN2, LOW);
  digitalWrite(IN3, HIGH);
  digitalWrite(IN4, LOW);
  long duration, distance;
  digitalWrite(trigPin, LOW);  // Added this line
  delayMicroseconds(2); // Added this line
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10); // Added this line
  digitalWrite(trigPin, LOW);
  duration = pulseIn(echoPin, HIGH);
```

```
  distance = (duration/2) / 29.1;
  Serial.println(distance);
  if(sinyalkontrol<0){sinyalkontrol=0;}
  if((distance)<450 && sinyalkontrol>-1 &&
sinyalkontrol<(maxpwm+1)){
  sinyalkontrol = kp * e0 + ki *(e1+e0) + kd * (e1-e0);
  if(sinyalkontrol>maxpwm){sinyalkontrol=maxpwm;}
  e1 = e0;
  e0 = setpoin - distance;
  {analogWrite(ENA, sinyalkontrol);
   analogWrite(ENB, sinyalkontrol);}
  if(sinyalkontrol<0)
  {analogWrite(ENA, 0);
    analogWrite(ENB, 0);}}
}}}
void motor_kiri(int dir, int pwm){
  if(dir!=0){
    digitalWrite(IN1, HIGH);
    digitalWrite(IN2, LOW);
    analogWrite(ENA, pwm);
    digitalWrite(IN3, HIGH);
    digitalWrite(IN4, LOW);
    analogWrite(ENB, int(pwm/2));
  }else{
    digitalWrite(IN1, LOW);
    digitalWrite(IN2, HIGH);
    analogWrite(ENA, pwm);
    digitalWrite(IN3, LOW);
    digitalWrite(IN4, HIGH);
    analogWrite(ENB, int(pwm/2));
  }
}
void motor_kanan(int dir, int pwm){
  if(dir!=0){
    digitalWrite(IN3, HIGH);
    digitalWrite(IN4, LOW);
```

```
    analogWrite(ENB, pwm);
    digitalWrite(IN1, HIGH);
    digitalWrite(IN2, LOW);
    analogWrite(ENA, int(pwm/2));
   }else{
    digitalWrite(IN3, LOW);
    digitalWrite(IN4, HIGH);
    analogWrite(ENB, pwm);
    digitalWrite(IN1, LOW);
    digitalWrite(IN2, HIGH);
    analogWrite(ENA, int(pwm/2));
   }
}
void motor_maju(int dir, int pwm){
  if(dir!=0){
    digitalWrite(IN1, HIGH);
    digitalWrite(IN2, LOW);
    analogWrite(ENA, pwm);
    digitalWrite(IN3, HIGH);
    digitalWrite(IN4, LOW);
    analogWrite(ENB, pwm);
   }else{
    digitalWrite(IN1, LOW);
    digitalWrite(IN2, HIGH);
    analogWrite(ENA, pwm);
    digitalWrite(IN3, LOW);
    digitalWrite(IN4, HIGH);
    analogWrite(ENB, pwm);
   }
}
String getValue(String data, char separator, int index){
 int found = 0;
 int strIndex[] = { 0, -1  };
 int maxIndex = data.length()-1;
 for(int i=0; i<=maxIndex && found<=index; i++){
  if(data.charAt(i)==separator || i==maxIndex){
```

```
  found++;
  strIndex[0] = strIndex[1]+1;
  strIndex[1] = (i == maxIndex) ? i+1 : i;
 }
}
return found>index ? data.substring(strIndex[0], strIndex[1]) : "";
}
```

BAB V
PENUTUP

Dari serangkaian penellitian yang dilakukan didapatkan saran dan kesimpulan sebagai berikut:

5.1 Kesimpulan

Berdasarkan hasil pengujian dan analisis, dapat diperoleh kesimpulan sebagai berikut:

1. Penggunaan metode contour matching dengan Hu moments mempunyai ketelitian yang tinggi untuk perubahan skala, rotasi, dan translasi. Pada translasi keakurasian pendeteksian mencapai 94% dengan perubahan besar landasan pacu terhadap ketinggian dan pada rotasi metode ini juga mencapai 94% , namun jika terjadi perubahan sudut pandang kamera keakurasian pendeteksian landasan berada dibawah 55%. Sehingga metode ini tidak cocok apabila sudut pandang kamera berubah terlalu besar.
2. Penggunaan kontrol PID lebih efektif daripada menggunakan hanya kontroler tipe P , kontroller PID pada balon udara yang digunakan mampu mengurangi sudut osilasi pada kontrol arah balon sebesar 50^{o} dan mampu mempertahankan nilai setpoin lebih lama dibandingkan kontroller tipe P.

5.2 Saran

1. Penggunaan raspberry pi sebagai pengolah citra dirasa kurang jika dijalankan dengan resolusi diatas 320 x 240 yang akan memberikan FPS (frame per second) yang rendah, diharapkan penggunaan processing unit dapat diganti dengan PC yang memiliki processor lebih cepat.
2. Volume balon udara yang akan digunakan untuk mengangkat beban sebaiknya mempunyai perbandingan minimal 1:1 antara 1 meter kubik helium dan 1 kg beban. Sehingga beban motor elevator untuk mengangkat beban tidak terlalu besar.
3. Diperlukan sebuah sistem yang berguna untuk mengatasi kekurangan pendeteksian landasan jika sudut pandang kamera berubah secara sudut, sistem dapat berupa sensor IMU yang dapat

mendeteksi seberapa besar sudut kamera bergerak dan dapat mengurangi efek dari gerakan yang menyebabkan sudut kamera tersebut berubah.

DAFTAR PUSTAKA

[1] Tan Kok Ping, Julian; Eng Ling, Ang; Jun Quan, Tan; Yea Dat, Chua. (2012). "Generic Unmanned Aerial Vehicle (UAV) for civilian application". Faculty of Engineering and Science, Universiti Tuanku Abdul Rahman (UTAR), Kuala Lumpur, Malaysia.

[2] Cho, Am; Kim, Jihoon; Lee, Sanghyo; Choi, Sujin; Lee, Boram; Kim, Bosung;Park, Noha; Kim, Dongkeon; Kee, Changdon. (2007). "Fully Automatic Taxiing, Takeoff and Landing of a UAV using a Single-Antenna GPS Receiver only". GNSS Lab, School of Mechanical and Aerospace Engineering, Seoul National University, Seoul, Korea.

[3] Hain, James. (2000)."Lighter-than-air Platforms (Blimps and Aerostats) for Oceanographic and Atmospheric Research and Monitoring". Associated Scientists at Woods Hole.

[4] Jerónimo, David; Alcácer, Ricardo; Alegria, F. C.; Lima, Pedro U. (2015). "Line Following and Ground Vehicle Tracking by an Autonomous Aerial Blimp". Institute for Systems and Robotics Lisboa.

[5] Carullo, Alessio; Parvis, Marco. (2001)."An Ultrasonic Sensor for Distance Measurement in Automotive Applications". Senior Member, IEEE.

[6] Ginkel, Robbert; Meerman, Iris; Mulder, Timo; Peters, Jorn. (2013). "Autonomous Landing of a Quadcopter on a Predefined Marker". Universiteit van Amsterdam.

[7] Suzuki, S. et al. (1985). "Topological structural analysis of digitized binary images by border following". Computer Vision, Graphics, and Image Processing.

[8] Hu, M.-K. (1962). "Visual pattern recognition by moment invariants". Information Theory, IRE Transactions on.

[9] Lin, Feng; Duan, Haidong; Qu, Xiaoguang. (2014). "PID Control Strategy for UAV Flight Control System based on Improved Genetic Algorithm Optimization". Automation Department, Shenyang Aerospace University, Shenyang 110136, China.

BIODATA PENULIS

Agung Andri Kurniawan dilahirkan di Nganjuk, pada tanggal 13 Mei 1994 dari pasangan Bapak Sukadarjono dan Ibu Purwantiningsih. Penulis adalah anak pertama dari dua bersaudara. Penulis menyelesaikan pendidikan dasar di SDN Gayungan 1 Surabaya dilanjutkan dengan pendidikan menengah di SMPN 22 Surabaya dan SMAN 15 Surabaya. Pada tahun 2012 Penulis memulai pendidikan di Jurusan Teknik Elektro, Fakultas Teknologi Industri, Institut Teknologi Sepuluh Nopember (ITS) Surabaya. Selama kuliah Penulis aktif membantu penyelenggaraan kegiatan dan aktif sebagai asisten laboratorium Elektronika Dasar dan praktikum Elektronika pada semester ganjil 2015-2016.

Email: agung.andri12@mhs.ee.its.ac.id