



**UNIVERSITAS INDONESIA**

**DETEKSI TRAYEKTORI *SHUTTLECOCK* PADA RUANG TIGA  
DIMENSI DENGAN ALGORITMA CAMSHIFT BERBASIS  
KALMAN FILTER DAN EPIPOLAR GEOMETRI**

**SKRIPSI**

**DEAN ZAKA HIDAYAT  
1106016821**

**FAKULTAS TEKNIK UNIVERSITAS INDONESIA  
DEPARTEMEN TEKNIK ELEKTRO  
TEKNIK KOMPUTER  
DEPOK  
JUNI  
2015**



**UNIVERSITAS INDONESIA**

**DETEKSI TRAYEKTORI *SHUTTLECOCK* PADA RUANG TIGA  
DIMENSI DENGAN ALGORITMA CAMSHIFT BERBASIS  
KALMAN FILTER DAN EPIPOLAR GEOMETRI**

**SKRIPSI**

**Diajukan sebagai salah satu syarat untuk memperoleh gelar Sarjana Teknik**

**DEAN ZAKA HIDAYAT  
1106016821**

**FAKULTAS TEKNIK UNIVERSITAS INDONESIA  
DEPARTEMEN TEKNIK ELEKTRO  
TEKNIK KOMPUTER  
DEPOK  
JUNI  
2015**

## **HALAMAN PERNYATAAN ORISINALITAS**

Skripsi ini adalah hasil karya saya sendiri,  
dan semua sumber baik yang dikutip maupun dirujuk  
telah saya nyatakan dengan benar.

Nama : Dean Zaka Hidayat

NPM : 1106016821

Tanda Tangan :



Tanggal : 5 Juni 2015

## HALAMAN PENGESAHAN

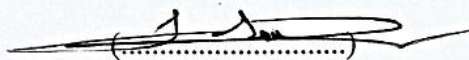
Skripsi ini diajukan oleh :

Nama : Dean Zaka Hidayat  
NPM : 1106016821  
Program Studi : Teknik Komputer  
Judul Skripsi : Deteksi Trayektori *Shuttlecock* Pada Ruang Tiga  
Dimensi menggunakan Algoritma Camshift  
Berdasarkan Kalman Filter dan Epipolar Geometri

Telah berhasil dipertahankan di hadapan Dewan Penguji dan diterima sebagai bagian persyaratan yang diperlukan untuk memperoleh gelar Sarjana Teknik pada Program Studi Teknik Komputer, Fakultas Teknik, Universitas Indonesia.

## DEWAN PENGUJI

Pembimbing : Dr. Ir. Dodi Sudiana, M.Eng.



Penguji : Boma Anantasatya Adhi ST.



Penguji : Muhammad Firdaus Syawalludin Lubis ST. (.....)



Ditetapkan di : Depok

Tanggal : 26 Juni 2015

**HALAMAN PERNYATAAN PERSETUJUAN PUBLIKASI TUGAS  
AKHIR UNTUK KEPENTINGAN AKADEMIS**

Sebagai civitas akademik Universitas Indonesia, saya yang bertanda tangan di bawah ini:

Nama : Dean Zaka Hidayat  
NPM : 1106016821  
Program Studi : Teknik Komputer  
Departemen : Teknik Elektro  
Fakultas : Teknik  
Jenis Karya : Skripsi

Demi pengembangan ilmu pengetahuan, menyetujui untuk memberikan kepada Universitas Indonesia Hak Bebas Royalti Non Eksklusif (Non-Exclusive Royalty Free Right) atas karya ilmiah saya yang berjudul:

Dengan Hak Bebas Royalti Non-Eksklusif ini Universitas Indonesia berhak menyimpan, mengalihmedia/format-kan, mengelola dalam bentuk pangkalan data (database), merawat, dan mempublikasikan skripsi saya selama tetap mencantumkan nama saya sebagai penulis/pencipta dan sebagai pemilik Hak Cipta.

Demikian Pernyataan ini saya buat dengan sebenarnya.

Dibuat di: Depok  
Pada tanggal: 27 Juli 2015  
Yang menyatakan:



Dean Zaka Hidayat

## KATA PENGANTAR

Puji syukur penulis panjatkan kehadirat Allah SWT sebab atas segala rahmat, karunia, dan hidayah-Nya, penulis dapat menyelesaikan skripsi ini dengan baik. Penulis menyadari bahwa skripsi ini tidak dapat diselesaikan tanpa bantuan dari banyak pihak. Oleh karena itu, penulis mengucapkan terima kasih kepada:

- 1) Bapak Dr. Ir. Dodi Sudiana, M.Eng selaku pembimbing seminar yang telah memberikan arahan, koreksi, dukungan, dan waktunya selama penulis mengerjakan seminar ini.
- 2) Orang tua dan keluarga penulis yang telah memberikan dukungan baik secara moril maupun materil sehingga penulis dapat menyelesaikan seminar ini.
- 3) Bapak Muhammad Firdaus Syawalludin Lubis selaku pembimbing riset *computer vision* di Tim Robotika Universitas Indonesia yang telah memberikan banyak bantuan dan masukan dalam membangun sistem deteksi benda di ruang tiga dimensi.
- 4) Teman-teman Tim Robotika Universitas Indonesia Vektor, Lintang, Alif, Handison, Bintang, Farid, Rafi, Sanadhi, Harianto, Dhani, Saifan, Ismi, Shiva, Ghana, Putri atas dukungannya dalam pengerjaan seminar ini.
- 5) Teman-teman teknik komputer 2011 Jodhi, Mitha, Tryan, Ibam, Hafizh, Zhafir, Dinar, Suryo, Emily, Yessy dan Keluarga Departemen Elektro lain.

Depok, Juni 2015

Penulis



## ABSTRAK

Nama : Dean Zaka Hidayat  
Program Studi : Teknik Komputer  
Judul : Deteksi Trayektori *Shuttlecock* pada Ruang Tiga Dimensi menggunakan Algoritma Camshift berbasis Kalman Filter dan Epipolar Geometri

Salah satu hal paling penting dalam penelitian di bidang olahraga bulu tangkis adalah data pergerakan *shuttlecock* yang menggambarkan trayektori dan kecepatan *shuttlecock*. Terdapat beberapa teknik yang dapat dipakai untuk memperoleh data tersebut, salah satunya dengan menggunakan teknik *image processing*, seperti teknik videografi atau optoelektronik. Kelebihan menggunakan kamera untuk mendeteksi gerakan sebuah obyek antara lain biayanya yang cukup murah bila dibandingkan laser dan radar serta kemudahan untuk mendapatkan alat-alat yang dibutuhkan. Adapun masalah yang dihadapi dalam membangun sistem ini adalah di dunia nyata *shuttlecock* bergerak dalam ruang tiga dimensi, sedangkan kamera hanya menangkap gambar dua dimensi. Karena itulah digunakan metode *epipolar geometry stereo vision* yang dioptimasi dengan algoritma camshift berbasis Kalman filter. Metode ini dipilih karena fleksibilitasnya dalam penentuan obyek sehingga obyek dapat dianggap sebagai satu titik ataupun rekonstruksi dari titik-titik yang sama yang dilihat dari perspektif kamera yang berbeda. Hasil pengujian sistem pada obyek bergerak menunjukkan sistem dapat mendeteksi rata-rata 83.33 persen trayektori *shuttlecock* dengan persentase deteksi rata-rata dalam satu trayektori 85.54 persen.

Kata kunci: *epipolar geometry, camshift, Kalman filter, background subtraction, shuttlecock*

## ABSTRACT

Name : Dean Zaka Hidayat  
Major : Computer Engineering  
Title : Shuttlecock Trajectory Detection in Three Dimensional Space using Kalman Filter based Camshift and Epipolar Geometry

One of the most important thing in a badminton sport science research is the data of shuttlecock movements that shows its trajectory and velocity. There are several techniques that can be used to get this, one of them is using image processing techniques, such as videography or optoelectronic techniques. The advantage of using camera to detect motion of an object is the cost is quite low when compared to laser and radar as well as easy to get the tools needed. The problems encountered in building this system is in the real world the shuttlecock move in three-dimensional space, while the camera only captures a two-dimensional image. Because of that, the epipolar geometry stereo vision algorithm method is used. This method is optimized with Kalman filter based camshift algorithm. This method was chosen because of its flexibility in the determination of the object so that the object can be regarded as one point or reconstructed as same points as seen from the perspective of different cameras. The result of the test shows that the system can detect an average of 83.33 percent shuttlecock trajectory with an average detection persentages in the trajectory 85.54 percent.

Keywords: : *epipolar geometry, camshift, Kalman filter, background subtraction, shuttlecock*



## DAFTAR ISI

<b>HALAMAN PERNYATAAN ORISINALITAS .....</b>	<b>i</b>
<b>HALAMAN PENGESAHAN .....</b>	<b>ii</b>
<b>KATA PENGANTAR .....</b>	<b>iii</b>
<b>ABSTRAK .....</b>	<b>v</b>
<b>ABSTRACT .....</b>	<b>vi</b>
<b>DAFTAR ISI .....</b>	<b>vii</b>
<b>DAFTAR GAMBAR .....</b>	<b>ix</b>
<b>DAFTAR TABEL .....</b>	<b>xi</b>
<b>BAB 1 PENDAHULUAN .....</b>	<b>1</b>
1.1 Latar Belakang .....	1
1.2 Tujuan Penelitian .....	2
1.3 Batasan Masalah .....	3
1.4 Metode Penelitian .....	3
1.5 Sistematika Penulisan .....	4
<b>BAB 2 DASAR TEORI .....</b>	<b>5</b>
2.1 Dasar Pengolahan Citra .....	5
2.1.1 Citra Biner (Monokrom) .....	7
2.1.2 Citra Grayscale (skala keabuan) .....	7
2.1.3 Citra Warna (true color) .....	8
2.2 Deteksi obyek dengan Computer Stereo Vision Epipolar geometri .....	9
2.3 Deteksi obyek pada gambar dua dimensi .....	12
2.3.1 Background Subtraction .....	14
2.3.2 Continously Adaptive Meanshift .....	15
2.3.3 Kalman Filter .....	17

2.4 OpenCV sebagai Computer Vision Library .....	19
2.5 Simulasi menggunakan rviz pada ROS (Robot Operating System) .....	20
2.5.1 Simulasi RVIZ.....	22
2.6 Hardware .....	23
2.6.1 Kamera PS3 Eye.....	23
<b>BAB 3 METODE PENELITIAN .....</b>	<b>27</b>
3.1 System Requirement.....	27
3.2 System Modelling.....	28
3.3 System and Software Design.....	30
3.3.1 Desain Perangkat Keras (Hardware) .....	31
3.3.2 Algoritma Perangkat Lunak (Software) .....	31
3.3.3 Background subtraction .....	34
3.3.4 Camshift berbasis Kalman filter.....	35
3.3.5 Epipolar geometri .....	39
<b>BAB 4 UJI COBA DAN ANALISIS .....</b>	<b>41</b>
4.1 Pengaturan dan Kalibrasi Kamera .....	41
4.2 Analisis Posisi Obyek terhadap Akurasi .....	44
4.3 Analisis Pengaruh Arah Gerak Obyek terhadap Hasil Deteksi .....	49
4.4 Analisis Pengaruh Perbedaan Warna Obyek dengan Latar .....	52
4.5 Analisis Pengaruh Cahaya terhadap Hasil Deteksi .....	53
<b>BAB 5 KESIMPULAN .....</b>	<b>55</b>
<b>DAFTAR PUSTAKA .....</b>	<b>57</b>

## DAFTAR GAMBAR

Gambar 2.1. Matriks dari Citra Digital .....	6
Gambar 2.2. Citra Biner .....	7
Gambar 2.3. Citra Grayscale .....	7
Gambar 2.4. Citra Warna .....	8
Gambar 2.5. Pengaturan dasar dua kamera untuk mengambil gambar.....	10
Gambar 2.6. Translasi dan rotasi kamera 2 terhadap kamera 1 .....	12
Gambar 2.7. Diagram algoritma deteksi pada bidang dua dimensi .....	13
Gambar 2.8. Skema background subtraction.....	14
Gambar 2.9. Skema meanshift .....	17
Gambar 2.10. Lambang OpenCV .....	19
Gambar 2.11. Struktur dan konten OpenCV .....	20
Gambar 2.12. Lambang Robot Operating System .....	21
Gambar 2.13. Sistem node dalam ROS.....	22
Gambar 2.14. Tampilan RVIZ .....	22
Gambar 2.15. Playstation 3 Eye yang digunakan sebagai kamera sistem .....	24
Gambar 2.16. Field of View dari kamera Playstation Eye.....	26
Gambar 3.1. Use Case Diagram dari Sistem.....	28
Gambar 3.2. Model sistem dalam bidang x dan y.....	29
Gambar 3.3. Model sistem dalam bidang z.....	29
Gambar 3.4. Diagram Blok .....	30
Gambar 3.5. Flowchart program perekam video .....	31
Gambar 3.6. Pseudocode program perekam video.....	33
Gambar 3.7. Algoritma Background Subtraction .....	34
Gambar 3.8. Pseudocode background subtraction.....	35
Gambar 3.9. Algoritma camshift berbasis Kalman filter .....	36
Gambar 3.10. Pseudocode camshift berbasis Kalman filter .....	39
Gambar 3.11. Algoritma Epipolar geometri .....	39

Gambar 3.12. Pseudocode epipolar geometri .....	40
Gambar 4.1. Model kalibrasi epipolar geometri .....	42
Gambar 4.2. Proses kalibrasi sudut kamera .....	44
Gambar 4.3. Grafik pada $y = 180$ .....	46
Gambar 4.4. Grafik pada $y = 240$ .....	46
Gambar 4.5. Grafik pada $y = 300$ .....	46
Gambar 4.6. Grafik pada $y = 360$ .....	47
Gambar 4.7. Grafik pada $y = 420$ .....	47
Gambar 4.8. Grafik pada $x = 240$ .....	48
Gambar 4.9. Grafik pada $x = 300$ .....	48
Gambar 4.10. Grafik pada $x = 360$ .....	49
Gambar 4.11. Tampilan sistem .....	50
Gambar 4.12. Tampilan Simulasi rviz dengan ROS .....	52
Gambar 4.13. Hasil Percobaan pada Malam Hari.....	54

## **DAFTAR TABEL**

Tabel 2.1. Field of View Playstation Eye	26
Tabel 4.1. Tabel Uji Epipolar geometri	44
Tabel 4.2. Tabel Uji Arah Obyek Mendekati Kamera	50
Tabel 4.3. Tabel Uji Arah Obyek Menjauhi Kamera	51
Tabel 4.4. Tabel Uji Warna Obyek menyerupai Latar	52

## **BAB 1**

### **PENDAHULUAN**

Pada bab ini akan dijabarkan mengenai latar belakang, tujuan, batasan masalah, metode penulisan, serta sistematika penulisan dari penelitian ini.

#### **1.1 Latar Belakang**

Riset di bidang ilmu keolahragaan saat ini mulai berkembang dengan pesat. Olahraga tidak lagi hanya menjadi cabang tersendiri, namun juga menjadi suatu subyek yang sangat menarik bagi para peneliti di seluruh dunia. Studinya meliputi bagaimana tubuh manusia bekerja ketika berolahraga, bagaimana olahraga dapat meningkatkan kesehatan tubuh, hingga riset-riset yang mendukung perkembangan pada berbagai cabang olahraga seperti riset mengenai aerodinamika bola sepak pada olahraga sepak bola.

Salah satu olahraga yang paling sering menjadi subjek riset dalam dunia ilmu keolahragaan adalah olahraga bulu tangkis. Mulai dari jenis raket, jenis *shuttlecock*, hingga pergerakan *shuttlecock* itu sendiri menjadi topik yang sangat menarik untuk diteliti.

Salah satu hal paling penting dalam penelitian di bidang olahraga bulu tangkis adalah data pergerakan *shuttlecock* yang menggambarkan trayektori dan kecepatan *shuttlecock*. Data tersebut cukup sulit untuk didapatkan dikarenakan bentuk shuttle yang unik yang menyebabkan modifikasi sekeci apapun pada *shuttlecock* akan mempengaruhi pergerakan trayektori *shuttlecock* sehingga data tidak lagi valid. Terdapat beberapa teknik yang dapat dipakai untuk melakukan hal ini, antara lain dengan teknologi laser, radar, atau dengan menggunakan teknik *image processing*, seperti teknik videografi atau optoelektronik. Teknologi laser yang dapat dipakai misalnya *LIDAR (Light Radar)* di mana laser ditembakkan ke segala arah untuk mengetahui posisi benda. Hal sama juga dilakukan oleh sistem radar biasa untuk mendeteksi obyek. Sedangkan untuk *image processing*,

digunakan alat berupa kamera dan komputer dengan kemampuan komputasi yang cukup kuat.

Kelebihan menggunakan kamera untuk mendeteksi gerakan sebuah obyek antara lain biayanya yang cukup murah bila dibandingkan laser dan radar serta kemudahan untuk mendapatkan alat-alat yang dibutuhkan. Meskipun, komputasi yang dibutuhkan lebih berat dibandingkan dengan laser atau radar, namun penggunaannya bisa lebih efektif dengan pilihan berbagai macam algoritma yang dapat diterapkan untuk berbagai kasus.

Algoritma dalam pendeteksian obyek bergerak yang dalam hal ini adalah sebuah *shuttlecock* adalah hal yang sangat penting dalam membangun sistem pengambilan data trayektori *shuttlecock* ini. Adapun masalah yang dihadapi dalam membangun sistem ini adalah di dunia nyata *shuttlecock* bergerak dalam ruang tiga dimensi, sedangkan kamera hanya menangkap gambar dua dimensi. Karena itulah diperlukan sistem pengindraan yang dapat mendapatkan posisi benda di ruang tiga dimensi. Sistem tersebut dapat dibuat dengan menerapkan konsep stereo vision.

Dalam stereo vision terdapat beberapa metode yang dapat digunakan. Di dalam sistem ini, digunakan metode *epipolar geometry stereo vision*. Metode ini dipilih karena fleksibilitasnya dalam penentuan obyek sehingga obyek dapat dianggap sebagai satu titik ataupun rekonstruksi dari titik-titik yang sama yang dilihat dari prespektif kamera yang berbeda.

## **1.2 Tujuan Penelitian**

- Mampu membuat alat/sistem yang mampu mendeteksi *shuttlecock* di ruang tiga dimensi
- Mampu membuat alat/sistem yang menganalisa trayektori *shuttlecock* dalam ruang tiga dimensi



- Menganalisis tingkat reliabilitas dan tingkat keakuratan alat/sistem yang telah dibuat dalam mendeteksi posisi *shuttlecock*

### 1.3 Batasan Masalah

Sistem dirancang untuk mendeteksi posisi *shuttlecock* di ruang tiga dimensi dengan kondisi yang disesuaikan. Sistem dikondisikan di luar lapangan. Sistem terdiri dari sebuah PC dan dua unit kamera yang diposisikan di luar lapangan sehingga kedua kamera dapat menangkap seluruh area lapangan bulu tangkis yang merupakan area pergerakan *shuttlecock*. *Shuttlecock* memiliki warna yang berbeda dari latar pengambilan gambar. Kamera akan melakukan pengambilan video dari sudut yang telah diatur sedemikian rupa. Video yang telah diambil dari kedua kamera kemudian akan diproses untuk mendapatkan rekonstruksi posisi, trayektori dan kecepatan dari *shuttlecock*. Seluruh pemrosesan data dari gambar dua dimensi yang ditangkap oleh kamera hingga rekonstruksi posisi *shuttlecock* dalam ruang tiga dimensi relatif terhadap lapangan bulu tangkis dilakukan oleh PC.

### 1.4 Metode Penelitian

Metode penelitian mengadopsi software engineering cycle, yaitu sebagai berikut:

1. *System Requirement*: menentukan spesifikasi sistem dengan melakukan studi literatur dan pengamatan pada aplikasi yang sudah ada.
2. *System and Software Design*: merancang simulasi algoritma
3. *Implementation*: mengimplementasikan rancangan simulasi, yaitu dengan membuat alat yang diintegrasikan dengan sensor.
4. *Testing*: melakukan pengujian terhadap performa alat.
5. *Analisis*: analisis hasil pengujian sistem.

## **1.5 Sistematika Penulisan**

### *Bab 1 Pendahuluan*

Pada bab ini akan dijabarkan mengenai latar belakang, tujuan, batasan masalah, metode penelitian, serta sistematika penulisan.

### *Bab 2 Dasar Teori*

Pada bab ini akan dijelaskan dasar teori dan algoritma yang berkaitan dengan pendeteksian obyek di ruang tiga dimensi. Selain itu juga akan dijelaskan dasar teori mengenai perangkat keras yang digunakan oleh sistem.

### *Bab 3 Metode Penelitian*

Pada bab ini akan dijelaskan mengenai perancangan sistem dan algoritma dalam menunjang pendeteksian *shuttlecock* di ruang tiga dimensi. Perancangan sistem mencakup *system requirement* dan *system software* desain.

### *Bab 4 Uji Coba dan Analisis*

Pada bab ini akan disampaikan analisis dari data hasil sistem yang dibangun

### *Bab 5 Kesimpulan*

Pada bab ini akan disampaikan kesimpulan dan analisis sistem pendeteksian obyek di ruang tiga dimensi.

## **BAB 2**

### **DASAR TEORI**

Pada bab ini akan dibahas mengenai teori-teori yang berkaitan dengan topik penelitian. Adapun yang akan dibahas yaitu deteksi obyek bergerak pada ruang tiga dimensi tertentu dengan salah satu metode dalam stereo vision yaitu epipolar geometri serta teori-teori pendukung untuk membangun sistemnya. Selain itu akan dibahas juga tentang *hardware-hardware* yang akan digunakan dalam sistem.

#### **2.1 Dasar Pengolahan Citra**

Sistem warna pada citra di landasi oleh teori fisika tentang cahaya dan panjang gelombang cahaya serta warna yang di bawa. Dalam ilmu fisika di sampaikan bahwa cahaya adalah gelombang yang memiliki panjang gelombang berbeda beda. Berdasarkan panjang gelombangnya cahaya dibagi menjadi cahaya tampak dan cahaya tidak tampak. cahaya tampak berada pada kisaran panjang gelombang 400 nm sampai dengan 700 nm. di bawah 400 nm merupakan gelombang tidak tampak – ultra violet dan diatas 700 nm merupakan gelombang tidak tampak infra merah.

Pada gelombang tampak, rentang terbesar ada pada gelombang merah, hijau dan biru. Peneliti sepakat bahwa warna warna lain data diperoleh dengan pencampuran ketiga warna tersebut dengan proporsi tertentu. pengkodean warna dengan metode ini di sebut sistem warna R(red), G(green) dan B(blue).

Selain memisahkan warna menjadi 3 komponen warna pokok (RGB) warna juga dapat di wakili dengan Intensitas (tingkat ke -terangan), Hue (warna itu sendiri) dan saturation (kedalaman).

Iluminasi/intensitas merupakan kekuatan cahaya yang diterima dari gelap sampai terang, tanpa peduli warna apa yang di pancarkan. Sedangkan hue adalah warna asli dari cahaya, tanpa peduli kekuatan cahayanya. Sedangkan saturation adalah banyaknya warna putih yang dicampurkan dengan hue. Misalkan kita

membedakan warna merah tua dengan warna merah muda, pada dasarnya merah, hanya untuk merah tua kandungan warna putihnya = 0 sedangkan pada merah muda kandungan warna putih/saturasinya lebih tinggi.

Pengkodean warna seperti ini disebut HIS (*Illumination, Hue, Saturation*) atau HSV (*Hue, Saturation, and Value/Intensitas*). Model warna HSV ini lebih cocok dengan persepsi warna yang dialami manusia.

Citra digital adalah sebuah fungsi 2D,  $f(x,y)$ , yang merupakan fungsi intensitas cahaya, dimana nilai  $x$  dan  $y$  merupakan koordinat spasial dan nilai fungsi di setiap titik  $(x,y)$  merupakan tingkat keabuan citra pada titik tersebut. Citra digital dinyatakan dengan sebuah matriks dimana baris dan kolomnya menyatakan suatu titik pada citra tersebut dan elemen matriksnya (yang disebut sebagai elemen gambar atau piksel) menyatakan tingkat keabuan pada titik tersebut. Matriks dari citra digital berukuran  $N \times M$  (tinggi  $\times$  lebar), dimana:

$$N = \text{jumlah baris} \quad 0 < y \leq N - 1$$

$$M = \text{jumlah kolom} \quad 0 \leq x \leq M - 1$$

$$L = \text{derajat keabuan} \quad 0 \leq f(x,y) \leq L - 1$$

Gambar 2.1 menunjukkan matriks dari citra digital:

$$F(x,y) \approx \begin{matrix} & f(0,0) & f(0,1) & \cdots & f(0,M-1) \\ f(1,0) & f(1,0) & f(1,1) & \cdots & f(1,M-1) \\ \vdots & \vdots & \vdots & & \vdots \\ f(N-1,0) & f(N-1,1) & & & f(N-1,M-1) \end{matrix}$$

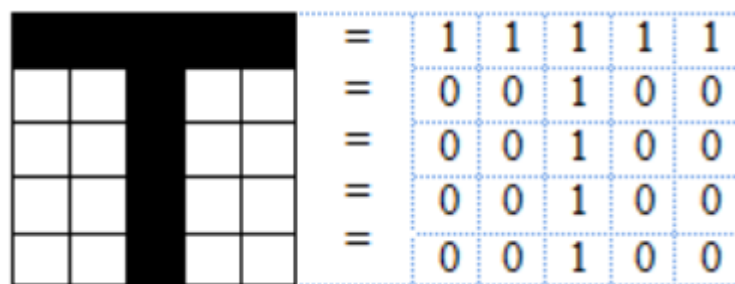
Gambar 2.1 Matriks dari Citra Digital

Dimana indeks baris ( $x$ ) dan indeks kolom ( $y$ ) menyatakan suatu koordinat titik pada citra, sedangkan  $f(x,y)$  merupakan intensitas (derajat keabuan) pada titik  $(x,y)$ .

Berdasarkan jenisnya, citra digital dapat dibagi menjadi 3, yaitu:

#### 2.1.1 Citra Biner (Monokrom)

Memiliki 2 buah warna, yaitu hitam dan putih. Warna hitam bernilai 1 dan warna putih bernilai 0. Untuk menyimpan kedua warna ini dibutuhkan 1 bit di memori. Contoh dari susunan piksel pada citra monokrom dapat dilihat pada gambar 2.2:



Gambar 2.2 Citra Biner

#### 2.1.2 Citra Grayscale (skala keabuan)

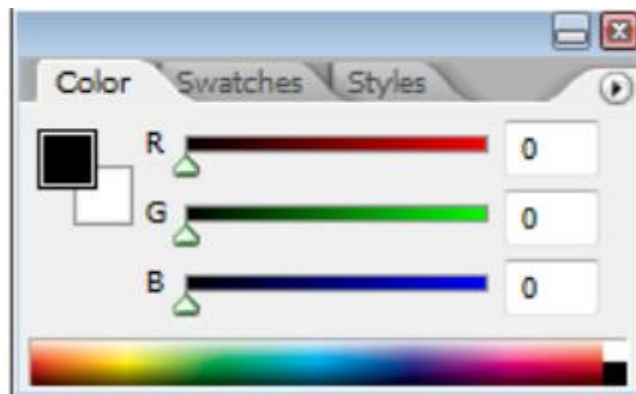
Citra grayscale mempunyai kemungkinan warna hitam untuk nilai minimal dan warna putih untuk nilai maksimal. Banyaknya warna tergantung pada jumlah bit yang disediakan di memori untuk menampung kebutuhan warna tersebut. Semakin besar jumlah bit warna yang disediakan di memori, maka semakin halus gradasi warna yang terbentuk. Contoh:



Gambar 2.3 Citra Grayscale

### 2.1.3 Citra Warna (true color)

Setiap piksel pada citra warna mewakili warna yang merupakan kombinasi tiga warna dasar, yaitu merah, hijau, dan biru (RGB = Red, Green, Blue). Setiap warna dasar menggunakan penyimpanan 8 bit = 1 byte (nilai maksimum 255 warna), jadi satu piksel pada citra warna diwakili oleh 3 byte.



Gambar 2.4 Citra Warna

Pengolahan citra digital adalah salah satu bentuk pemrosesan informasi dengan inputan berupa citra (image) dan keluaran yang juga berupa citra atau dapat juga bagian dari citra tersebut. Tujuan dari pemrosesan ini adalah memperbaiki kualitas citra agar mudah diinterpretasi oleh manusia atau mesin computer. Operasi-operasi pada pengolahan citra digital secara umum dapat diklasifikasikan sebagai berikut:

1. Perbaikan kualitas citra (image enhancement), contohnya perbaikan kontras gelap/terang, penajaman (sharpening), dan perbaikan tepian objek (edge enhancement).
2. Restorasi citra (image restoration), contohnya penghilangan kesamaran (deblurring).
3. Pemampatan citra (image compression).
4. Segmentasi citra (image segmentation).

5. Pengorakan citra (image analysis), contohnya pendeteksian tepi objek (edge enhancement) dan ekstraksi batas (boundary).
6. Rekonstruksi citra (image reconstruction).

## 2.2 Deteksi obyek dengan Computer Stereo Vision Epipolar geometri

*Computer Vision* adalah sebuah cabang ilmu dalam dunia komputer yang melingkupi metode untuk mentransformasi data dari gambar atau video 2D/3D menjadi keputusan-keputusan tertentu atau representasi lainnya. Data yang masuk bisa juga termasuk informasi kontekstual seperti “kamera terletak di atas mobil” atau “sensor jarak menunjukkan bahwa obyek terletak dalam jarak 1 meter”. Keputusan yang diambil bisa saja menunjukkan “terdapat orang di dalam gambar” atau “terdapat 14 sel tumor dalam gambar”. Representasi-representasi lain bisa saja berupa bentuk abu-abu dari gambar atau menghilangkan gerakan kamera pada urutan gambar. [1] Ini berarti data yang masuk pada *computer vision* tidak hanya berupa data dalam piksel yang diambil oleh kamera tapi juga data-data dari posisi kamera itu sendiri ataupun sensor-sensor yang mendukung fungsi kamera seperti sensor jarak dan lain-lain.

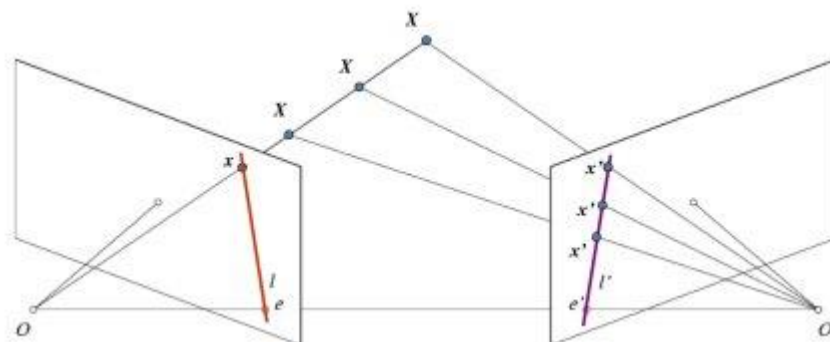
Dua gambar yang diambil dari perspektif yang berbeda dihubungkan oleh sesuatu yang disebut dengan epipolar geometri. Hubungan kedua gambar tersebut dapat digambarkan sebagai berikut, bila diambil titik sembarang  $x$  dari gambar pertama, bila titik tersebut merupakan proyeksi 3D titik  $X$  dari gambar, maka proyeksi  $x'$  berada pada sebuah garis yang ditentukan oleh posisi  $x$  yang disebut dengan garis epipolar. [2] Dari pengertian tersebut, maka epipolar geometri dapat dituliskan sebagai

$$x^T F x' = 0 \quad (2.1)$$

di mana  $F$  adalah matriks 3x3 yang disebut dengan matriks fundamental.



Cara umum yang sering dipakai untuk mendapatkan epipolar geometri dari dua gambar mencakup dua tahapan utama. Pada tahap pertama, dua titik fitur dideteksi di kedua gambar secara terpisah, kemudian dari dua titik yang ditemukan itu, dibuatlah korespondensi antar gambar yang kemudian dijadikan sebagai titik fitur baru. Algoritma deteksi fitur dan korespondensi antar gambar yang sering digunakan antara lain adalah Harris Corner detector dan juga Sift Infariant Feature Transform (SIFT). Pada tahap kedua, matriks fundamental ditentukan melalui hasil dari fitur-fitur yang berkorespondensi. Ini biasanya diawali dengan solusi linear yang kemudian dioptimasi dengan optimasi non-linear (misalnya, LMedS). Kebanyakan metode untuk tahap ini dapat digambarkan sebagai Maximum Likelihood Estimation (MSE), dan kualitas dari estimasinya bergantung pada akurasi dari korespondensi fitur. [2]



Gambar 2.5. Pengaturan dasar dua kamera untuk mengambil gambar

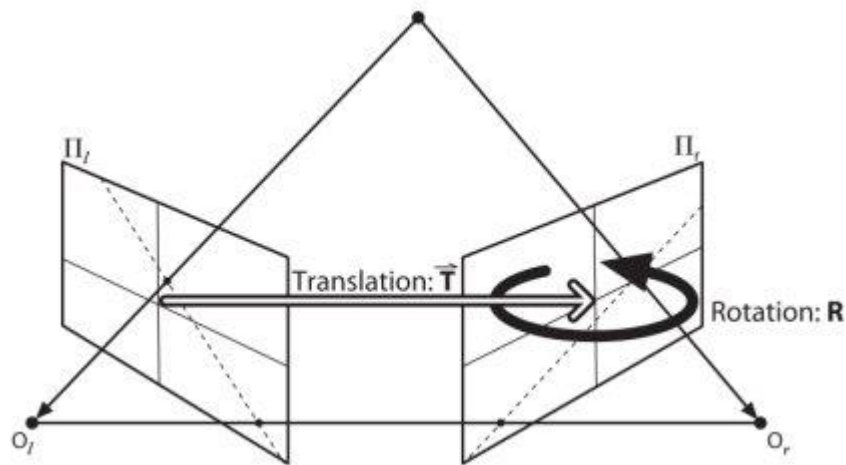
Di atas terdapat gambar pengaturan dasar dari kamera untuk mengambil dua gambar dari perspektif yang berbeda dari pemandangan yang sama. Jika kita hanya menggunakan kamera kiri, kita tidak bisa menemukan titik 3D sesuai dengan titik  $x$  dalam gambar karena setiap titik terproyeksi pada jalur  $OX$  ke titik yang sama pada bidang gambar. Tapi, mempertimbangkan hasil dari gambar kamera kanan juga. Sekarang, titik yang lain pada garis  $OX$  terproyeksi ke titik yang berbeda ( $x'$ ) dalam bidang sebelah. Jadi dengan dua gambar tersebut, kita bisa mentriangulasi titik 3D yang benar. Ini adalah konsep dari epipolar geometri.

Proyeksi titik-titik yang berbeda pada  $OX$  membentuk garis pada bidang sebelah kanan (garis  $l'$ ). Titik-titik ini disebut epiline berdasarkan titik  $x$ . Artinya, untuk menemukan titik  $x$  pada gambar kanan, cari sepanjang epiline ini. Titik tersebut harus berada di suatu tempat di baris tersebut. Ini disebut batas epipolar. Demikian pula semua titik akan memiliki epiline yang sesuai di gambar lainnya. Bidang  $XOO'$  disebut bidang epipolar.

$O$  dan  $O'$  adalah pusat dari kamera. Dari pengaturan yang diberikan di atas, kita dapat melihat bahwa proyeksi kamera yang tepat  $O$  terlihat pada gambar sebelah kiri pada titik  $e$ . Hal ini disebut dengan epipole. Epipole adalah titik perpotongan garis melalui pusat kamera dan bidang gambar. Demikian pula  $e'$  adalah epipole kamera kiri. Dalam beberapa kasus, kita tidak dapat menemukan epipole dalam gambar, karena bisanya epipole berada di luar gambar.

Semua epiline pasti melewati epipolenya. Jadi untuk menemukan lokasi epipole, kita dapat menemukan banyak epilines dan menemukan titik persimpangannya.

Maka untuk mendapatkan kedalaman gambar, kita harus fokus pada menemukan garis epipolar dan epipole. Tetapi untuk menemukan garis epipolar dan epipole-nya, kita perlu dua hal lain, yaitu Matriks Fundamental ( $F$ ) dan Matriks Esensial ( $E$ ). Matriks esensial berisi informasi tentang translasi dan rotasi, yang menggambarkan lokasi kamera kedua relatif terhadap kamera pertama di koordinat global.



Gambar 2.6. Transalasi dan rotasi kamera 2 terhadap kamera 1

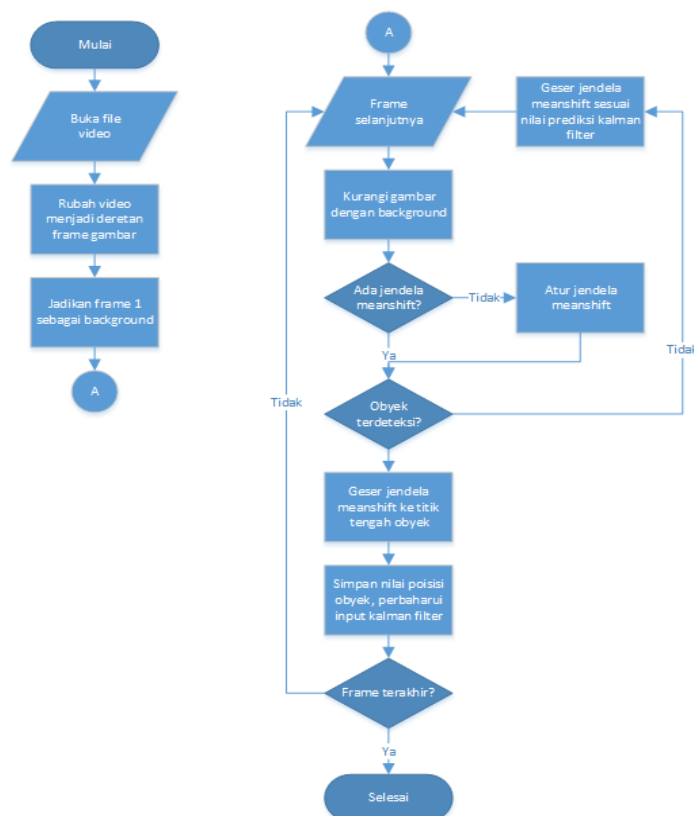
Namun, pada kenyataannya, pengukuran yang akan kita lakukan dilakukan dalam koordinat piksel. Matriks fundamental berisi informasi yang sama dengan matriks essential ditambah dengan informasi intrinsik dari kedua kamera sehingga kita dapat berhubungan dengan kedua kamera dalam koordinat piksel. Sederhananya, Matriks fundamental  $F$ , menghubungkan sebuah titik pada gambar ke sebuah garis di gambar lainnya. Ini dapat dikalkulasi melalui titik-titik yang saling berkorespondensi dari kedua gambar. [3]

### 2.3 Deteksi obyek pada gambar dua dimensi

Pada *epipolar geometry stereo vision*, salah satu hal yang paling penting adalah kemampuan sistem untuk mendapatkan fitur-fitur yang dapat dikorespondensikan. Ini artinya algoritma untuk mendeteksi obyek pada gambar dua dimensi di masing-masing kamera harus membaca fitur yang sama pada titik yang sama. Untuk rekonstruksi gambar tiga dimensi secara penuh, proses ini akan menjadi proses yang cukup berat untuk prosesor karena prosesor harus memproses fitur-fitur di tiap titik yang ditemukan untuk dapat merekonstruksi kedalaman dari gambar.

Namun, pada perancangan sistem ini, sistem dibuat sehingga tidak perlu memproses tiap titik fitur yang berkorespondensi. Dalam sistem ini, obyek dianggap sebagai satu titik yang dilihat dari dua prespektif gambar yang berbeda untuk dapat menemukan posisi obyek di ruang tiga dimensi.

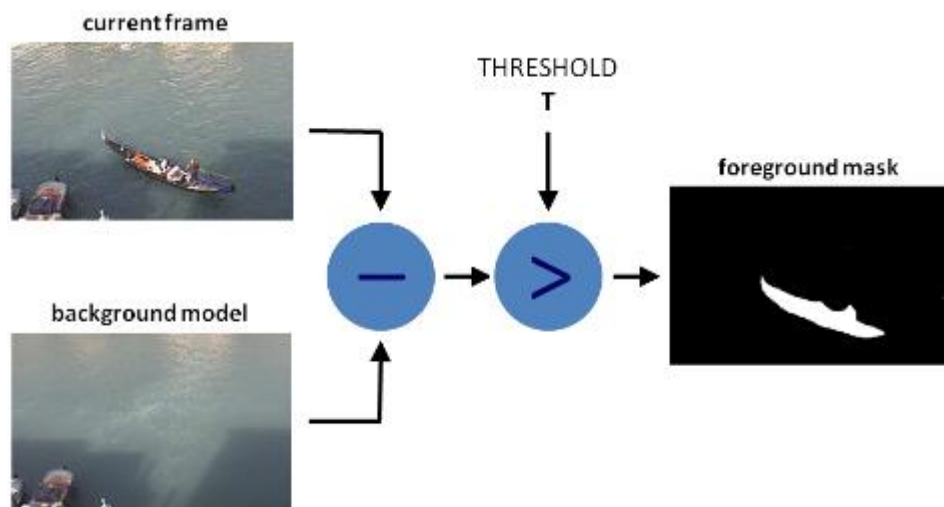
Obyek yang akan dideteksi oleh sistem ini, yaitu *shuttlecock*, adalah obyek yang cukup sulit untuk dideteksi karena warnanya yang putih dan pergerakannya yang cukup cepat. Karena itu, sistem ini akan didesain untuk memanfaatkan beberapa algoritma untuk mendapatkan posisi *shuttlecock* dalam bidang dua dimensi di masing-masing kamera. Ada dua hal penting yang dapat dimanfaatkan dalam pendeteksian obyek *shuttlecock*, yaitu obyek merupakan benda yang bergerak, dan latar dapat diasumsikan tidak bergerak. Maka, algoritmanya dapat dilihat pada gambar 2.7, sebagai berikut:



Gambar 2.7 Diagram algoritma deteksi pada bidang dua dimensi

### 2.3.1 Background Substraction

Seperti dapat dilihat di atas, algoritma pendeteksian obyek *shuttlecock* dapat dibagi menjadi beberapa tahap. Diawali dengan mengambil gambar background. Gambar background adalah gambar di mana tidak ditemukan obyek dalam gambar dan gambar merupakan latar dari obyek nantinya yang tidak bergerak. Hal ini penting untuk tahap selanjutnya nanti. Tahap kedua adalah tahap pengambilan gambar dengan obyek di dalamnya. Dari sini kita masuk ke tahap ketiga, tahap awal pendeteksian obyek bergerak. Skema paling sederhana untuk mendeteksi benda bergerak dalam sebuah urutan gambar adalah dengan cara menggunakan latar belakang yang tetap untuk mengurangi gambar selanjutnya dengan obyek bergerak. Gambar yang kemudian didapatkan kemudian dapat dianalisis untuk mendapatkan obyek yang dicari. [4]



Gambar 2.8 Skema background subtraction

Ini artinya background subtraction memiliki dua tahap utama, yang pertama adalah inisiasi gambar. Hal ini telah dilakukan pada tahap pertama dari algoritma sistem ini. Tahap selanjutnya adalah menangkap gambar terkini untuk menganalisis perubahan pada gambar dari gambar awal hingga gambar terakhir. [5]

### 2.3.2 Continously Adaptive Meanshift

Namun, subtraksi background saja tidaklah cukup, dibutuhkan suatu metode untuk memisahkan obyek yang akan dideteksi, dalam hal ini *shuttlecock* dengan lingkungan sekitarnya. Hal ini perlu dilakukan karena bagaimana pun lingkungan diatur untuk membedakan background dengan obyek, gangguan tetap akan terjadi.

Karena itu lah, dalam sistem ini digunakan algoritma *camshift* (*continously adaptive meanshift*). Algoritma ini dipilih karena kemampuannya untuk menentukan dan memperbaharui jendela deteksi obyek dan memperbaharui ukuran, posisi, dan bentuk jendela. Hal ini memungkinkan sistem untuk mengabaikan gangguan yang berada di luar jendela deteksi.

Masukan dari algoritma camshift berupa histogram yang menggambarkan suatu nilai. Dalam sistem ini, masukan yang digunakan adalah nilai Hue dari nilai HSV (*Hue, Saturation, Value*) gambar. Ruang warna HSV dipilih karena dianggap mampu mengekspresikan warna dengan lebih akurat.

Sistem akan melakukan ekstraksi komponen H pada gambar, lalu komponen ini akan dibagi menjadi pembagian  $m$  dengan setiap nilai berkorespondensi dengan sebuah nilai sub-karakteristik. Dengan ini, seluruh area target dapat dikarakterisasi dengan nilai ini. Pada setiap sub-fitur, fungsi distribusi densitas berbasis kernel digunakan untuk mengkalkulasi probabilitas distribusi. Lalu, karakteristik probabilitas ke- $u$  dapat dituliskan sebagai berikut:

$$q_u = C \sum_{i=1}^n k \left( \left\| \frac{x_0 - x_i}{h} \right\|^2 \right) \delta[b(x_i) - u] \quad (2.1)$$

di mana  $x_0$  adalah titik tengah dari jendela pencarian,  $x_i$  adalah posisi ke- $i$  dari piksel,  $k(\|x\|^2)$  adalah fungsi kernel,  $h$  adalah radius jendela,  $b$  dan fungsi  $\delta$  menentukan apakah piksel pada titik  $x_i$  sesuai dengan nilai karakteristik ke- $u$ . Konstan  $C$  untuk normalisasi siturunkan dari kondisi  $\sum_{u=1}^m \hat{q}_u = 1$ , di mana:

$$C = \frac{1}{\sum_{i=1}^n k\left(\left\|\frac{x_0 - x_i}{h}\right\|^2\right)} \quad (2.2)$$

karena penjumlahan dari fungsi delta untuk  $u = 1 \dots m$  sama dengan satu.

formula (2.3.2.1), asumsikan  $y_0$  sebagai titik tengah dari kandidat area target pada frame saat ini, maka probabilitas ke- $u$  dari nilai karakteristik adalah:

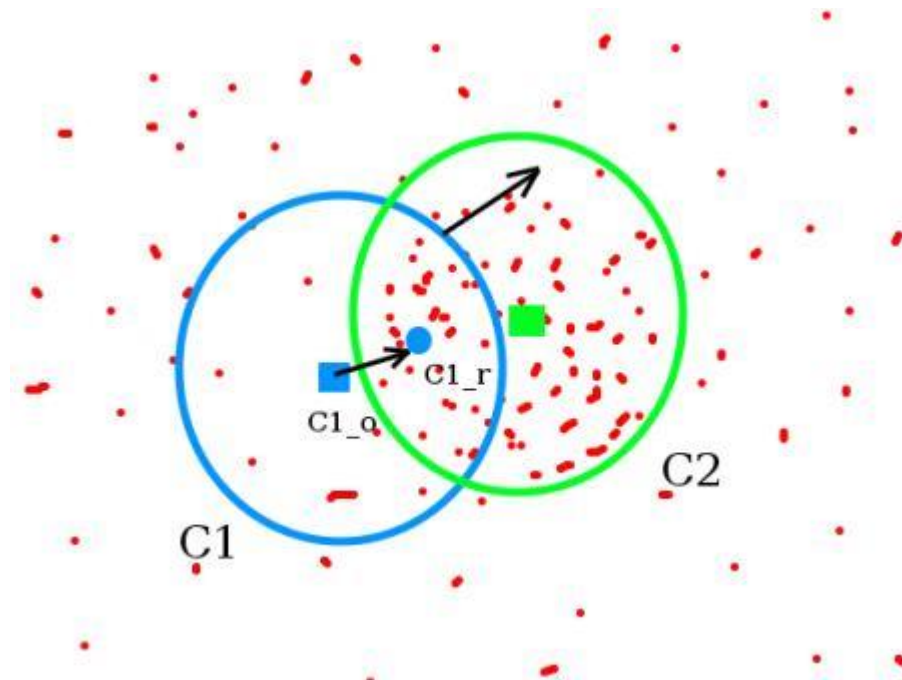
$$p_u(y_0) = C \sum_{i=1}^n k\left(\left\|\frac{y_0 - x_i}{h}\right\|^2\right) \delta[b(x_i) - u] \quad (2.3)$$

Fungsi similaritas menentukan kesamaan antara model target awal dengan target kandidat, metode pengukuran similaritas biasanya termasuk menggunakan koefisien *Bhattacharyya*, pengukuran informasi *Fisher*, dan teknik perpotongan histogram. Dalam sistem ini, metode meanshift yang digunakan adalah dengan menentukan perpotongan histogram.

Algoritma camshift adalah hasil dari penggunaan algoritma meanshift pada urutan gambar kontinyu di mana hasil dari frame sebelumnya digunakan sebagai nilai awal jendela pencarian pada frame selanjutnya [6]. Bila iterasi ini berlanjut, maka deteksi obyek bergerak dapat dilakukan. Adapun langkah-langkahnya adalah sebagai berikut:

1. Atur gambar sebagai area pencarian
2. Inisialisasi ukuran dan posisi jendela pencarian
3. Kalkulasi distribusi probabilitas warna dalam jendela pencarian
4. Jalankan meanshift untuk mendapatkan posisi dan ukuran jendela baru
5. Pada frame selanjutnya, inisiasi posisi dan ukuran jendela pencarian berdasarkan langkah 4. Kembali ke langkah 3.





Gambar 2.9 Skema meanshift

### 2.3.3 Kalman Filter

Algoritma Kalman filter di sini adalah algoritma yang digunakan untuk memprediksi lokasi obyek yang paling mungkin pada frame saat ini berdasarkan hasil dari pencarian obyek pada frame-frame sebelumnya, algoritma ini lalu mencari obyek pencarian pada area di sekitar lokasi. Bila target ditemukan pada area pencarian, lanjutkan proses ke frame selanjutnya. Kunci dari Kalman filter adalah prediksi dan perbaruan nilai [6].

Kita dapat mendefinisikan vektor keadaan  $X_k = [x, y, v_x, v_y]$ , vektor pengukuran  $Z_k = [x, y]^T$ , di mana  $x$  dan  $v_x$  adalah gambar target pada arah horizontal dari posisi dan kecepatan, sedangkan  $y$  dan  $v_y$  merupakan gambar target pada arah vertikal dari posisi dan kecepatan. Dari situ, maka model Kalman filternya adalah sebagai berikut:

$$\text{Persamaan gerakan: } X_k = F \cdot X_{k-1} + W_k \quad (2.4)$$

$$\text{Persamaan observasi: } Z_k = H \cdot X_k + V_k \quad (2.5)$$

di mana  $W_k$  dan  $V_k$  adalah pergerakan dan pengukuran vektor gangguan yang memenuhi distribusi Gaussian  $p(w) \sim N(0, Q)$ ,  $p(v) \sim N(0, R)$ ,  $F$  adalah matriks transisi kondisi dan  $H$  adalah matriks pengukuran.

Sedangkan untuk persamaan prediksi dan persamaan perbaruan nilai dapat dituliskan sebagai berikut:

$$\text{Persamaan prediksi 1: } X_k' = F \cdot X_{k-1} \quad (2.6)$$

$$\text{Persamaan prediksi 2: } P_k' = F \cdot P_{k-1} \cdot F^T + Q \quad (2.7)$$

$$\text{Persamaan Kalman-gain: } K_k' = P_k' \cdot H^T \cdot (H \cdot P_k' \cdot H^T + R)^{-1} \quad (2.8)$$

$$\text{Persamaan perbaruan 1: } X_k = X_k' + K_k \cdot (Z_k - H \cdot X_k') \quad (2.9)$$

$$\text{Persamaan perbaruan 2: } P_k = P_k' - K_k \cdot H \cdot P_k' \quad (2.10)$$

Nilai dari matriks transisi kondisi  $F$ , matriks pengukuran  $H$ , matriks kovarians gangguan proses  $Q$ , dan matriks kovarians gangguan pengukuran  $R$  dapat dinyatakan sebagai berikut:

$$F = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, Q = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix},$$

$$H = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}, R = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

## 2.4 OpenCV sebagai Computer Vision Library

Sistem ini menggunakan *OpenCV Library*. *OpenCV* adalah sebuah *Computer Vision Library* yang dulunya dikembangkan oleh Intel dan sekarang didukung oleh Willow Garage yang bermain pada bidang real-time image processing. *OpenCV* bersifat open-source dan dapat diakses secara bebas di <http://opencv.org>. *Library* ini ditulis dalam bahasa C dan C++ dan dapat dijalankan di bawah Linux, Windows, Mac OS X, iOS, dan Android. Bahasa pemrograman lain yang dapat digunakan antara lain Python, Java, Ruby, Matlab, dan berbagai bahasa lainnya yang saat ini masih dalam pengembangan. [1]



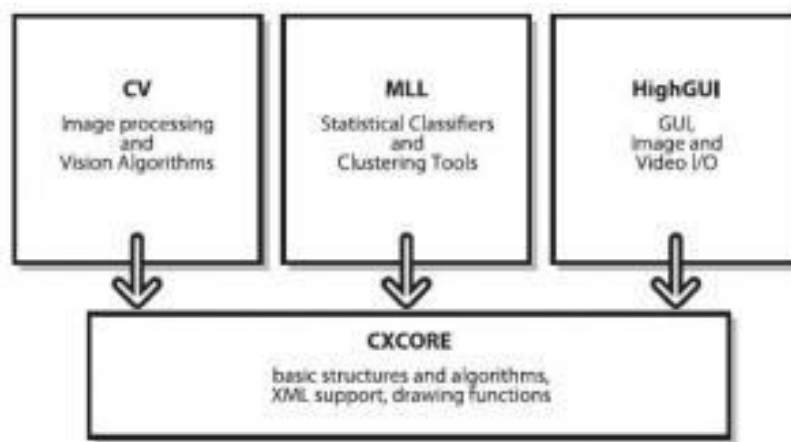
Gambar 2.10. Lambang OpenCV

OpenCV dirilis di bawah lisensi BSD dan karena OpenCV itu gratis baik untuk penggunaan akademis dan komersial. Ditulis dalam bahasa C/C++ yang dioptimalkan, library ini dapat memanfaatkan multi-core processing. Diaktifkan dengan OpenCL, program yang dibuat dapat mengambil keuntungan dari akselerasi hardware yang mendukung pemrosesan dengan platform heterogen. Diadopsi di seluruh dunia, OpenCV memiliki lebih dari 47 ribu orang dari komunitas pengguna dan diperkirakan jumlah unduhan melebihi 9 juta. Rentang penggunaan dari seni interaktif, untuk inspeksi tambang, aplikasi di web atau pun dalam dunia robotika.

OpenCV sendiri terdiri dari 5 library, yaitu :

- *CV* : untuk algoritma Image processing dan Vision.
- *ML* : untuk machine learning library
- *Highgui* : untuk GUI, Image dan Video I/O.
- *CXCORE* : untuk struktur data, support XML dan fungsi-fungsi grafis.
- *CvAux*

Struktur dan konten OpenCV dapat dilihat pada Gambar 2.11 berikut:



Gambar 2.11. Struktur dan konten OpenCV

## 2.5 Simulasi menggunakan rviz pada ROS (Robot Operating System)

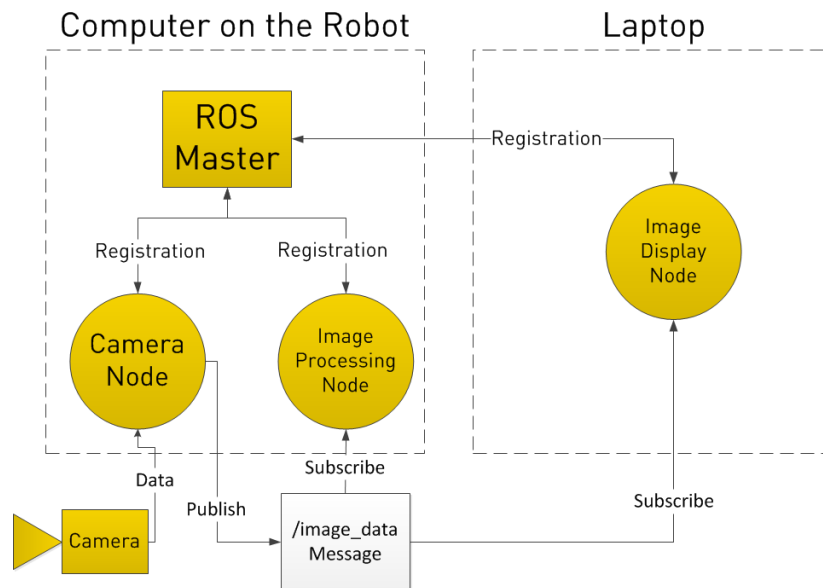
Robot Operating System (ROS) adalah kumpulan dari software frameworks untuk pengembangan software robot, memberikan fungsi menyerupai sistem operasi pada berbagai sistem komputer yang heterogen. ROS menyediakan fitur standar sistem operasi seperti abstraksi hardware, kendali devais secara low-level, implementasi dari fungsi-fungsi umum, pengiriman pesan antar proses, dan manajemen paket [7].



Gambar 2.12. Lambang Robot Operating System

Ke depannya, sistem ini diharapkan dapat dengan mudah diimplementasikan ataupun dikembangkan oleh pihak-pihak lain. Karena itu lah, dipilih framework ROS sebagai framework utama, hal ini bertujuan agar semua program yang dibuat dalam sistem ini menganut pada satu konvensi yang mudah digunakan kembali oleh pihak-pihak lain tanpa memperhatikan arsitektur yang berbeda dari sistem.

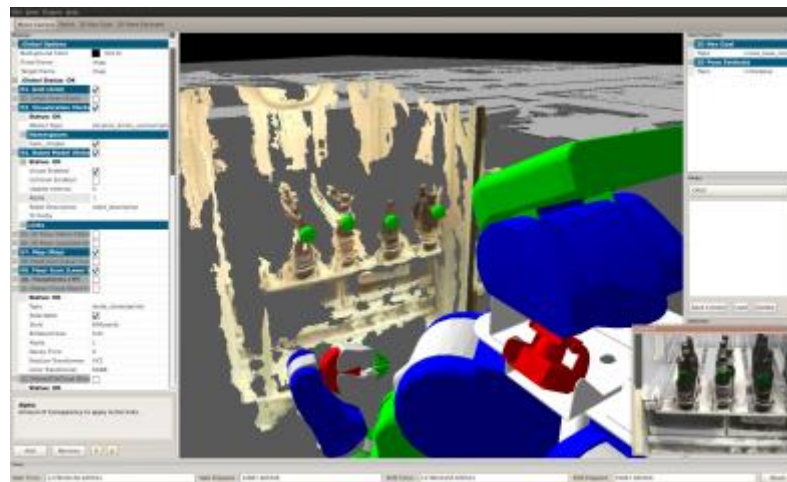
Sesuai namanya, ROS memang lebih banyak digunakan dalam dunia robotika, namun peralatan (*tool*), *library*, *driver*, dan konvensi yang digunakan memudahkan pembuatan program yang kompleks pada berbagai platform. Dalam ROS dikenal terminologi *topics*, *messages*, dan *nodes*. Sistem apa pun yang diprogram, apakah itu robot humanoid, robot beroda, ataupun pesawat, programmer hanya harus memperhatikan *topics* mana yang dipakai untuk mengirimkan perintah, tipe data *message* yang digunakan, dan *nodes* mana yang akan dijalankan. Kesamaan terminologi programming ini tentu akan memudahkan programmer untuk memprogram berbagai jenis sistem serta untuk berbagi dengan programmer-programmer lain dari berbagai belahan dunia.



Gambar 2.13. Sistem node dalam ROS

### 2.5.1 Simulasi RVIZ

RVIZ adalah salah satu *package* yang disediakan di dalam ROS yang digunakan untuk memvisualisasikan robot, *point clouds*, dan lain sebagainya [7].



Gambar 2.14. Tampilan RVIZ

Sebagai sebuah package yang digunakan untuk visualisasi, RVIZ sendiri memiliki berbagai kelebihan, yaitu:

1. RVIZ sangat cocok digunakan untuk melakukan *troubleshooting* pada sistem penginderaan. Terdapat pengaturan informasi visual yang dapat langsung ditampilkan atau disembunyikan.
2. RVIZ menyediakan visualisasi 3D yang dapat dinavigasikan secara bebas dengan menggunakan mouse.
3. RVIZ memiliki fitur *interactive marker*. Fitur ini mempermudah pengguna untuk melakukan penandaan area ataupun ruang dalam ruang 3D. Bahkan, pengguna bisa mengatur pengindraannya secara manual pada saat program masih berjalan. Contohnya, memilih area tertentu dalam ruang 3D dan mengabaikan area di luar area yang dipilih pengguna.
4. Dengan RVIZ, pengguna dapat menggunakan beberapa proses penginderaan dalam waktu bersamaan sehingga pengguna dapat menampilkan data dari beberapa devais ataupun file secara bersamaan.

## **2.6 Hardware**

### **2.6.1 Kamera PS3 Eye**

PlayStation Eye (merek dagang PLAYSTATION Eye) adalah perangkat kamera digital, mirip dengan webcam, untuk PlayStation 3. Teknologi ini menggunakan computer vision dan gesture recognition untuk memproses gambar yang diambil oleh kamera. Hal ini memungkinkan pemain untuk berinteraksi dengan game menggunakan gerakan dan deteksi warna serta suara melalui mikrofon built-in. Ini adalah penerus EyeToy untuk PlayStation 2, yang dirilis pada tahun 2003.





Gambar 2.15. Playstation 3 Eye yang digunakan sebagai kamera sistem

Kamera ini diluncurkan pertama kali bersamaan dengan game The Eye of Judgement di Amerika Serikat pada tanggal 23 Oktober 2007, di Jepang dan Australia pada 25 Oktober 2007 dan di Eropa pada tanggal 26 Oktober 2007.

PlayStation Eye juga dirilis sebagai produk yang berdiri sendiri di Amerika Serikat, Eropa, dan Australia. Desainer EyeToy Richard Marks menyatakan bahwa EyeToy digunakan sebagai model untuk desain awalnya. Pada tahun 2013 Sony mengumumkan PlayStation Eye akan diganti dengan PlayStation Camera untuk konsol PlayStation 4 untuk bersaing dengan Microsoft Corporation Kinect dan Nintendo Wii remote Plus.

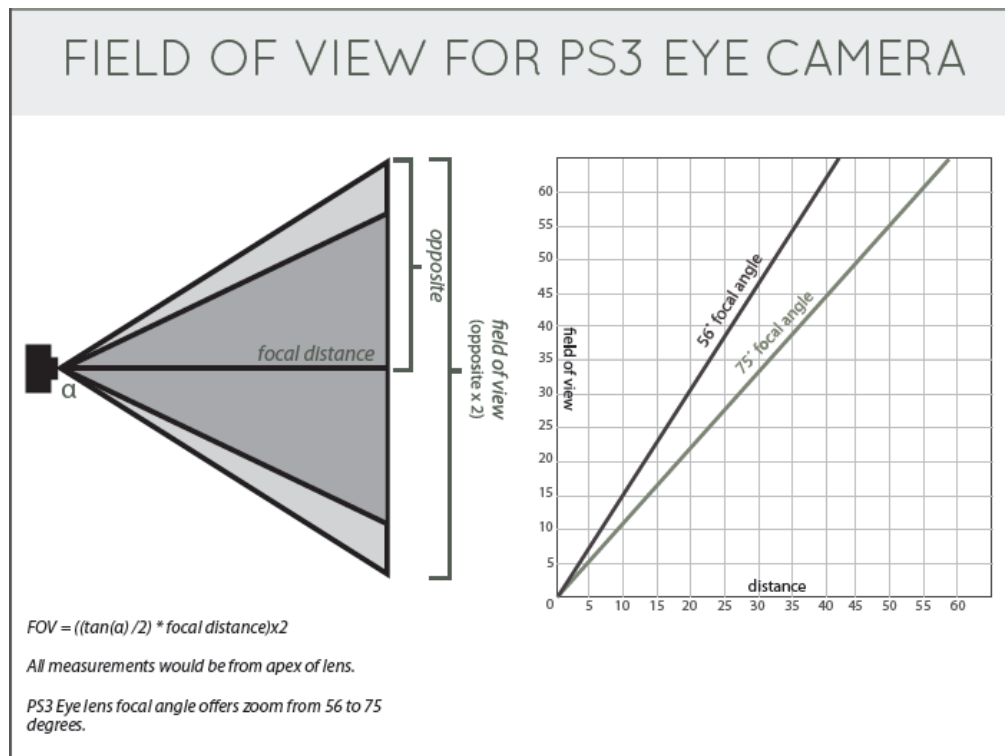
PlayStation Eye mampu menangkap video standar dengan frame rate 60 hertz pada resolusi  $640 \times 480$  pixel, dan 120 hertz pada  $320 \times 240$  piksel, yang merupakan empat kali resolusi dan dua kali frame rate dari EyeToy, menurut Sony, frame rate yang lebih tinggi, hingga  $320 \times 240$  pada 187 atau  $640 \times 480$  pada 75 fps, dapat dipilih dengan aplikasi khusus (Freetrack dan Linuxtrack).

PlayStation Eye juga memiliki dua kali sensitivitas dari EyeToy tersebut, Sony bekerja sama dengan perusahaan chip sensor OmniVision Technologies pada desain chip sensor menggunakan piksel sensor yang lebih besar, memungkinkan untuk operasi cahaya rendah yang lebih efektif. Sony menyatakan bahwa PlayStation Eye dapat menghasilkan kualitas video yang mumpuni di bawah pencahayaan yang disediakan oleh televisi.

Kamera ini memiliki dua pengaturan fokus tetap dengan lensa zoom yang disesuaikan. Dipilih secara manual dengan memutar lensa barel, PlayStation Eye dapat diatur ke sudut pandang  $56^\circ$  (red dot) mirip dengan EyeToy untuk close-up framing dalam aplikasi chatting, atau sudut pandang  $75^\circ$  (blue dot) untuk menangkap sudut pandang yang digunakan dalam aplikasi game interaktif.

Playstation Eye mampu memberikan keluaran video ke konsol tanpa terkompresi atau dengan kompresi optimal JPEG. Adapun kedalaman warnanya yaitu sebesar 8 bit per piksel. [6]

Playstation Eye dipilih sebagai kamera untuk sistem ini karena frame-rate nya yang cukup tinggi sehingga bisa mendapatkan detail perubahan untuk gambar yang akan di analisis dengan lebih baik. Selain itu, karena di desain untuk mendeteksi gesture, lensanya sudah cukup baik dalam menangkap gambar sehingga kalibrasi intrinsik kamera yang dibutuhkan sangat minimal. Sistem mekanik yang bebas memungkinkan kita untuk mengatur sudut kamera secara bebas, hal ini sangat penting untuk menentukan nilai-nilai dalam epipolar geometri. [7]



Gambar 2.16. Field of View dari kamera Playstation Eye

Tabel 2.6.1.1. Field of View Playstation Eye

angle	$\tan(\alpha)$	focal distance	opposite	field of view
56°	1.4823	1 inch	.53 inches	1.06 inches
75°	3.7306	1 inch	.77 inches	1.54 inches
56°	1.4823	2 inches	1.06 inches	2.12 inches
75°	3.7306	2 inches	1.53 inches	3.06 inches
56°	1.4823	4 inches	2.13 inches	4.26 inches
75°	3.7306	4 inches	3.07 inches	6.14 inches
56°	1.4823	8 inches	4.25 inches	8.50 inches
75°	3.7306	8 inches	6.14 inches	12.28 inches
56°	1.4823	16 inches	8.51 inches	17.02 inches
75°	3.7306	16 inches	12.28 inches	24.58 inches
56°	1.4823	24 inches	12.76 inches	25.52 inches
75°	3.7306	24 inches	18.42 inches	36.84 inches
56°	1.4823	36 inches	19.14 inches	38.28 inches
75°	3.7306	36 inches	27.62 inches	55.24 inches

## BAB 3

### METODE PENELITIAN

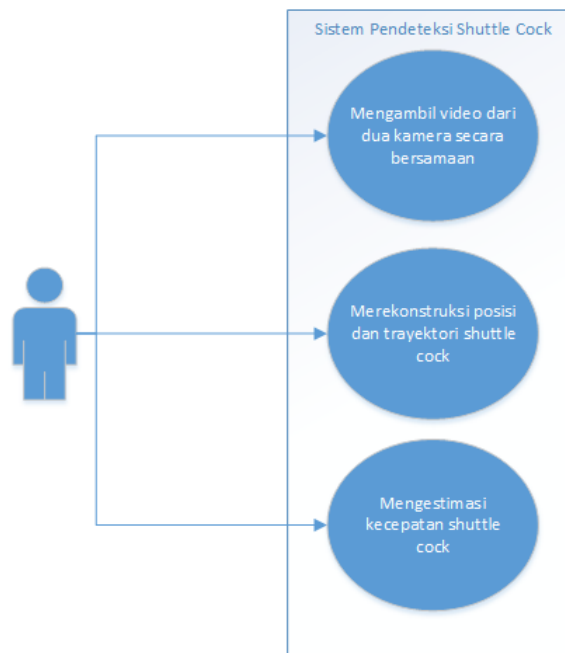
Pada bab ini akan dibahas mengenai perancangan sistem berkaitan dengan sistem yang akan dibangun untuk mendeteksi gerakan *shuttlecock*. Berdasarkan *Software Development Life Cycle* (SLDC), setelah menentukan tema dan batasan masalah serta tujuan, maka tahapan berikutnya dalam pembuatan sistem pendeteksian *shuttlecock* adalah perencanaan. Tahapan perencanaan ini meliputi *system requirement*, desain, dan implementasi. Pengujian sistem dan analisa tidak akan dibahas pada bab ini. Dalam mendokumentasikan setiap tahapan SLDC, *Unified Modeling Language* (UML) akan digunakan sebagai metode standar. Dengan UML, rancangan perangkat keras serta alur kerja dapat direpresentasikan ke dalam diagram-diagram yang memiliki fungsi masing-masing.

#### 3.1 System Requirement

*System Requirement* merupakan tahapan yang mendefinisikan sistem dan fitur yang dibutuhkan. Tahapan ini memegang peranan penting dalam perancangan fungsionalitas sistem yang akan dibuat. Terdapat sebuah *shuttlecock* yang akan dideteksi posisinya oleh shuttle.

Di bawah ini adalah hasil pengumpulan *requirement* yang didapat melalui studi literatur:

- Dua kamera diposisikan di luar lapangan di dua ujung yang berbeda di mana setiap kamera dapat menangkap seluruh lapangan dalam Field of View-nya.
- Dua kamera di arahkan ke arah background yang sebisa mungkin memiliki warna yang berbeda dengan *shuttlecock* yang digunakan
- Dua kamera mengambil gambar secara bersamaan dan merekamnya dalam bentuk file video

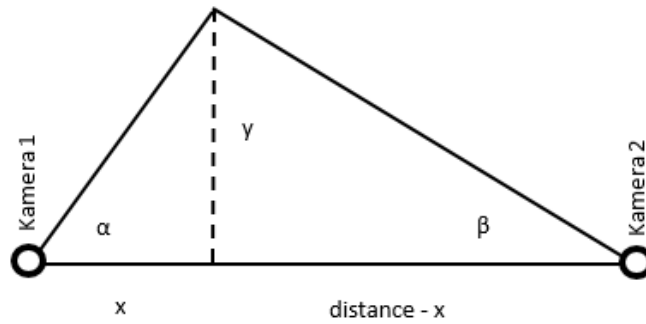


Gambar 3.1. Use Case Diagram dari Sistem

Fungsi-fungsi di atas merupakan fungsi-fungsi yang harus ada pada sistem ini guna memenuhi kebutuhan pengguna yang akan menggunakannya. *Requirement* di atas merupakan *requirement* dasar yang masih dapat dikembangkan lagi sehingga menjadi sebuah sistem yang lebih baik. Gambar di atas menggambarkan fungsi yang sudah diolah dalam *use case* diagram.

### 3.2 System Modelling

Dalam pembuatan sistem ini, diperlukan sebuah model untuk mendapatkan hasil posisi obyek dalam ruang tiga dimensi. Gambar 3.2 merupakan permodelan yang dibuat untuk sistem ini.



Gambar 3.2. Model sistem dalam bidang x dan y

Dari model tersebut dapat diturunkan persamaan sebagai berikut:

$$\tan \alpha \times x = \tan \beta \times (distance - x)$$

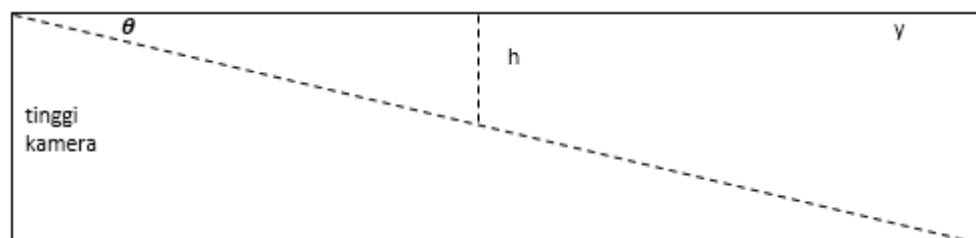
$$\tan \alpha \times x = (\tan \beta \times distance) - (\tan \beta \times x)$$

$$\tan \beta \times distance = (\tan \alpha \times x) - (\tan \beta \times x)$$

$$x = \frac{\tan \beta \times distance}{\tan \alpha + \tan \beta} \quad (3.1)$$

$$r = \sqrt{x^2 + y^2} \quad (3.2)$$

Sedangkan untuk bidang z dapat dibuat permodelan seperti pada gambar 3.3 sebagai berikut:



Gambar 3.3. Model sistem dalam bidang z

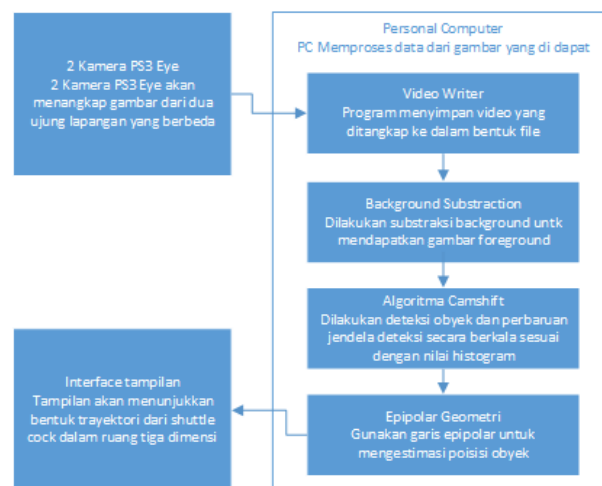
$$\tan \theta = \frac{h}{r}$$

$$h = r \times \tan \theta \quad (3.3)$$

$$z = \text{tinggi kamera} \pm h \quad (3.4)$$

### 3.3 System and Software Design

Sistem pengenalan pendeteksi posisi *shuttlecock* yang dibuat dalam penelitian ini dirancang untuk memberikan hasil yang akurat dari posisi *shuttlecock*, trayektori serta kecepatannya. Sistem ini akan dikendalikan oleh sebuah PC. Sedangkan untuk input gambar yang akan digunakan pada sistem akan didapatkan dari dua kamera Playstation Eye yang ditempatkan di kedua ujung lapangan dan menangkap seluruh area pergerakan *shuttlecock*. Gambar 3.4 menunjukkan diagram blok dari keseluruhan sistem:



Gambar 3.4. Diagram Blok

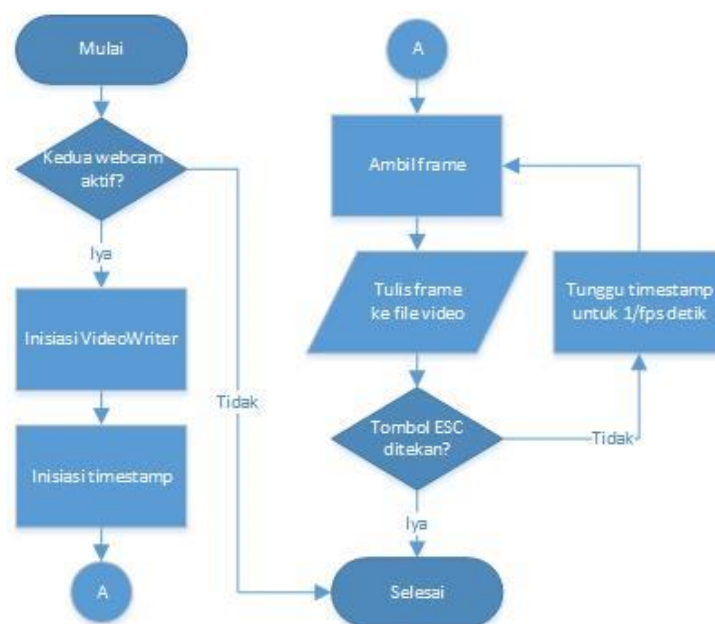
Adapun pengambilan data sistem terdiri dari dua tahap, yaitu tahap pengambilan video dan tahap pemrosesan video menjadi data pergerakan *shuttlecock*.

### 3.3.1 Desain Perangkat Keras (Hardware)

Sistem yang digunakan terdiri dari dua buah kamera dan sebuah PC. Dua kamera diposisikan di luar lapangan di dua ujung yang berbeda di mana setiap kamera dapat menangkap seluruh lapangan dalam Field of View-nya. Dua buah kamera ini juga di arahkan ke latar yang sebisa mungkin memiliki warna yang berbeda dari *shuttlecock* untuk mengurangi gangguan-gangguan yang tidak diinginkan.

### 3.3.2 Algoritma Perangkat Lunak (Software)

Algoritma yang dipakai pada sistem ini dapat dibagi menjadi dua bagian utama, yaitu bagian pengambilan data video, serta bagian pengolahan data.



Gambar 3.5. Flowchart program perekam video



Gambar di atas menunjukkan diagram alur untuk program perekam video. Program ini memanfaatkan fungsi *videowriter* yang disediakan oleh OpenCV. Namun, fungsi ini memiliki satu kelemahan, yaitu pengaturan untuk fps (*frame per second*) tidak memperhitungkan waktu proses sehingga hasil video tidak akurat. Karena itu lah diperlukan satu fungsi timestamp untuk melakukan pengecekan di setiap iterasi sehingga waktu iterasi memiliki periode yang sesuai dengan fps video. Gambar 3.6 merupakan pseudocode dari program pengambil video dalam ROS:

```

ros::init (argc, argv, "doublerecord");
ros::NodeHandle nh;

int fps = 25;

VideoCapture cap1(1);
if(!cap1.isOpened()){
    cout << "Error opening video stream 1" << endl;
}
VideoCapture cap2(2);
if(!cap2.isOpened()){
    cout << "Error opening video stream 2" << endl;
}

int frame_width_1= cap1.get(CV_CAP_PROP_FRAME_WIDTH);
int frame_height_1= cap1.get(CV_CAP_PROP_FRAME_HEIGHT);
VideoWriter
video1("/home/deanzaka/datatemp/video1.avi",CV_FOURCC('M','J','P','G'),fps,
Size(frame_width_1,frame_height_1),true);

int frame_width_2= cap2.get(CV_CAP_PROP_FRAME_WIDTH);
int frame_height_2= cap2.get(CV_CAP_PROP_FRAME_HEIGHT);
VideoWriter
video2("/home/deanzaka/datatemp/video2.avi",CV_FOURCC('M','J','P','G'),fps,
Size(frame_width_2,frame_height_2),true);
while(nh.ok()){
    ros::Time start = ros::Time::now();
    Mat frame1, frame2;
    cap1 >> frame1;
    cap2 >> frame2;
    video1.write(frame1);
    video2.write(frame2);

```

```

imshow( "Frame 1", frame1 );
imshow( "Frame 2", frame2 );
char c = (char)waitKey(1);
if( c == 27 ) break;
float delay = ros::Time::now().toSec() - start.toSec();
float period = 1/fps;
while(delay < period) {
    delay = ros::Time::now().toSec() - start.toSec();
}
cout << "\n\n Delay = \t" << delay << "\n\n";
ros::spinOnce();
}

```

Gambar 3.6. Pseudocode program perekam video

Pada bagian pengolah data, digunakan serangkaian algoritma untuk mendapatkan posisi obyek beregerak, yang dalam hal ini adalah sebuah *shuttlecock*, dalam ruang tiga dimensi. Proses ini dapat dibagi menjadi tiga bagian utama bagian utama, yaitu background subtraction, camshift (continously adaptive meanshift) berbasis Kalman filter, dan yang terakhir adalah penentuan posisi benda melalui epipolar geometri.

Pada bagian pertama, background subtraction, kamera pertama-tama akan mengambil gambar background. Gambar background adalah gambar di mana tidak ditemukan obyek dalam gambar dan gambar merupakan latar dari obyek nantinya yang tidak bergerak. Tahap ini disebut dengan tahap preprocessing. Hal ini sangat bermanfaat itu tahap-tahap selanjutnya.

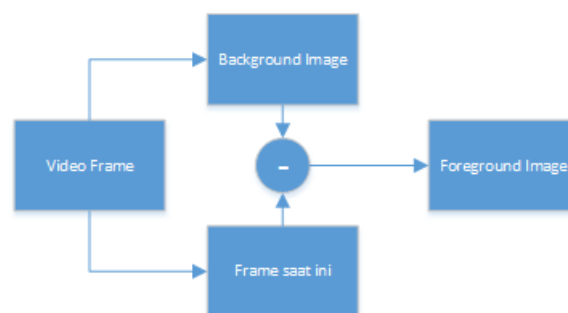
Bagian selanjutnya menggunakan algoritma camshift, pada tahap ini pengguna menentukan nilai histogram warna dengan cara mengatur jendela inisiasi. Nilai histogram ini kemudian digunakan untuk mencari obyek dengan nilai histogram yang sama, obyek yang dideteksi kemudian diambil titik tengahnya. Jendela meanshift kemudian akan bergeser sesuai dengan posisi dan ukuran dari obyek yang terdeteksi, jendela yang sudah berpindah ini kemudian akan digunakan pada frame selanjutnya, begitu seterusnya.

Adapun fungsi dari memasukkan algoritma Kalman filter ke dalam algoritma camshift adalah sebagai nilai prediksi bila pada saat pengambilan trayektori, terdapat frame-frame di mana tidak ditemukan obyek. Ini berguna agar laju trayektori tetap terpenuhi dan jendela pencarian camshift tetap bergerak sesuai dengan arah prediksi gerakan obyek, sehingga jendela tidak akan kehilangan obyek ketika obyek muncul kembali.

Tahap terakhir yaitu mengestimasi posisi dari obyek dengan menganggap obyek merupakan sebuah titik pada dua buah kamera dengan sudut pandang yang berbeda, sehingga dapat ditarik garis epipolar (epiline) untuk mendapatkan letak benda dalam ruang tiga dimensi relatif terhadap kamera. Dengan menambahkan nilai-nilai jarak kamera dari lapangan, maka kita dapat menentukan posisi *shuttlecock* relatif terhadap lapangan.

### 3.3.3 Background subtraction

Background subtraction mendapatkan gambar foreground yang merupakan obyek yang dicari dengan cara mengurangi gambar saat ini dengan gambar background. Sisanya berarti dapat dianggap sebagai obyek. Sistem ini, menggunakan fungsi `backgroundsubtractionMOG2` yang disediakan oleh OpenCV. Fungsi ini berdasarkan jurnal dari Z. Zivcovic yang berjudul “*Improved Adaptive Gaussian Mixture Model for Background Subtraction*”. [5]



Gambar 3.7. Algoritma Background Subtraction

Gambar 3.8 merupakan pseudocode untuk algoritma background subtraction:

```
pMOG1 = new BackgroundSubtractorMOG2(); //MOG2 approach
pMOG2->operator()(image2, fgMaskMOG2);
fgMaskMOG2.copyTo(channel2[2]);
channel2[1] = Mat::zeros(480, 640, CV_8UC1 );
channel2[0] = Mat::zeros(480, 640, CV_8UC1 );
merge(channel2, 3, image2);

//morphological opening (removes small objects from the foreground)
erode(image2, image2, getStructuringElement(MORPH_ELLIPSE, Size(8, 8)) );
dilate(image2, image2, getStructuringElement(MORPH_ELLIPSE, Size(8, 8)) );

//morphological closing (removes small holes from the foreground)
dilate(image2, image2, getStructuringElement(MORPH_ELLIPSE, Size(8, 8)) );
erode(image2, image2, getStructuringElement(MORPH_ELLIPSE, Size(8, 8)) );
```

Gambar 3.8. Pseudocode background subtraction

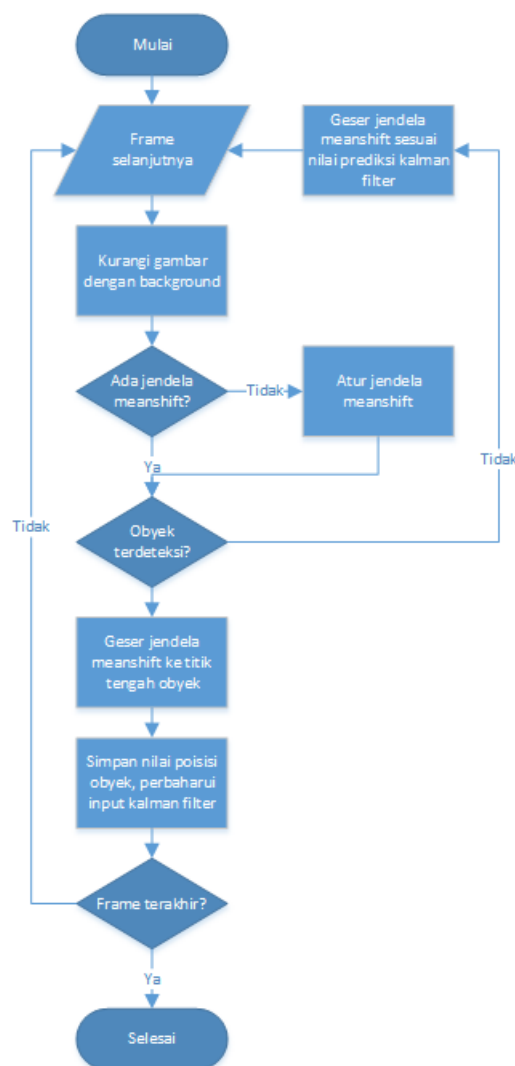
#### 3.3.4 Camshift berbasis Kalman filter

Langkah berikutnya setelah didapatkan foreground adalah menerapkan algoritma camshift berbasis Kalman filter. Algoritma ini dipilih karena kemampuannya untuk menentukan dan memperbaharui jendela deteksi obyek dan memperbaharui ukuran, posisi, dan bentuk jendela. Hal ini memungkinkan sistem untuk mengabaikan gangguan yang berada di luar jendela deteksi.

Masukan dari algoritma camshift berupa histogram yang menggambarkan suatu nilai. Dalam sistem ini, masukan yang digunakan adalah nilai Hue dari nilai HSV (*Hue, Saturation, Value*) gambar. Ruang warna HSV dipilih karena dianggap mampu mengekspresikan warna dengan lebih akurat.

Sedangkan algoritma Kalman filter di sini adalah algoritma yang digunakan untuk memprediksi lokasi obyek yang paling mungkin pada frame saat ini berdasarkan hasil dari pencarian obyek pada frame-frame sebelumnya, algoritma ini lalu mencari obyek pencarian pada area di sekitar lokasi. Bila target ditemukan

pada area pencarian, lanjutkan proses ke frame selanjutnya. Kunci dari Kalman filter adalah prediksi dan perbaruan nilai. Hal ini dibutuhkan karena pada deteksi benda yang bergerak cukup cepat terdapat frame-frame di mana obyek tidak terdeteksi, sehingga agar jendela pencarian tetap bergerak sesuai dengan pergerakan obyek, jendela pencarian diberikan nilai prediksi dari Kalman filter. Gambar 3.9 merupakan diagram yang menjelaskan cara kerja algoritma camshift berbasis Kalman filter:



Gambar 3.9. Algoritma camshift berbasis Kalman filter

Berikut pseudocode untuk algoritma camshift berbasis Kalman filter:

```
KalmanFilter KF1(4, 2, 0);

// initialization of KF
KF1.transitionMatrix = *(Mat_<float>(4, 4) << 1,0,1,0, 0,1,0,1, 0,0,1,0, 0,0,0,1);
Mat_<float> measurement1(2,1); measurement1.setTo(Scalar(0));

KF1.statePre.at<float>(0) = 0; // initial value
KF1.statePre.at<float>(1) = 0; // initial value
KF1.statePre.at<float>(2) = 0;
KF1.statePre.at<float>(3) = 0;
setIdentity(KF1.measurementMatrix);
setIdentity(KF1.processNoiseCov, Scalar::all(.005));
setIdentity(KF1.measurementNoiseCov, Scalar::all(1e-1));
setIdentity(KF1.errorCovPost, Scalar::all(.1));

while(nh.ok())
{
    ros::Time start = ros::Time::now();

    // First predict, to update the internal statePre variable
    Mat prediction1 = KF1.predict();
    Point predictPt1(prediction1.at<float>(0),prediction1.at<float>(1));
    int _vmin1 = vmin1, _vmax1 = vmax1;

    cvtColor(image1, hsv1, COLOR_BGR2HSV);
    inRange(hsv1, Scalar(0, smin1, MIN(_vmin1,_vmax1)),
        Scalar(180, 256, MAX(_vmin1, _vmax1)), mask1);
    int ch1[] = {0, 0};
    hue1.create(hsv1.size(), hsv1.depth());
    mixChannels(&hsv1, 1, &hue1, 1, ch1, 1);

    if( trackObject1 < 0 )
    {
        Mat roi1(hue1, selection1), maskroi(mask1, selection1);
        calcHist(&roi1, 1, 0, maskroi, hist1, 1, &hsize, &phranges);
        normalize(hist1, hist1, 0, 255, CV_MINMAX);

        trackWindow1 = selection1;
        trackObject1 = 1;

        histimg1 = Scalar::all(0);
        int binW = histimg1.cols / hsize;
```

```

Mat buf1(1, hsize, CV_8UC3);
for( int i = 0; i < hsize; i++ )
    buf1.at<Vec3b>(i) = Vec3b(saturate_cast<uchar>(i*180./hsize), 255, 255);
cvtColor(buf1, buf1, CV_HSV2BGR);

for( int i = 0; i < hsize; i++ ) {
    int val2 = saturate_cast<int>(hist1.at<float>(i)*histimg1.rows/255);
    rectangle( histimg1, Point(i*binW,histimg1.rows),
               Point((i+1)*binW,histimg1.rows - val2),
               Scalar(buf1.at<Vec3b>(i)), -1, 8 );
}
}

calcBackProject(&hue1, 1, 0, hist1, backproj1, &phranges);
backproj1 &= mask1;
int flagShift1 = meanShift(backproj1, trackWindow1,
                           TermCriteria( CV_TERMCRIT_EPS | CV_TERMCRIT_ITER, 10, 1 ));
RotatedRect trackBox1;
if(flagShift1 != 0) {
    trackBox1 = CamShift(backproj1, trackWindow1,
                        TermCriteria( CV_TERMCRIT_EPS | CV_TERMCRIT_ITER, 10, 1 ));
}
else if(0 < predictPt1.y < 480){
    Rect preWindow1(predictPt1.x, predictPt1.y, 10, 10);
    trackBox1 = CamShift(backproj1, preWindow1,
                        TermCriteria( CV_TERMCRIT_EPS | CV_TERMCRIT_ITER, 10, 1 ));
}
else {
    Rect preWindow1(xCam1, yCam1, 50, 50);
    trackBox1 = CamShift(backproj1, preWindow1,
                        TermCriteria( CV_TERMCRIT_EPS | CV_TERMCRIT_ITER, 10, 1 ));
}

if(trackBox1.size.width != 0 && trackBox1.size.height != 0 )
    ellipse( image1, trackBox1, Scalar(0,255,0), 3, CV_AA );

cout << "\nCam 1 XY Position: \t\t";
cout << trackBox1.center.x << "\t" << trackBox1.center.y;
if(trackBox1.center.x != 0 || trackBox1.center.y != 0) {
    xCam1 = trackBox1.center.x;
    yCam1 = trackBox1.center.y;
    measurement1(0) = xCam1;
    measurement1(1) = yCam1;

    // The update phase

```

```

    Mat estimated1 = KF1.correct(measurement1);
    Point statePt1(estimated1.at<float>(0),estimated1.at<float>(1));
}

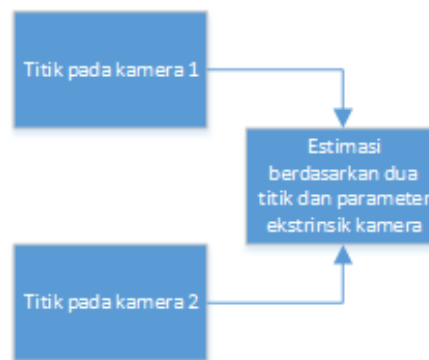
prediction1 = KF1.predict();
Point predictPt1(prediction1.at<float>(0),prediction1.at<float>(1));
cout << "\nCam 1 Predicted Position: \t";
cout << predictPt1.x << "\t" << predictPt1.y;
circle(image1 , Point( predictPt1.x,predictPt1.y), 16.0, Scalar( 0, 155, 255), 3, 8 );
}

```

Gambar 3.10. Pseudocode camshift berbasis Kalman filter

### 3.3.5 Epipolar geometri

Ketika obyek sudah ditemukan dan dianggap sebagai sebuah titik oleh kamera, maka dapat diestimasi posisinya dalam ruang tiga dimensi dengan cara membandingkannya dengan gambar dari kamera dari sudut pandang lain. Salah satu algoritma yang dapat dipakai adalah epipolar geometri.



Gambar 3.11. Algoritma Epipolar geometri

Berikut pseudocode untuk algoritma background subtraction dapat dilihat pada gambar 3.12 sebagai berikut:



```

If( trackObject1 > 0 && trackObject2 > 0) {
    double dist = 600;

    // convert pixel position to angle
    double angleX1 = 90 - ((posX1*64) / 640);
    double angleX2 = (((posX2*64) / 640) + 26);

    // calculate tangensial value for angles
    double tan1 = tan( angleX1 * PI / 180.0 );
    double tan2 = tan( angleX2 * PI / 180.0 );

    // calculate object position
    int posX, posY;
    posX = (tan1 * dist) / (tan1 + tan2);
    posY = (tan2 * posX);

    double stand = 102.0;
    double posR, angleZ, tanZ;

    posR = sqrt(posX*posX + posY*posY);

    if(posY2 > 240) {
        angleZ = ((posY2*48) / 480) - 24;
        tanZ = tan(angleZ * PI / 180.0);
        posZ = posR * tanZ;
        posZ = stand - posZ;
    }
    else if (posY2 < 240){
        angleZ = 24 - ((posY2*48)/480);
        tanZ = tan(angleZ * PI / 180.0);

        posZ = posR * tanZ;
        posZ = stand + posZ;
    }
    else posZ = stand;
}

```

Gambar 3.12. Pseudocode epipolar geometri

## **BAB 4**

### **UJI COBA DAN ANALISIS**

#### **4.1 Pengaturan dan Kalibrasi Kamera**

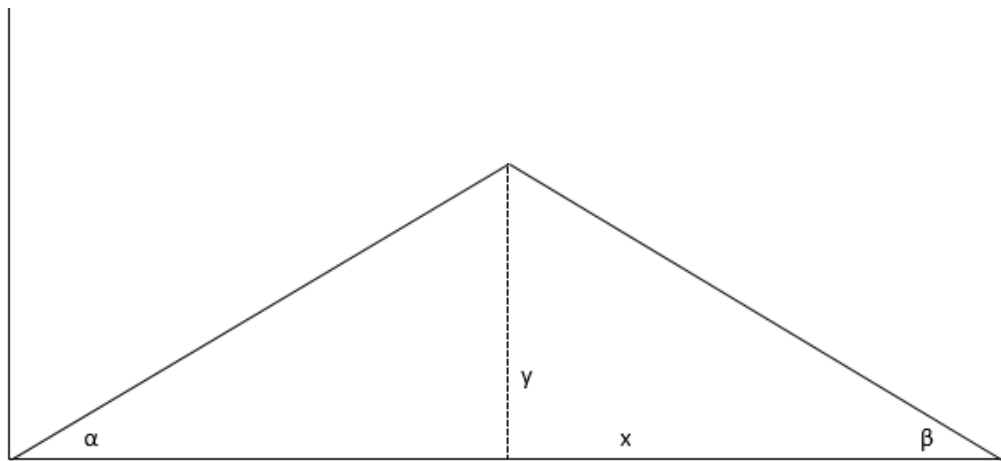
Salah satu hal yang paling penting dalam sistem ini adalah nilai matriks esensial dari kedua kamera yang digunakan. Matriks esensial berisi informasi tentang translasi dan rotasi, yang menggambarkan lokasi kamera kedua relatif terhadap kamera pertama di koordinat global.

Kamera yang digunakan pada sistem ini tidak memiliki kemampuan untuk mengukur derajat kemiringan kamera, sehingga sudut dari posisi kamera harus diatur secara manual oleh pengguna. Karena itu, diperlukan metode kalibrasi posisi kamera yang dapat digunakan untuk mengatur posisi kamera. Hal ini, dapat dilakukan dengan membalik proses yang digunakan oleh sistem ini, di mana pada penggunaan normal sistem menggunakan posisi obyek pada piksel untuk dapat menentukan posisi obyek pada ruang tiga dimensi, maka kita juga dapat menentukan posisi obyek pada piksel yang tepat untuk posisi obyek pada ruang tiga dimensi yang sudah kita ketahui. Adapun tahap-tahapnya adalah sebagai berikut:

1. Tentukan posisi obyek dalam dunia nyata yang sejajar dengan horizon kamera.
2. Cari sudut kamera satu dan kamera dua dengan mengetahui jarak koordinat obyek terhadap kamera.
3. Sesuaikan sudut yang didapatkan dengan offset kamera.
4. Konversi nilai sudut dalam derajat menjadi menjadi nilai piksel pada kamera

5. Arahkan kamera sehingga obyek yang terdeteksi memiliki nilai piksel yang sesuai dengan perhitungan

Misalkan, untuk obyek yang diletakkan sejauh 1.8 meter pada sumbu x dan 3 meter pada sumbu y. Bila sudut offset yang digunakan adalah 26 derajat, maka permodelan sistem dapat digambarkan sebagai berikut:



Gambar 4.1. Model kalibrasi epipolar geometri

Pada gambar di atas  $\alpha$  adalah sudut pada kamera satu,  $\beta$  adalah sudut pada kamera dua,  $x$  adalah jarak kamera ke titik tengah antara dua kamera, sedangkan  $y$  adalah jarak dari garis antara kamera satu dan kamera dua ke obyek. Maka perhitungan untuk mendapatkan posisi obyek pada koordinat piksel tiap kamera adalah sebagai berikut:

$$y = \tan \beta \times x$$

$$1.8 \text{ meter} = \tan \beta \times 3 \text{ meter}$$

$$\tan \beta = \frac{1.8 \text{ meter}}{3 \text{ meter}}$$

$$\beta = 30.96375653^\circ$$

Dari hasil tersebut, dengan mengetahui nilai matriks fundamental di mana lebar gambar yang di ambil kamera pada 640 piksel adalah  $64^\circ$  dan tinggi gambar yang diambil kamera pada 480 piksel adalah  $48^\circ$ , serta offset kamera adalah  $26^\circ$ , dapat ditentukan posisi obyek yang tepat pada kamera dua, yaitu:

$$Cam\ 2\ pixel\ x = \frac{\beta - 26^\circ}{64^\circ} \times 640$$

$$Cam\ 2\ pixel\ x = 49.6$$

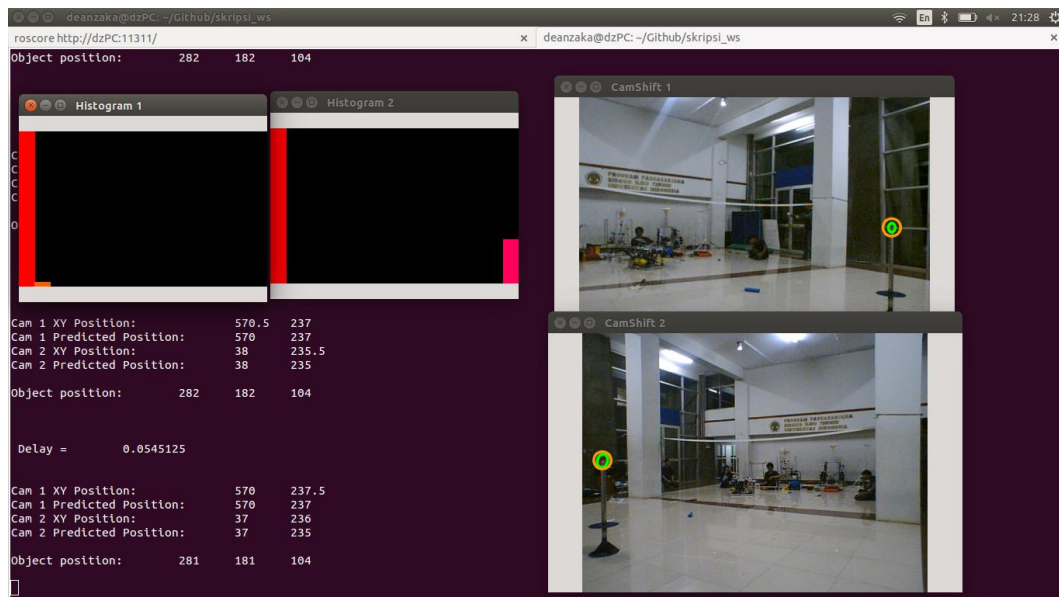
Sedangkan untuk kamera satu yang berseberangan dengan kamera dua, maka posisi obyek yang tepat pada kamera 1 adalah

$$Cam\ 1\ pixel\ x = 640 - 49.6 = 590.4$$

Setelah mendapatkan posisi obyek pada sumbu x, untuk sumbu y, dikarenakan obyek berada pada satu horizon dengan kedua kamera, maka nilai posisi obyek pada sumbu ya adalah setengah kali nilai piksel y, yaitu:

$$Cam\ pixel\ y = \frac{480}{2} = 240$$

Gambar 4.2 menunjukkan tampilan proses kalibrasi sudut kamera:



Gambar 4.2. Proses kalibrasi sudut kamera

## 4.2 Analisis Posisi Obyek terhadap Akurasi

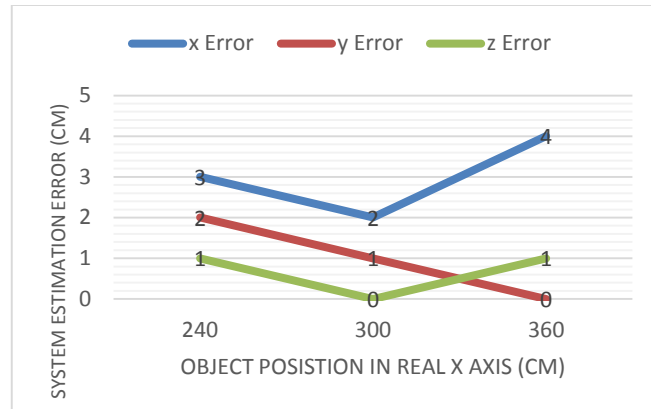
Sebelum melakukan pengujian pada obyek bergerak, dilakukan pengujian pada obyek statis. Hal ini dilakukan untuk mengetahui nilai error yang terjadi pada posisi-posisi obyek tertentu. Pengujian ini dilakukan dengan cara meletakkan obyek pada tiang dengan ketinggian yang sama dengan tiang penyangga kamera dan mulai mengambil gambarnya pada posisi-posisi tertentu. Berikut data yang didapatkan dari pengujian tersebut:

Tabel 4.1. Tabel Uji Epipolar geometri

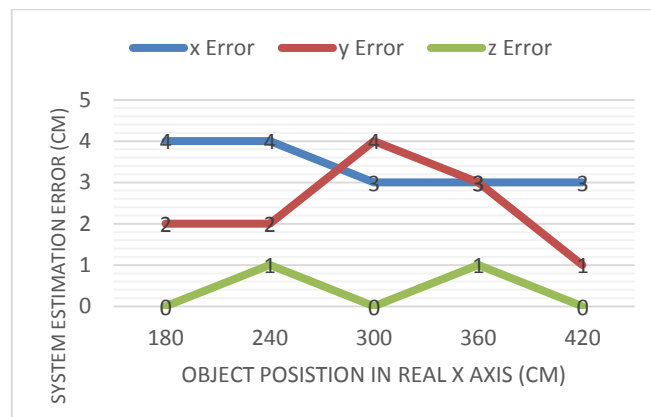
Real (cm)			System (cm)			Error (cm)		
x	y	z	x	y	z	x	y	z
0	180	102						
60	180	102						
120	180	102						
180	180	102						
240	180	102	243	182	103	3	2	1
300	180	102	298	181	102	2	1	0
360	180	102	364	180	103	4	0	1
420	180	102						
480	180	102						
540	180	102						
600	180	102						
0	240	102						

60	240	102						
120	240	102						
180	240	102	176	238	102	4	2	0
240	240	102	236	242	103	4	2	1
300	240	102	297	244	102	3	4	0
360	240	102	357	243	101	3	3	1
420	240	102	417	239	102	3	1	0
480	240	102						
540	240	102						
600	240	102						
0	300	102						
60	300	102	57	296	104	3	4	2
120	300	102	119	299	103	1	1	1
180	300	102	179	304	104	1	4	2
240	300	102	236	304	104	4	4	2
300	300	102	296	305	104	4	5	2
360	300	102	354	305	105	6	5	3
420	300	102	412	302	105	8	2	3
480	300	102	472	299	105	8	1	3
540	300	102	532	295	105	8	5	3
600	300	102						
0	360	102	5	345	107	5	15	5
60	360	102	56	364	105	4	4	3
120	360	102	120	363	105	0	3	3
180	360	102	179	364	105	1	4	3
240	360	102	237	364	106	3	4	4
300	360	102	295	364	106	5	4	4
360	360	102	352	364	106	8	4	4
420	360	102	409	363	107	11	3	5
480	360	102	469	360	106	11	0	4
540	360	102	530	358	107	10	2	5
600	360	102	597	343	107	3	17	5
0	420	102	5	379	109	5	41	7
60	420	102	55	428	106	5	8	4
120	420	102	120	426	107	0	6	5
180	420	102	180	425	107	0	5	5
240	420	102	238	423	107	2	3	5
300	420	102	293	423	107	7	3	5
360	420	102	350	421	107	10	1	5
420	420	102	407	420	107	13	0	5
480	420	102	465	424	107	15	4	5
540	420	102	528	425	109	12	5	7
600	420	102	596	381	109	4	39	7

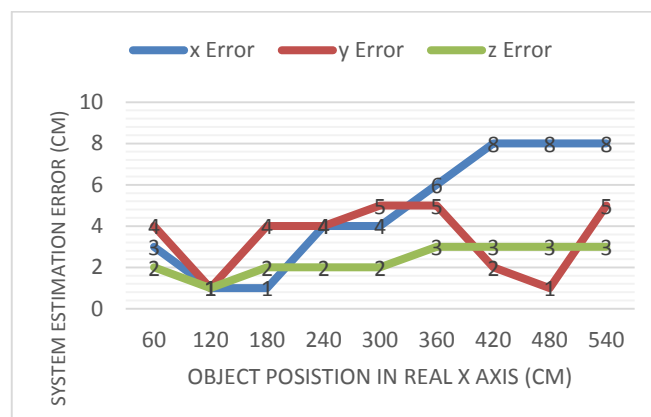
Dari tabel di atas, dapat dilakukan analisis dengan menentukan pengaruh posisi obyek dengan akurasi pada sumbu x dan sumbu y dari dunia nyata. Hasilnya dapat dilihat melalui grafik-grafik berikut:



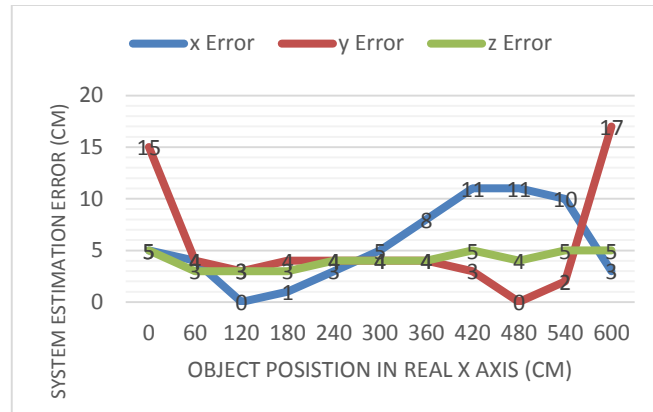
Gambar 4.3. Grafik pada y = 180



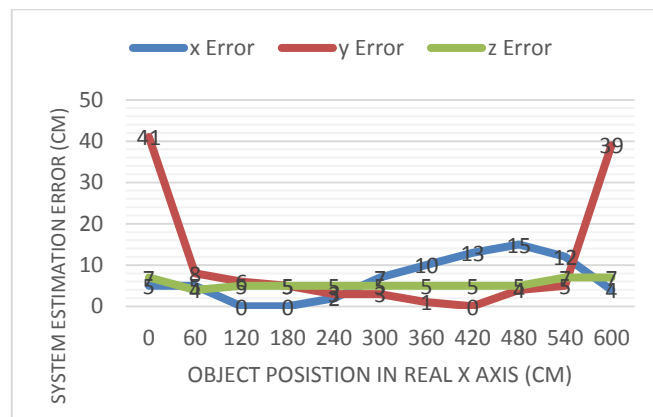
Gambar 4.4 Grafik pada y = 240



Gambar 4.5. Grafik pada y = 300



Gambar 4.6. Grafik pada y = 360



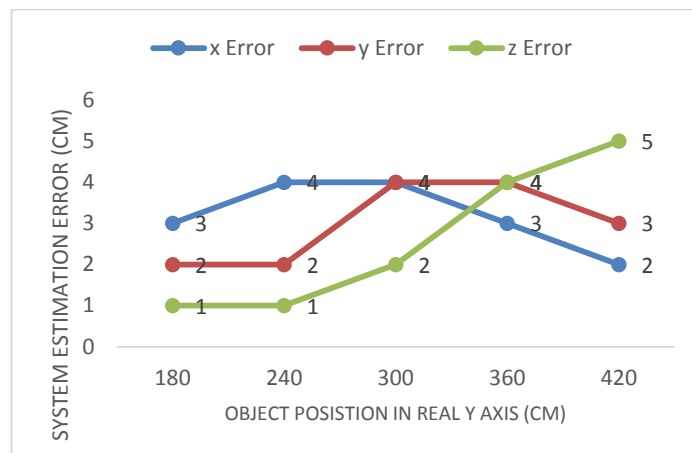
Gambar 4.7. Grafik pada y = 420

Dari grafik-grafik di atas dapat dianalisis bahwa pengaruh peletakkan obyek pada sumbu x dunia nyata (semakin ke kanan atau semakin ke kiri) memiliki pengaruh kecil terhadap estimasi sistem terhadap posisi benda pada sumbu z (ketinggian obyek). Sedangkan pada sumbu y, ketika benda diletakkan pada jarak di bawah 300 cm, error tertinggi terjadi pada titik tengah benda, hal ini terjadi akibat benda belum melewati titik tengah pada piksel sehingga error terbesar terjadi pada estimasi di tengah. Ketika benda berada lebih jauh dari 300 cm pada sumbu y, error lebih banyak terjadi pada saat benda berada di paling kanan atau paling kiri daerah deteksi, hal ini dikarenakan kamera hanya menangkap kedua benda di ujung-ujungnya sehingga terjadi error yang cukup besar. Hal menarik terjadi pada pembacaan sistem di sumbu x, karena meskipun peletakkan benda

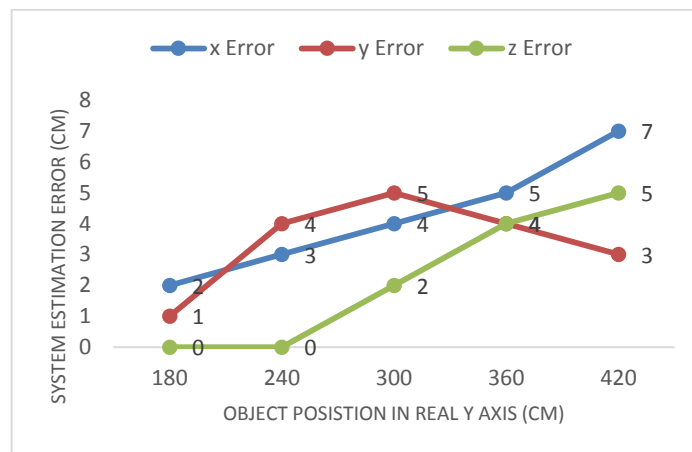


dan kamera simetris namun ternyata pembacaan sistem memiliki error yang lebih besar ketika obyek yang di deteksi berada di kanan. Hal ini dikarenakan adanya perbedaan cahaya latar antara kamera satu dan kamera 2.

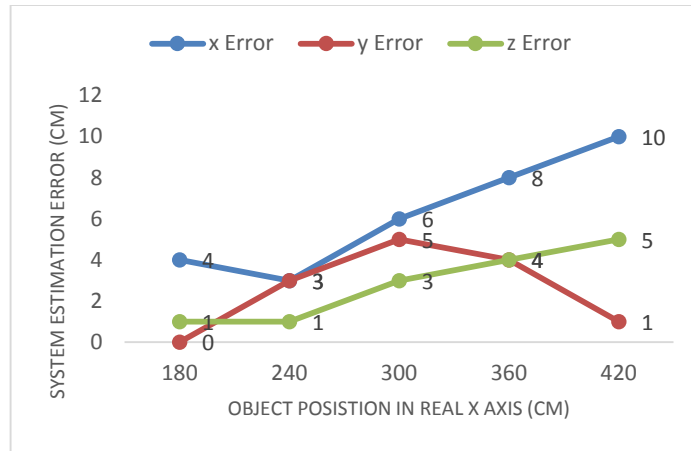
Selanjutnya, untuk analisis pengaruh peletakkan obyek pada sumbu y di dunia nyata, dapat dilihat pada grafik-grafik berikut:



Gambar 4.8. Grafik pada x = 240



Gambar 4.9. Grafik pada x = 300

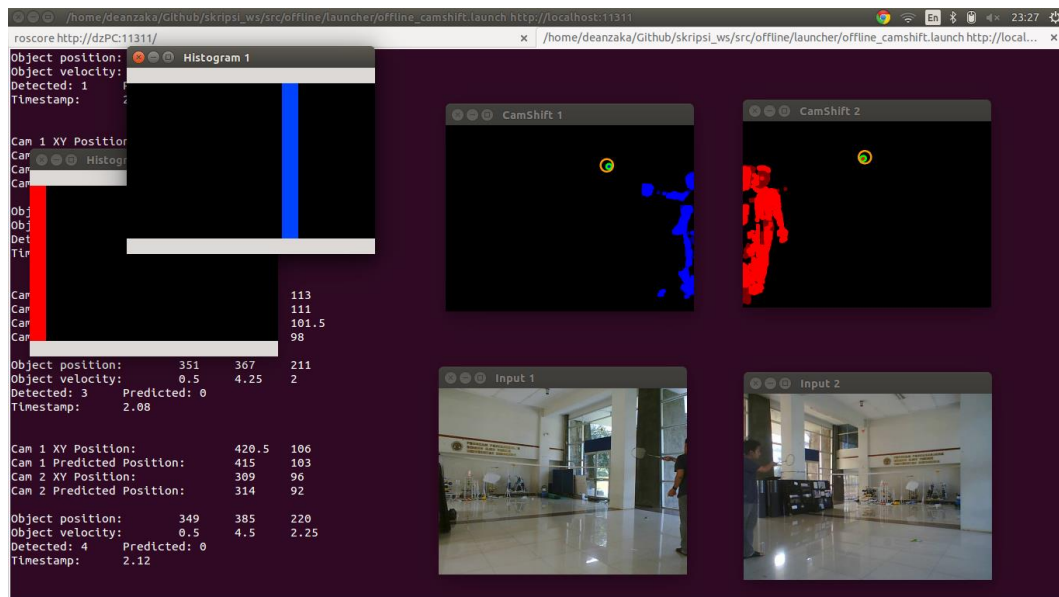


Gambar 4.10. Grafik pada x = 360

Dapat dilihat dari grafik-grafik di atas, nilai error selalu bertambah seiring dengan bertambahnya posisi benda di sumbu y. Hal ini, terjadi terutama pada nilai sumbu x dan sumbu z. Hal ini dikarenakan semakin jauh benda dari kamera, maka piksel yang terbaca dari obyek juga menjadi semakin kecil, sehingga error dari pembacaannya pun meningkat.

### 4.3 Analisis Pengaruh Arah Gerak Obyek terhadap Hasil Deteksi

Setelah dilakukan pengujian pada obyek statis, sistem siap melakukan pendeteksian pada obyek bergerak. Pada bagian ini dilakukan analisis terhadap pengaruh arah gerak obyek kepada hasil deteksi. Adapun parameter yang digunakan yaitu persentase trayektori yang didapatkan oleh sistem serta persentase perbandingan obyek terdeteksi dengan obyek terprediksi pada trayektori yang didapatkan. Tampilan sistem deteksi dapat dilihat pada gambar berikut:



Gambar 4.11. Tampilan sistem

Persentase trayektori didapatkan dengan membandingkan waktu sejak obyek pertama kali terdeteksi dengan waktu sejak obyek pertama kali dipukul hingga obyek mendarat di permukaan tanah. Pada percobaan ini digunakan *shuttlecock* berwarna merah, hal ini dilakukan untuk membedakan warna obyek dengan latar. Hasil dari uji coba tersebut dapat dilihat pada tabel berikut:

Tabel 4.2. Tabel Uji Arah Obyek Mendekati Kamera

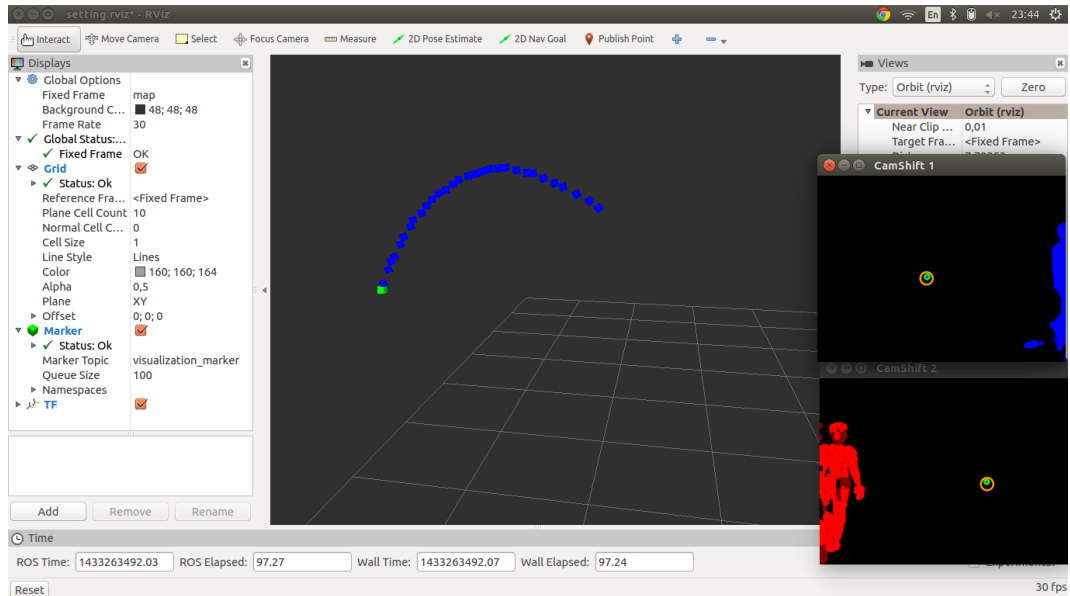
No	Trayektori terdeteksi dari pukulan pertama hingga mendarat				Obyek terdeteksi dalam trayektori		
	Pukulan pertama (detik)	Terdeteksi pertama (detik)	Mendarat (detik)	Persentase	Terdeteksi	Terprediksi	Persentase
1	3,4	4	4,88	59,46	20	2	90,91
2	5,92	6,24	7,2	68,42	24	0	100,00
3	1,36	1,84	2,88	60,00	25	2	92,59
4	3	3,64	4,6	74,36	25	0	96,77
5	1,72	2,12	3,28	75,68	30	1	100,00
6	4,2	4,56	5,68	70,59	28	0	96,15
7	1,16	1,56	2,52	84,62	25	1	100,00
8	3,4	3,64	4,96	82,50	33	0	97,06
9	1,4	1,68	3,00	81,69	33	1	89,29
10	3,74	4	5,16	75,00	25	3	92,86
11	1,12	1,48	2,56	80,00	26	2	96,97
12	4,56	4,88	6,16	65,63	32	1	91,30
13	2,12	2,56	3,40	76,32	21	2	100,00
14	4,16	4,52	5,68	76,32	30	0	96,55
15	1,4	1,76	2,92	76,32	28	1	96,67
16	3,72	4,08	5,24	76,67	29	1	100,00

17	1,2	1,48	2,40	77,78	23	0	96,43
18	3,4	3,72	4,84	69,70	27	1	95,83
19	1,32	1,72	2,64	69,44	23	1	100,00
20	3,48	3,92	4,92	69,44	26	0	100,00
Average				73,50	Average		96,47

Tabel 4.3. Tabel Uji Arah Obyek Menjauhi Kamera

No	Trayektori terdeteksi dari pukulan pertama hingga mendarat				Obyek terdeteksi dalam trayektori		
	Pukulan pertama (detik)	Terdeteksi pertama (detik)	Mendarat (detik)	Persentase	Terdeteksi	Terprediksi	Persentase
1	1,76	1,96	3,2	86,11	29	3	90,63
2	5,48	5,68	7,2	88,37	38	1	97,44
3	1,76	2,04	3,44	83,33	31	5	86,11
4	4,36	4,64	6,04	83,33	33	2	94,29
5	2	2,2	3,56	87,18	30	6	83,33
6	4,49	4,6	6,08	93,08	34	4	89,47
7	2,32	2,6	3,96	82,93	27	8	77,14
8	4,92	5,24	6,60	80,95	31	6	83,78
9	1,28	1,52	2,76	83,78	24	8	75,00
10	3,72	4,16	5,32	72,50	27	3	90,00
11	1,24	1,52	2,60	79,41	25	6	80,65
12	3,76	4	5,16	82,86	28	2	93,33
13	1,32	1,56	2,68	82,35	27	5	84,38
14	7,52	7,72	9,04	86,84	26	8	76,47
15	2,56	2,88	4,28	81,40	31	5	86,11
16	5,08	5,4	6,36	75,00	26	3	89,66
17	2,44	2,64	3,80	85,29	25	4	86,21
18	4,56	4,8	6,04	83,78	24	7	77,42
19	1,16	1,48	2,84	80,95	34	4	89,47
20	3,8	4	5,36	87,18	28	7	80,00
Average				83,33	Average		85,54

Seperti dapat dilihat pada kedua tabel di atas, obyek yang dideteksi mendekati kamera memiliki trayektori terdeteksi lebih pendek dibandingkan bola yang menjauhi kamera, hal ini dikarenakan titik awal trayektori dimulai jauh dari kamera sehingga sulit terbaca oleh sistem. Namun dapat dilihat bahwa persentase pembacaan obyek dari trayektori yang di dapat lebih besar bila obyek mendekati kamera, ini artinya obyek yang mendekati kamera tidak memiliki banyak hilang deteksi pada saat bergerak. Hal ini dikarenakan sebagian besar trayektori terdeteksi dari obyek berada dekat dengan kamera, sedangkan pada saat obyek menjauhi kamera, sebagian besar trayektori yang terbaca berada jauh dari kamera. Adapun hasil pembacaan yang disimulasikan dalam sistem rviz dari ROS dapat dilihat pada gambar berikut:



Gambar 4.12. Tampilan Simulasi rviz dengan ROS

#### 4.4 Analisis Pengaruh Perbedaan Warna Obyek dengan Latar

Seperti yang telah disebutkan dalam percobaan sebelumnya, *shuttlecock* yang digunakan pada percobaan adalah *shuttlecock* berwarna merah, hal ini dimaksudkan untuk membedakan warna obyek dengan warna latar. Dalam percobaan kali ini, sistem dicoba dengan menggunakan *shuttlecock* yang memiliki warna yang sama dengan latar. Hasilnya dapat dilihat pada tabel berikut ini:

Tabel 4.4. Tabel Uji Warna Obyek menyerupai Latar

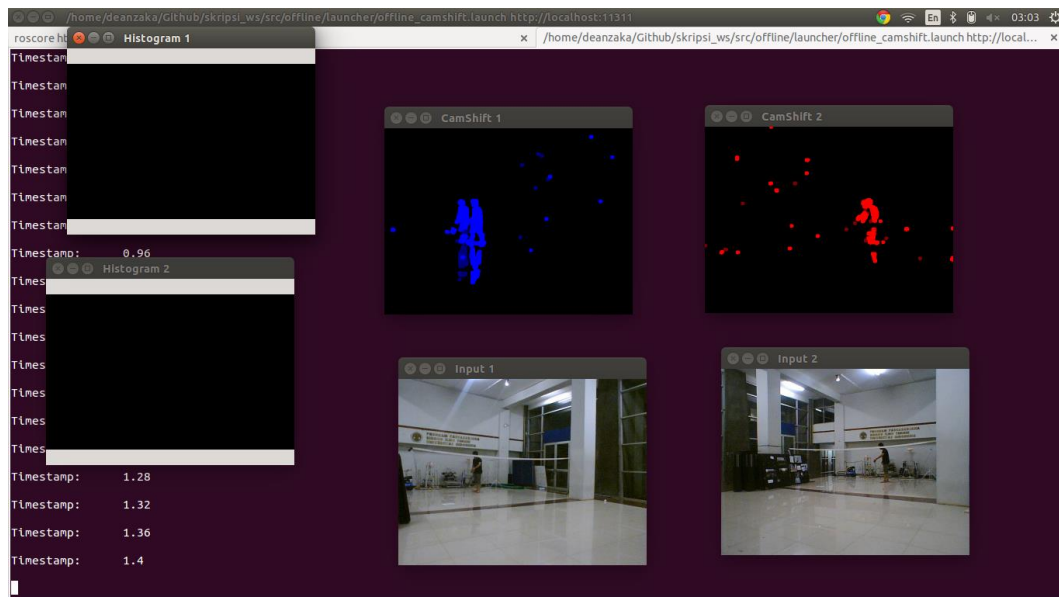
No	Trayektori terdeteksi dari pukulan pertama hingga mendarat				Obyek terdeteksi dalam trayektori		
	Pukulan pertama (detik)	Terdeteksi pertama (detik)	Mendarat (detik)	Persentase	Terdeteksi	Terprediksi	Persentase
1	0,44	1,28	2	46,15	17	2	89,47
2	2,76	4,2	4,36	10,00	8	2	80,00
3	1	1,88	2,28	31,25	11	0	100,00
4	4,2	5,28	5,92	37,21	16	1	94,12
5	0,72	1,92	2,16	16,67	6	1	85,71
6	2,6	3,52	3,92	30,30	9	2	81,82
7	0,64	1,8	2,28	29,27	12	1	92,31
8	2,3	3,56	4,08	29,21	12	2	85,71
9	1,32	2,16	2,48	27,59	8	0	100,00
10	3,48	4,8	4,96	10,81	3	1	75,00

11	1,8	2,84	3,20	25,71	9	1	90,00
12	4	5,28	5,52	15,79	7	0	100,00
13	2	2,92	3,56	41,03	14	3	82,35
14	4,24	5,68	5,92	14,29	7	0	100,00
15	1,4	2,68	2,96	17,95	8	0	100,00
16	3,88	5,08	5,32	16,67	7	0	100,00
17	0,76	2,16	2,40	14,63	7	0	100,00
18	3,2	4,28	4,60	22,86	9	0	100,00
19	1,88	2,88	3,28	28,57	11	0	100,00
20	3,92	4,96	5,48	33,33	13	1	92,86
Average				24,96	Average		92,47

Dapat dilihat pada tabel di atas persentase trayektori terbaca turun drastis dibandingkan dengan uji coba menggunakan obyek berwarna merah. Namun, obyek tidak benar-benar menghilang dari sistem. *Shuttlecock* masih dapat terbaca pada akhir trayektori. Hal ini dikarenakan posisi *shuttlecock* sudah mendekati kamera sehingga piksel warna yang ditangkap mulai bisa dibaca perbedaannya. Selain itu, dapat dilihat pada persentase obyek terdeteksi pada trayektori terbaca berada di atas 90%, hal ini menunjukkan begitu trayektori terdeteksi, pembacaan obyek cukup baik meskipun obyek memiliki warna menyerupai latar.

#### 4.5 Analisis Pengaruh Cahaya terhadap Hasil Deteksi

Pada analisis pengaruh cahaya pada pembacaan, sistem dicoba dengan menggunakan pencahayaan lampu normal pada malam hari. Hasilnya, terdapat noise yang cukup banyak sehingga sistem tidak dapat membedakan mana obyek mana titik-titik yang diakibatkan oleh noise. Ini berarti, pencahayaan lingkungan memiliki dampak yang sangat besar pada sistem ini. Adapun hasil dari percobaan dapat dilihat pada gambar di bawah ini:



Gambar 4.13. Hasil Percobaan pada Malam Hari

## **BAB 5**

### **KESIMPULAN**

1. Hasil pengujian sistem pada obyek statis menunjukkan sistem dapat mendeteksi posisi obyek di ruang tiga dimensi dengan menggunakan metode epipolar geometri. Namun, masih terlihat adanya peningkatan nilai error posisi obyek yang meningkat seiring dengan meningkatnya jarak obyek dengan kamera, hal ini dikarenakan faktor kalibrasi sudut kamera yang dilakukan secara manual sehingga tidak benar-benar sesuai dengan perhitungan..
2. Hasil pengujian sistem pada obyek bergerak menunjukkan sistem dapat mendeteksi rata-rata 83.33% trayektori *shuttlecock* dengan persentase deteksi rata-rata dalam satu trayektori 85.54%. Serta sistem masih dapat melanjutkan deteksi meskipun kehilangan obyek di tengah-tengah trayektori, maka dapat disimpulkan algoritma camshift berbasis Kalman filter adalah algoritma yang baik dalam pendeteksian trayektori *shuttlecock*.
3. Setelah dilakukan perubahan arah *shuttlecock* menuju ke arah kamera, trayektori terdeteksi turun menjadi rata-rata 73.5% namun persentase deteksi meningkat hingga 96.47%. Hasil ini sesuai dengan hasil pengujian pada obyek statis di mana obyek yang lebih dekat dengan kamera akan memiliki akurasi pembacaan posisi yang lebih tinggi.
4. Setelah dilakukan perubahan warna obyek sehingga menyerupai latar gambar yang di ambil. Trayektori terdeteksi turun hingga rata-rata 24.96%. Hal ini menunjukkan perbedaan warna obyek dengan latar gambar sangat berpengaruh pada hasil yang didapatkan sistem.
5. Setelah dilakukan percobaan dengan pencahayaan minimal pada malam hari, hasilnya menunjukkan bahwa sistem tidak dapat membedakan obyek dengan titik-titik yang dihasilkan oleh *noise*, sehingga sistem sama sekali tidak dapat



digunakan. Hal ini menunjukkan bahwa pencahayaan ruangan memiliki pengaruh yang sangat signifikan terhadap sistem.

## DAFTAR PUSTAKA

- [1] A. Kaehler dan G. Bradsky, *Learning OpenCV*, O'Reilly, 2013.
- [2] W. Li dan B. Li, "Map Estimation of Epipolar Geometry by EM Algorithm," dalam *IEEE International Conference on Image Processing*, San Antonio, 2007.
- [3] G. Bradsky, "OpenCV 3.0.0-dev documentation," [Online]. Available: [http://docs.opencv.org/trunk/doc/py\\_tutorials/py\\_calib3d/py\\_epipolar\\_geometry/py\\_epipolar\\_geometry.html](http://docs.opencv.org/trunk/doc/py_tutorials/py_calib3d/py_epipolar_geometry/py_epipolar_geometry.html). [Diakses 18 Desember 2014].
- [4] H. Sekkati, R. Laganieri, A. Mitiche dan R. Youmaran, "Robust background subtraction using geodesic active contours in ICA subspace for video surveillance application," dalam *Ninth Conference on Computer and Robot Vision*, Toronto, 2012.
- [5] "OpenCV 3.0.0-dev documentation," [Online]. Available: [http://docs.opencv.org/trunk/doc/py\\_tutorials/py\\_video/py\\_bg\\_subtraction/py\\_bg\\_subtraction.html](http://docs.opencv.org/trunk/doc/py_tutorials/py_video/py_bg_subtraction/py_bg_subtraction.html). [Diakses 18 Desember 2014].
- [6] S. Huang dan J. Hong, "Moving Object Tracking System Based On Camshift," dalam *International Conference on Consumer Electronics, Communications and Networks (CECNet)*, Xianning, 2011.
- [7] "ROS," [Online]. Available: <http://www.ros.org/>. [Diakses 29 05 2015].
- [8] "PLAYSTATIONEye Brings Next-Generation Communication to PLAYSTATION3," Sony Computer Entertainment, 26 April 2007. [Online]. Available: <http://www.us.playstation.com/News/PressReleases/396>. [Diakses 18 Desember 2014].

- [9] Rochester Institute of Technology, “cias.rit.edu,” [Online]. Available: <http://cias.rit.edu/~nmtp/2063809/vision2020/FOV-1.pdf>. [Diakses 18 Desember 2014].