TUGAS AKHIR – TM 141585

PENENTUAN KOORDINAT 3 DIMENSI TARGET TUNGGAL PADA SISTEM PELONTAR PELURU *AUTOTRACKING* MENGGUNAKAN STEREO KAMERA BERSUDUT BETHA 15 DERAJAT

FEBRIANA PUSPARANIAYU PASA
NRP 2111 100 025

Dosen Pembimbing:
Arif Wahjudi, ST, MT, PhD.

JURUSAN TEKNIK MESIN
Fakultas Teknologi Industri
Institut Teknologi Sepuluh Nopember
Surabaya 2017

LAPORAN TUGAS AKHIR – TM141585

PENENTUAN KOORDINAT 3 DIMENSI TARGET TUNGGAL PADA SISTEM PELONTAR PELURU AUTOTRACKING MENGGUNAKAN STEREO KAMERA BERSUDUT BETHA 15 DERAJAT

FEBRIANA PUSPARANIAYU PASA
NRP 2112 100 011

Dosen Pembimbing
Arif Wahjudi, ST, MT, PhD.

JURUSAN TEKNIK MESIN
Fakultas Teknologi Industri
Institut Teknologi Sepuluh Nopember
Surabaya
2017

FINAL PROJECT – TM141585

THREE DIMENSION COORDINATE DETERMINATION OF A SINGLE TARGET IN AN AUTOTRACKING SENTRY GUN SYSTEM USING STEREO CAMERA WITH 15 DEGREE BETHA ANGLE

FEBRIANA PUSPARANIAYU PASA
NRP 2112 100 011

Advisor
Arif Wahjudi, ST, MT, PhD.

MECHANICAL ENGINEERING
Faculty of Industrial Technology
Sepuluh Nopember Institute of Technology
Surabaya
2017

PENENTUAN KOORDINAT 3 DIMENSI TARGET TUNGGAL PADA SISTEM PELONTAR PELURU *AUTOTRACKING* MENGGUNAKAN STEREO KAMERA BERSUDUT β 15 DERAJAT

TUGAS AKHIR

Diajukan Untuk Memenuhi Salah Satu Syarat
Memperoleh Gelar Sarjana Teknik
pada
Program Studi S-1 Jurusan Teknik Mesin
Fakultas Teknologi Industri
Institut Teknologi Sepuluh Nopember

Oleh :

FEBRIANA PUSPARANIAYU PASA

NRP. 2112 100 011

Disetujui oleh Tim Penguji Tugas Akhir :

1. Arif Wahjudi, ST, MT, PhD. ..... (Pembimbing)
   NIP. 197303222001121001
2. Prof. Dr. Ing. I Made Londen B., ME. ..... (Penguji I)
   NIP. 195811061986011002
3. Hendro Nurhadi, Dipl. Ing.,PhD. ..... (Penguji II)
   NIP. 197511202002121002
4. Dinny Harnany, ST, MSc. ..... (Penguji III)
   NIP. 2100201405001

SURABAYA

JANUARI, 2017

PENENTUAN KOORDINAT 3 DIMENSI TARGET TUNGGAL PADA SISTEM PELONTAR PELURU *AUTOTRACKING* MENGGUNAKAN STEREO KAMERA BERSUDUT BETHA 15 DERAJAT

Nama	**: FEBRIANA P. P.**
NRP	**: 2112 100 011**
Jurusan	**: TEKNIK MESIN FTI-ITS**
Dosen Pembimbing	**: ARIF WAHJUDI, ST., MT., PhD.**

ABSTRAK

Teknologi pengukuran saat ini sudah berkembang dengan pesat. Teknik pengukuran yang saat ini sedang banyak dikembangkan adalah *image processing*. *Image processing* sendiri pengaplikasiannya sangat luas. Teknik pengukuran ini dapat diaplikasikan dalam bidang industri, bidang manufaktur hingga bidang militer. Dalam bidang militer, teknik pengukuran *image processing* mulai dimanfaatkan untuk pembuatan sistem senjata otomatis. Sistem senjata otomatis ini dikembangkan untuk mengurangi jumlah korban penembakan yang disebabkan oleh kesalahan manusia. Pada saat ini sudah ada beberapa macam *prototype* mesin penembak berupa pelontar peluru otomatis. Oleh sebab itu, pada Tugas Akhir ini diharapkan dapat terbentuk sebuah program pendeteksi dan penentu koordinat 3 dimensi suatu objek dengan menggunakan stereo kamera yang dapat diaplikasikan pada alat *prototype* mesin pelontar peluru otomatis.

Langkah kerja penelitian ini pertama adalah melakukan proses kalibrasi stereo kamera dengan memanfaatkan gambar papan catur yang diperoleh dari *library* OpenCV. Dari proses kalibrasi ini akan diperoleh nilai-nilai intrinsik dari kamera. Setelah itu, dilakukan proses pengolahan data gambar yang ditangkap oleh stereo kamera dengan menggunakan metode *image processing*. Proses ini akan dilakukan dengan bantuan perangkat lunak Visual Studio 2015 dan *library* OpenCV 2.10. Pendeteksian objek dilakukan dengan mendeteksi warna tertentu dari suatu

objek. Setelah objek berhasil terdeteksi, nilai koordinat 3 dimensi objek dicari dengan memanfaatkan nilai intrinsik kamera. Proses penentuan koordinat z, x, dan y dari objek menggunakan metode *algorithm triangulation*.

Hasil yang diperoleh dari Tugas Akhir ini adalah sebuah program yang bisa mendeteksi dan melacak objek berdasarkan warna dengan tingkat keberhasilan 94,6% dan dapat menentukan koordinat 3 dimensi dari objek tersebut dengan nilai error terbesar untuk jarak z 3,92%, koordinat x 0,64% dan koordinat y 2,5%. Program hasil penelitian Tugas Akhir ini akan diaplikasikan pada alat pelontar peluru.

Kata kunci: ***coordinate calculation, image processing, non-contact measurement, object detection, stereo camera calibration.***

THREE DIMENSION COORDINATE DETERMINATION OF A SINGLE TARGET IN AN AUTOTRACKING SENTRY GUN SYSTEM USING STEREO CAMERA WITH 15 DEGREE BETHA ANGLE

Name	**: FEBRIANA P. P.**
NRP	**: 2112 100 011**
Department	**: TEKNIK MESIN FTI-ITS**
Academic Supervisor	**: ARIF WAHJUDI, ST., MT., PhD.**

ABSTRACT

Nowadays, one of measurement technology which many people developed is image processing. Image processing usually is applied in the industrial and manufacturing world but it has been developed in the military section. In the military, image processing is used for the making of automatic weapon system. This automatic weapon system was made to reduce the number of people fall victim due to a misfired or a stray bullets. There are some firing machine prototypes such as an automatic sentry gun. In the hope of making a new system for the automatic sentry gun, a program of single object detection and 3 dimension coordinate determination using stereo camera was made to be applied in the prototype system.

The first step in the process was running a calibration program for the stereo camera using a black and white chessboard image as an input. The calibration program can be obtained from OpenCV library. From this calibration process, intrinsic values of the cameras was obtained. The next step was processing the images captured by the cameras using image processing method. Object detection was made by detecting a certain colour of an object. After succeeded in the detection, the object's 3 dimension coordinate was calculated by using the cameras' intrinsic value which is its focal length. The calculation of z, x, and y coordinate used the algorithm triangulation method.

Results taken from this project was a program which can be used to detect an object based on a certain colour with a successful rate of 94.6% and determine its 3 dimension coordinate with the highest error 3.92% for z coordinate, 0,64% for x coordinate and 2.5% for y coordinate. This program was planned to be applied on a sentry gun prototype.

Keyword: coordinate calculation, image processing, non-contact measurement, object detection, stereo camera calibration.

KATA PENGANTAR

Puji dan syukur marilah kita panjatkan kehadirat Tuhan Yang Maha Esa. Karena setiap anugerah dan karunia-Nya, penulis mampu menyelesaikan Tugas Akhir yang berjudul **Penentuan Koordinat 3 Dimensi Target Tunggal pada Sistem Pelontar Peluru *Autotracking* Menggunakan Stereo Kamera Bersudut Betha 15 Derajat.**

Tidak lupa ucapan terima kasih penulis sampaikan kepada berbagai pihak yang telah membantu dalam penyelesaian Tugas Akhir ini. Tanpa orang-orang tersebut, mustahil laporan tugas akhir ini dapat diselesaikan dengan baik. Adapun ucapan terima kasih penulis sampaikan kepada:

1. **Kedua orang tua serta kakak dan adik** penulis yang selalu mendoakan, mendidik, dan memberi semangat putra-putrinya untuk melakukan yang terbaik.
2. **Bapak Arif Wahjudi, S.T., M.T., Ph.D.,** selaku dosen pembimbing Tugas Akhir yang selalu memberikan ilmu, kritik dan saran, serta bimbingannya selama proses penyelesaian Tugas Akhir ini.
3. **Bapak Prof. Dr. Ing. I Made Londen Batan, M.E., Bapak Hendro Nurhadi, Dipl. Ing., Ph.D.,** dan **Ibu Dinny Harnany, S.T., M.T.**, selaku dosen penguji Tugas Akhir, yang telah menyempatkan berbagi ilmu, sehingga tugas akhir ini dapat terselesaikan dengan baik.
4. **Bapak Prof. Dr. Ir. Abdullah Shahab, M.Sc.**, selaku dosen wali yang telah membimbing penulis selama masa studi di Teknik Mesin ITS.
5. **Agung Kartika Fibrianto** dan **Deris Triana Noor** selaku partner dalam pembuatan Tugas Akhir ini.
6. Segenap keluarga besar Laboratorium Perancangan dan Pengembangan Produk yang selalu memberikan semangat dan motivasi kepada penulis.

7. Dosen dan karyawan Jurusan Teknik Mesin FTI-ITS yang telah memberikan ilmu, pengalaman, dan bantuannya selama masa studi.
8. Serta semua pihak yang secara langsung ataupun tidak langsung terlibat dalam proses penyelesaian tugas akhir ini.

Dengan selesainya Tugas Akhir ini, penulis berharap laporan ini dapat bermanfaat khususnya bagi penulis sendiri dan umumnya untuk kita semua.

Surabaya, 20 Januari 2017

Febriana Pusparaniayu Pasa

DAFTAR ISI

HALAMAN JUDUL i
LEMBAR PENGESAHAN iii
ABSTRAK v
KATA PENGANTAR ix
DAFTAR ISI xi
DAFTAR GAMBAR xiii
DAFTAR TABEL xv
BAB 1 1
PENDAHULUAN 1
1.1 Latar Belakang 1
1.2 Rumusan Masalah 2
1.3 Batasan Masalah 2
1.4 Tujuan Peneletian 3
1.5 Manfaat Penelitian 3
BAB 2 5
TINJAUAN PUSTAKA 5
2.1 Tinjauan Pustaka 5
2.2 Dasar Teori 6
2.2.1. *Stereo Vision* 6
2.2.2. Gambar Digital 7
2.2.3. Kedalaman Gambar 8
2.2.4. Koordinat 3 Dimensi 13
BAB 3 17
METODE PENELITIAN 17
3.1 *Flowchart* Metodologi Penelitian 17
3.2 Proses Persiapan 18
3.3 Proses Perancangan Program 19
3.3.1. Cara Kerja 20
3.3.2. Proses dan Pengerjaan 21
3.3.3. *Flowchart* Perancangan Program Kalibrasi Stereo Kamera 23
3.3.4. *Flowchart* Perancangan Program Pendeteksi dan Penentu Koordinat 24
3.4 Tahap Implementasi 25

BAB 4.....27
PERANCANGAN DAN PEMBUATAN PROGRAM 27
4.1 Implementasi Program dalam Sistem.....27
4.2 Konstruksi Program.....27
4.2.1. Program Pendeteksian Objek 28
4.2.2. Pencarian Variabel Pendeteksian Objek.....39
4.2.3. Program Penentuan Koordinat x, y, z.....43
4.3 Proses Kalibrasi.....45
BAB 5.....49
PENGUJIAN PENDETEKSIAN OBJEK.....49
5.1 Pendeteksian Objek 49
5.2 Pendeteksian Jarak (Koordinat z).....50
5.3 Pendeteksian Koordinat x.....54
5.4 Pendeteksian Koordinat y.....56
BAB 6.....61
VERIFIKASI PROGRAM 61
6.1 Verifikasi Program untuk Jarak z 61
6.2 Verifikasi Program untuk Jarak x.....62
6.3 Verifikasi Program untuk Jarak y.....64
BAB 7.....67
KESIMPULAN DAN SARAN 67
7.1 Kesimpulan.....67
7.2 Saran.....67
DAFTAR PUSTAKA 69
LAMPIRAN.....71

DAFTAR GAMBAR

Gambar 2.1 Titik *scene* dari kamera streo.....6
Gambar 2.2 Warna RGB 8
Gambar 2.3 *Mother Theresa in grayscale*.....10
Gambar 2.4 *Mother Theresa in binary image* 11
Gambar 2.5 Gambar *grayscale* dan gambar proses *threshold*....12
Gambar 2.6 Sistem koordinat 3 dimensi 15
Gambar 2.7 Pemodelan sederhana sistem stereo kamera 15
Gambar 3.1 Diagram alir metodologi penelitian.....18
Gambar 3.2 *Prototype* Alat Pelontar Peluru beserta bagian bagiannya.....19
Gambar 3.3 Gambar asli dengan gambar hasil *threshold*.....22
Gambar 3.4 Diagram alir rancangan program Kalibrasi Stereo Kamera.....23
Gambar 3.5 Diagram alir perancangan program Pendeteksi Objek.....24
Gambar 4.1 Diagram alir proses pengolahan dari gambar RGBke HSV 29
Gambar 4.2 Gambar RGB sebagai gambar masukan *frame* kiri dan kanan.....30
Gambar 4.3 Gambar *threshold* 31
Gambar 4.4 Diagram alir proses *threshold*.....32
Gambar 4.5 Gambar hasil *threshold* yang telah diproses *noise filtering* 33
Gambar 4.6 Diagram alir proses *erode* dan *dilate*.....34
Gambar 4.7 Gambar *threshold* yang telah dilakukan proses *erode* 35
Gambar 4.8 Gambar *threshold* yang telah dilakukan proses *dilate* 35
Gambar 4.9 *Trackbar* yang digunakan untuk mencari perubahan gambar HSV menjadi gambar *threshold*.....36
Gambar 4.10 Proses *threshold* dan *noise filtering*.....38
Gambar 4.11 Informasi hasil pencarian nilai HSV dengan *trackbar*.....39
Gambar 4.12 Diagram alir pencarian koordinat tepi objek 40

Gambar 4.13 Diagram alir pembuatan persegi.....42
Gambar 4.14 Diagram alir penentuan koordinat x, y, dan z objek terhadap kamera.....44
Gambar 4.15 Papan catur yang digunakan untuk proses kalibrasi.....46
Gambar 4.16 Diagram alir proses kalibrasi *stereo vision*.....46
Gambar 4.17 Proses kalibrasi *stereo vision*.....47
Gambar 5.1 Hasil pendeteksian objek bola tenis.....49
Gambar 5.2 Hasil kesalahan pendeteksian objek bola tenis.....50
Gambar 5.3 Skema pengambilan data z dan x 51
Gambar 5.4 Grafik persamaan kompensasi nilai z untuk z < 800 52
Gambar 5.5 Grafik persamaan kompensasi nilai z untuk 800 ≤ z < 1100.....53
Gambar 5.6 Grafik persamaan kompensasi nilai z untuk 1100 ≤ z 53
Gambar 5.7 Grafik persamaan kompensasi hasil pengujian nilai koordinat x.....55
Gambar 5.8 Grafik persamaan kompensasi hasil pengujian nilai koordinat y.....58

DAFTAR TABEL

Tabel 6.1 Hasil Pengujian Program Nilai z dengan Persamaan Kompensasi 61
Tabel 6.2 Hasil Pengujian Nilai x dengan Persamaan Kompensasi 63
Tabel 6.3 Selisih x Terdeteksi dengan x Sebenarnya dengan Persamaan Kompensasi 64
Tabel 6.4 Hasil Pengujian Progam Nilai y dengan Persamaan Kompensasi 65
Tabel 6.5 Selisih y Terdeteksi dengan y Sebenarnya dengan Persamaan Kompensasi 66

(Halaman ini sengaja dikosongkan)

BAB 1
PENDAHULUAN

1.1 Latar Belakang

Saat ini pengembangan teknologi pengukuran sangat luas. Salah satu bidang yang memanfaatkan teknologi pengukuran adalah bidang militer. Teknologi pengukuran dimanfaatkan oleh militer dalam sistem persenjataannya. Pemanfaatan teknologi pengukuran dapat diaplikasikan pada mesin persenjataan sehingga menjadi lebih akurat dibandingkan dengan militer manusia. Mesin ini nantinya diharapkan dapat menggantikan militer manusia sebagai usaha untuk meminimalisir korban penembakan yang disebabkan oleh kesalahan manusia.

Image processing sendiri merupakan metode pengolahan data berupa gambar untuk mendapatkan gambar yang lebih disempurnakan atau untuk memperoleh informasi yang dibutuhkan. Dalam pengukuran, *image processing* dapat digunakan untuk memperoleh data berupa jarak maupun posisi. *Input* yang berupa gambar atau video dapat diperoleh dengan menggunakan baik mono kamera maupun stereo kamera. *Image processing* sendiri terbagi menjadi 3 tahap, yaitu *pre-processing*, *processing*, dan *post-processing*. *Pre-processing* sendiri adalah proses *thresholding* dimana gambar disegmentasi agar dapat membedakan objek yang akan diproses dengan latar belakangnya. Kemudian *processing* merupakan tahap pendeteksian objek pada gambar. Tahap terakhir adalah *post-processing* dimana pengguna mengekstraksi informasi yang dibutuhkan dari gambar dengan. Gambar yang diperoleh dari kamera nantinya dapat diolah dengan menggunakan bantuan perangkat lunak, contohnya *Visual Studio*, untuk memperoleh informasi yang dibutuhkan, misalnya jarak suatu objek dari kamera.

Saat ini sudah ada alat pelontar peluru yang dapat menembak target secara otomatis dengan mengetahui koordinat dari target tersebut. Alat ini memanfaatkan *image processing* untuk mendeteksi dan mendapatkan data jarak serta koordinat x dan y

dari target. Dengan menggunakan 2 kamera (stereo kamera), alat ini dapat mengetahui jarak dan koordinat x dan y dari target. Kamera yang digunakan dalam alat ini memiliki posisi dimana kedua sumbu kameranya sejajar. Karena kedua sumbu kamera sejajar maka terdapat *blind area* yang cukup luas dimana target tidak dapat tertangkap oleh kedua kamera.

Berdasarkan permasalahan di atas maka digunakanlah suatu sistem pelacakan dan penentuan koordinat 3 dimensi dengan menggunakan stereo kamera yang memiliki sudut antara kedua sumbu kamera. Dengan memanfaatkan metode *image processing* dan perangkat stereo kamera, diusulkan sebuah aplikasi pendeteksian suatu target yang dapat menentukan jarak dan koordinat target dari kamera. Kedua kamera nantinya akan dibentuk sudut antara kedua sumbu kamera untuk memperkecil *blind area* di antara kedua kamera. Aplikasi ini nantinya akan digunakan pada alat pelontar peluru *auto-tracking* sehingga alat ini dapat mengetahui posisi dan jarak target secara otomatis ketika kamera menangkap gambar target.

1.2 Rumusan Masalah

Dari uraian pada latar belakang dapat dirumuskan masalah sebagai berikut:

1. Bagaimana mengalibrasi dua gambar hasil tangkapan dua kamera yang bersebelahan dengan sudut tertentu?
2. Bagaimana menentukan koordinat 3 dimensi objek terhadap kamera?

1.3 Batasan Masalah

Untuk menyederhanakan tugas akhir ini, maka rumusan masalah yang akan dibahas antara lain:

1. Menggunakan 2 kamera (*stereo camera*) yang posisinya tidak bergerak.
2. Objek yang dipakai adalah bola tenis berdiameter 6,5cm.

3. Menggunakan sistem pencahayaan *backlight indoor* pada waktu siang hari.
4. Terdapat *blind area* diantara 2 kamera.
5. Kedua kamera membentuk sudut β sebesar 15° dimana sudut β adalah sudut yang dibentuk antara kamera dengan garis sumbu z.
6. Hasil kalibrasi diasumsikan valid.

1.4 Tujuan Penelitian

Tujuan dari proposal tugas akhir ini adalah sebagai berikut:

1. Mengetahui cara mengalibrasi dua gambar hasil tangkapan dua kamera yang bersebelahan dengan sudut tertentu.
2. Dapat menentukan koordinat 3 dimensi objek terhadap kamera.

1.5 Manfaat Penelitian

Manfaat dari proposal tugas akhir ini adalah sebagai berikut:

1. Dapat digunakan sebagai alat ukur jarak.
2. Dapat dijadikan sebagai referensi dalam bidang *image processing* dalam penelitian berikutnya.
3. Dapat digunakan untuk pengembangan otomatisasi senjata-senjata militer.

(Halaman ini sengaja dikosongkan)

BAB 2
TINJAUAN PUSTAKA

2.1 Tinjauan Pustaka

Pesatnya perkembangan teknologi di bidang pengukuran, terutama teknologi pendeteksian objek, menghasilkan banyak eksperimen yang dilakukan untuk diaplikasikan dalam mempermudah kehidupan manusia. Pendeteksian mulai dari objek tunggal hingga *multiple object* serta menggunakan mono kamera hingga stereo kamera, semua sudah pernah diusulkan melalui *paper-paper*. Dari banyak *paper* yang membahas teknik pendeteksian menggunakan kamera, ada beberapa yang dapat dimanfaatkan dalam pendeteksian koordinat 3 dimensi objek tunggal dengan menggunakan stereo kamera.

Terdapat penelitian terlebih dahulu yang membahas pendeteksian objek 2D pada sebuah gambar. Penelitian ini melakukan pendeteksian banyak objek dengan menggunakan metode pendeteksian acuan range warna objek pada setiap *frame*. Metode ini membutuhkan sampel sebanyak 1000 *frame* dengan kecepatan 50 *fps* (*frame per second*). Ukuran agambar yang digunakan pada penelitian ini sebesar 352 x 288 piksel. Objek yang digunakan pada penelitian ini adalah bola dan pendeteksiannya dilakukan dengan pendekatan histogram warna RGB (*Red Green Blue*). Namun penelitian ini tidak dapat digunakan untuk menentukan jarak objek dengan kamera karena hanya menggunakan mono kamera [3].

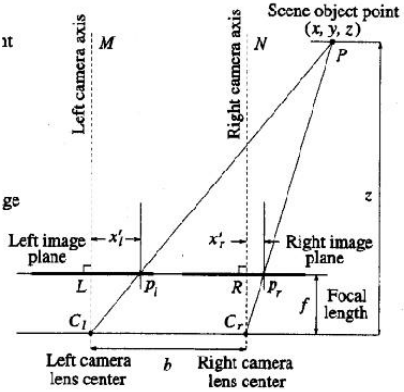
Terdapat juga penelitian yang mengukur jarak sumber cahaya terhadap kamera dengan metode perbandingan jumlah piksel. Dalam penelitian ini, sumber cahaya berasal dari laser berwarna hijau yang kemudian ditangkap oleh kamera. Hasil gambar yang ditangkap oleh kamera nanti diubah menjadi gambar biner, dimana gambar cahaya yang tertangkap menjadi berwarna putih. Dari gambar biner inilah nanti banyaknya piksel berwarna putih akan dihitung. Banyaknya nilai piksel dari tiap gambar ini

yang akan diproses dengan menggunakan Matlab untuk mengestimasi jarak sumber cahaya terhadap kamera [4].

Selain itu, ada juga penelitian yang membahas pendeteksian dan pengukuran jarak suatu objek dengan menggunakan stereo kamera. Penelitian ini dapat mengestimasi jarak suatu objek terhadap kamera dengan menggunakan metode *triangulation*. Penelitian ini menggunakan perangkat lunak *visual studio* dengan menggunakan bahasa C yang diperoleh dari *library* openCV. Parameter yang digunakan dalam penelitian ini adalah nilai *focal length* dari kamera. Nilai *focal length* sendiri dapat diperoleh dari program kalibrasi yang telah disediakan oleh *openCV*. Kekurangan dari penelitian ini adalah daerah yang tidak tertangkap oleh kedua kamera (*blind spot*) terlalu luas sehingga tidak dapat mendeteksi objek jarak dekat [1].

2.2 Dasar Teori

2.2.1 *Stereo Vision*



Gambar 2.1 Titik *scene* dari kamera stereo
Sumber : R. Jain, 1995, "Machine Vision"

Stereo vision merupakan sebuah teknik untuk mendapatkan impresi kedalaman gambar yang diperoleh dari dua alat penglihatan. Teknik ini memampukan pengguna untuk menentukan jarak suatu objek dari alat penglihatan. Selain itu,

teknik ini juga dapat digunakan untuk memperoleh informasi bentuk 3D dari suatu objek yang tertangkap secara 2D pada dua buah kamera. Pemanfaatan teknik ini akhirnya banyak diaplikasikan dalam pengembangan sensor dan robotika.

Stereo vision yang memanfaatkan dua buah kamera dapat menghitung jarak objek terhadap kamera dengan melibatkan beberapa parameter. Parameter-parameter yang dibutuhkan untuk mengukur jarak adalah *focal length* (*f*), jarak antar kamera (*b*) serta jarak antara pusat proyeksi kamera dengan garis imajinasi (*x'l*, *x'r*). *Focal length* adalah jarak antar kamera dengan sensor digital dimana objek terbentuk. Nilai *focal length* tiap kamera dapat diperoleh dari hasil kalibrasi kamera yang berupa matrik. Nilai matrik ini berisikan informasi nilai intrinsik kamera, yaitu nilai *focal length* dan pusat lensa dari tiap kamera (Cl, Cr).

2.2.2 Gambar Digital

Gambar merupakan representasi objek fisik 3 dimensi kedalam bentuk 2 dimensi. Bentuk dari gambar sendiri bermacam-macam, ada yang berupa gambar berwarna, ada yang hanya berupa gambar hitam-putih. Kumpulan gambar yang ditampilkan dengan kecepatan *frame per second* tertentu akan membentuk suatu video. Gambar digital merupakan gambar dua dimensi yang dapat ditampilkan pada layar komputer dalam bentuk piksel.

Gambar digital tersusun dari piksel-piksel berbentuk matrik dari fungsi intensitas cahaya dengan ukuran M x N. Fungsi intensitas cahaya dapat ditulis sebagai f(x,y) yang mana f merupakan nilai amplitudo pada koordinat x dan y. Secara matematis, gambar digital dapat dituliskan seperti persamaan (1):

$$f(x,y) = \begin{bmatrix} f(0,0) & f(0,1) & \dots & f(0,N-1) \\ f(1,0) & f(1,1) & \dots & f(1,N-1 \\ \vdots & \vdots & & \vdots \\ f(M-1,0) & f(M-1,1) & \dots & f(M-1,N-1) \end{bmatrix} \quad \text{.....(1)}$$

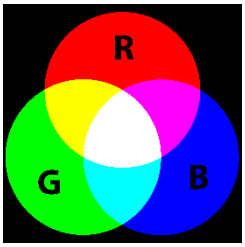
2.2.3 Kedalaman Gambar

Nilai yang biasa digunakan untuk menentukan kualitas suatu gambar adalah banyaknya piksel yang dimiliki suatu gambar. Semakin banyak atau padat piksel yang ada pada suatu gambar maka kualitas atau resolusi yang dimiliki gambar tersebut juga semakin meningkat. Piksel sendiri merupakan representasi titik terkecil dari suatu gambar yang memiliki informasi yang nantinya dapat diproses.

Berdasarkan warna-warna penyusunnya, gambar digital dapat dibagi menjadi tiga macam gambar, yaitu:

1. **Gambar Berwarna**

   Gambar berwarna merupakan gambar yang memiliki informasi warna di setiap pikselnya. Informasi warna ini dapat dibentuk dari gabungan komponen-komponen satu set *channel* warna. Gambar berwarna memiliki format gambar RGB (*Red Green Blue*) yang mana warna yang dihasilkan tiap pikselnya terdiri dari gabungan informasi tingkatan nilai dasar merah, hijau dan biru.



Gambar 2.2 Warna RGB

Sumber : https://bpiinc.files.wordpress.com/2011/11/

Gambar warna (8 bit) merupakan gambar yang tiap pikselnya memiliki jumlah warna maksimum yang dapat digunakan sebanyak 256 warna. Setiap piksel pada gambar warna diwakili oleh warna yang merupakan kombinasi dari tiga warna dasar yaitu merah, hijau, dan biru.

Gambar warna (16 bit) atau biasa disebut gambar *highcolor* merupakan gambar yang tiap pikselnya diwakili dengan 65.536 warna (2 *byte memory*). Dalam formasi bitnya, nilai merah dan biru mengambil tempat 5 bit di kanan dan kiri. Komponen hijau memiliki 5 bit ditambah 1 bit ekstra.

Gambar warna (24 bit) merupakan gambar yang tiap pikselnya diwakili dengan 16.777.216 variasi warna. Tingkt variasi ini mampu untuk memvisualisasikan seluruh warna yang dapat dilihat oleh penglihatan manusia. Setiap poin informasi pixel (RGB) disimpan ke dalam 1 *byte* data dengan 8 bit pertama menyimpan nilai biru, kemudian diikuti dengan nilai hijau pada 8 bit kedua dan pada 8 bit terakhir merupakan nilai warna merah.

2. Gambar *Grayscale*

Gambar *grayscale* merupakan gambar yang mana pikselnya memiliki informasi nilai tingkat keabuan atau intensitas warna putih pada gambar. Tingkat intensitas terendah akan menghasilkan warna hitam sedangkan tingkat intensitas tertinggi akan menghasilkan warna putih. Pada umumnya gambar *grayscale* memiliki kedalam piksel 8 bit, dengan 256 derajat keabuan, namun ada juga yang memiliki kedalam piksel 16 bit. Gambar *grayscale* dengan kedalaman piksel 16 bit biasanya digunakan pada gambar yang membutuhkan ketelitian yang tinggi.

Gambar 2.3 *Mother Theresa in Grayscale*
Sumber: S. Jayaraman, "Digital Image Processing", page 21

Seperti ditunjukkan pada gambar 2.3, dapat dilihat bahwa gambar hanya terdiri dari warna abu-abu dengan tingkat intensitas yang berbeda. Ada yang memiliki tingkat intensitas keabuan yang rendah sehingga terlihat mendekati warna hitam. Ada juga yang memiliki tingkat intensitas keabuan yang tinggi sehingga warnanya mendekati warna putih.

Untuk mengubah gambar dengan format RGB menjadi format *grayscale* dapat menggunakan metode sebagai berikut:

$$\frac{R+G+B}{3} \quad \text{.....(2)}$$

Dimana : R = nilai warna merah pada piksel
G = nilai warna hijau pada piksel
B = nilai warna biru pada piksel

Dengan menggunakan metode di atas pada setiap piksel sebuah gambar maka akan diperoleh gambar dengan format *grayscale*.

3. Gambar Biner

Gambar biner merupakan gambar yang hanya memiliki dua macam warna, yaitu putih dan hitam. Pada gambar biner, setiap piksel hanya memiliki 2 macam nilai, yaitu 1 untuk warna hitam dan 0 untuk warna putih. Pada gambar 2.4 dapat dilihat gambar dengan format biner, yang mana gambar hanya memiliki warna hitam dan putih.

Gambar 2.4 *Mother Theresa in Binary Image*
Sumber: S. Jayaraman, "Digital Image Processing", page 20

Selain tiga macam yang telah disebutkan sebelumnya, ada juga beberapa macam gambar digital lainnya, antara lain:

Gambar *Threshold*

Thresholding merupakan proses segmentasi gambar. Segmentasi gambar sendiri digunakan untuk membedakan antara objek dengan latar belakangnya. Proses segmentasi sendiri dilakukan dengan mengisolasi objek pada gambar yang berbentuk format *grayscale* menjadi format biner. Pada saat melakukan proses *thresholding* perlu diperhatikan nilai *threshold* yang akan ditentukan. Nilai piksel yang berada di bawah nilai *threshold* akan diubah menjadi warna hitam,

sedangkan piksel yang memiliki nilai di atas nilai *threshold* akan diubah menjadi warna putih.

Gambar 2.5 Gambar *grayscale* dan gambar proses *threshold*
Sumber: https://www.mathworks.com/help/images/ref/multithresh.html

Gambar HSV

Pada pengolahan suatu warna citra terdapat berbagai macam model warna. Model warna RGB (Red,Green,Blue) merupakan model warna yang paling sering digunakan. Contoh penggunaan warna ini yaitu pada hasil layar tangkapan kamera. Model warna berikutnya yaitu HSV yang terdi dari 3 komponen yaitu Hue, Saturation, Value.

1. *Hue* merupakan suatu ukuran panjang gelombang yang terdapat pada warna dominan yang diterima oleh penglihatan. Hue menyatakan warna yang sebenarnya, seperti merah, violet, kuning dan digunakan untuk menentukan kemerahan (*redness*), kehijauan (*greenness*) dan sebagainya.
2. *Saturation* atau *chroma* adalah kemurnian atau kekuatan warna, ukuran banyaknya cahaya putih yang bercampur pada *hue*.
3. *Value* adalah nilai kecerahan dari warna. Nilainya mulai dari 0 – 100 %. Apabila nilainya 0 maka warna yang dihasilkan menjadi hitam, semakin besar nilai maka

semakin cerah dan muncul variasi – variasi baru dari warna tersebut

Untuk mengubah citra HSV menjadi citra RGB bisa digunakan persamaan seperti berikut ini:

$$V = \max(r, g, b)$$

$$S = \begin{cases} 0, & v = 0 \\ 1 - \dfrac{\min(r, g, b)}{v}, & v > 0 \end{cases}$$

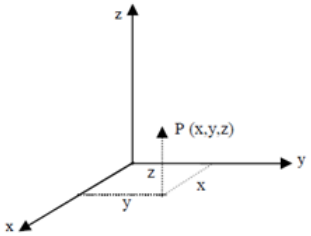
$$H = \begin{cases} 0, jika\ S = 0 \\ \frac{60*(g-b)}{s*v}, jika\ v = r \\ 60*\left[2+\frac{b-r}{s*v}\right], jika\ v=g \\ 60*\left[4+\frac{r-g}{s*v}\right], jika\ v=b \end{cases} \quad \text{.....(3)}$$

Jika Saturation S=0, maka hue tidak terdefinisi atau dengan kata lain tidak memiliki hue berarti *monochrome*. Hue (H) lalu dikonversi menjadi derajat/degrees dengan cara mengalikan dengan 60 sehingga menghasilkan HSV dengan S dan V antara 0 dan 1 dan H antara 0 – 360.

2.2.4 Koordinat 3 Dimensi

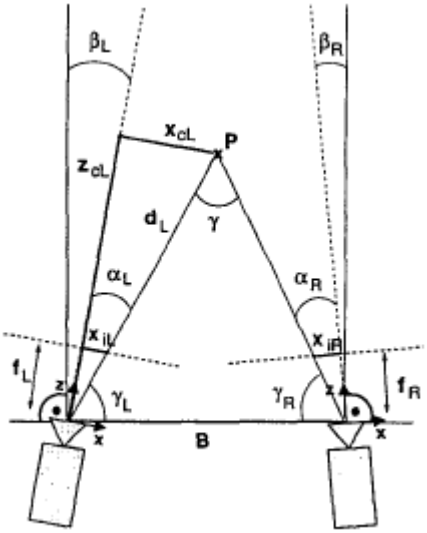
Sistem koordinat kartesius merupakan sistem yang digunakan untuk menentukan posisi suatu titik, garis maupun gambar dalam suatu bidang baik 2 dimensi maupun 3 dimensi. Dalam sistem koordinat 2 dimensi, digunakan dua garis yang saling tegak lurus dalam satu bidang sebagai sumbu koordinat. Sumbu yang ke arah horizontal diberi label axis-x, sedangkan sumbu yang ke arah vertikal diberi label axis-y. Dalam bidang 3 dimensi, ditambahkan satu sumbu lagi yang arahnya tegak lurus terhadap bidang xy. Sumbu yang tegak lurus terhadap bidang xy ini diberi label sumbu z. Sumbu-sumbu pada

koordinat kartesian ini bersifat orthogonal (antar sumbu saling tegak lurus).



Gambar 2.6 Sistem koordinat 3 dimensi
Sumber : https://zenosoft.files.wordpress.com/2014/05/42.png

Untuk menentukan koordinat 3 dimensi dari objek yang ditangkap oleh kamera dapat menggunakan metode *triangulation algorithm*.



Gambar 2.7 Pemodelan sederhana sistem stereo kamera
Sumber: Visual determination of 3D grasping points on unknown objects with a binocular camera system

Berdasarkan gambar 2.8, parameter-parameter yang dibutuhkan untuk mencari nilai koordinat titik P antara lain, nilai *focal length* (*f*), jarak titik P terhadap sumbu pusat kamera di bidang fokus kamera (x_i), jarak titik P terhadap sisi atas gambar yang tertangkap kamera (y_i), sudut antara kamera terhadap garis vertikal (β) serta jarak antar kedua kamera (B). Dari parameter-parameter tersebut dapat diperoleh nilai parameter-parameter lainnya dengan menggunakan rumus:

$$\alpha = arctan\frac{x_i}{f} \quad \text{.....(4)}$$

$$d = \frac{B\cos(\alpha_L+\beta_L)}{\tan(\alpha_L+\beta_L+\alpha_R+\beta_R)} + B\sin(\alpha_L+\beta_L) \quad \text{.....(5)}$$

$$z_c = d\cos\alpha \quad \text{.....(6)}$$

$$x_c = \frac{x_i \times z_c}{f} \quad \text{.....(7)}$$

$$y_c = \frac{y_i \times z_c}{f} \quad \text{.....(8)}$$

Dimana: α = sudut antara titik P dengan sumbu tengah kamera
d = jarak antara titik P dengan kamera
z_c = garis proyeksi jarak antara titik P dengan kamera terhadap sumbu tengah kamera
x_c = koordinat aktual titik P terhadap sumbu x
y_c = koordinat aktual titik P terhadap sumbu y

(Halaman ini sengaja dikosongkan)

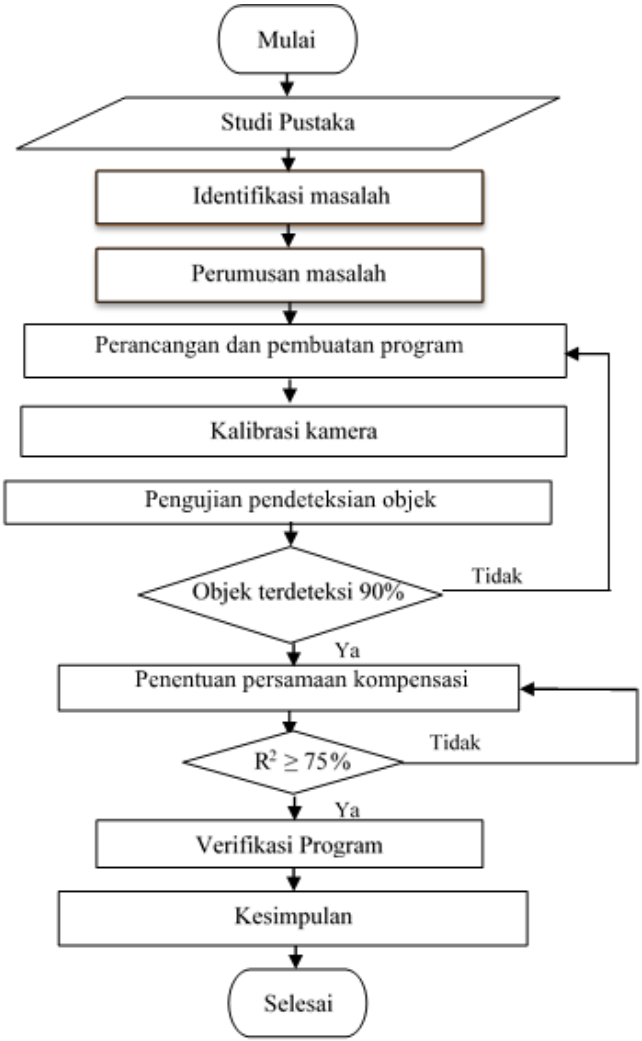
BAB 3
METODE PENELITIAN

Pada Bab 3 ini akan dibahas mengenai langkah-langkah sistematis yang akan dijadikan sebagai acuan kerangka penelitian yang bertujuan untuk menciptakan aplikasi pendeteksi, dan penentu koordinat 3 dimensi suatu objek. Selain itu juga dijelaskan parameter-parameter proses yang akan digunakan dalam penelitian ini.

3.1 Flowchart Metodologi Penelitian

Metodologi yang akan digunakan dalam penelitian ini secara garis besar terdiri dari tiga proses utama, yaitu proses persiapan, proses perancangan program serta proses analisa data. Proses persiapan terdiri dari studi pustaka yang dilakukan untuk mempelajari dasar-dasar dari penelitian, kemudian dilakukan identifikasi serta perumusan masalah. Setelah proses persiapan, dilakukan proses perancangan program, yang mana hasil pendeteksian dan pengukuran jarak dapat diperoleh dari program tersebut. Setelah memperoleh hasil pendeteksian, lalu memasuki proses akhir yaitu analisa data, yang mana dilakukan pembahasan hasil penelitian sehingga dapat ditarik kesimpulan dari tugas akhir ini.

`



Gambar 3.1 Diagram alir metodologi penelitian

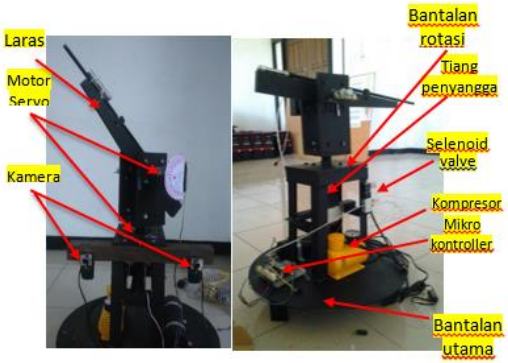
3.2 Proses Persiapan

Proses persiapan terdiri dari tiga tahapan yaitu studi pustaka, identifikasi masalah dan perumusan masalah. Studi

pustaka merupakan acuan referensi yang penulis gunakan untuk mendalami permasalahan yang akan diteliti, yang mana dalam hal ini mengenai pembuatan aplikasi program pendeteksi dan penentu koordinat 3 dimensi suatu objek. Studi pustaka dilakukan dengan memanfaatkan buku-buku referensi, *paper-paper* dan jurnal-jurnal yang berkaitan dengan permasalahan yang akan dibahas. Setelah itu dilakukan identifikasi masalah untuk menyusun sistem pendeteksi dan penentu koordinat 3 dimensi suatu objek. Terakhir, tahap perumusan masalah berisi perancangan program untuk pendeteksi dan penentu koordinat 3 dimensi suatu objek.

3.3 Proses Perancangan Program

Pada proses ini dilakukan perancangan program kalibrasi stereo kamera serta perancangan program untuk pendeteksian dan penentuan koordinat x, y, dan z objek terhadap dua kamera. Program yang akan dirancang penulis ini nantinya akan diaplikasikan pada *prototype* alat pelontar peluru yang mana skema dari alat tersebut dapat dilihat pada gambar 3.2 sebagai berikut:



Gambar 3.2 *Prototype* Alat Pelontar Peluru beserta bagian – bagiannya

Seperti yang terlihat pada gambar 3.2, *prototype* alat pelontar peluru tersebut terdiri dari beberapa bagian seperti

selongsong peluru, landasan utama, bantalan rotasi, kamera, motor, kompresor dan lain-lain. *Prototype* ini memiliki bantalan utama berupa plat besi setebal 50 mm yang berfungsi sebagai pondasi tempat dipasangnya tiang penyangga untuk bantalan rotasi dan tempat dipasangnya *selenoid valve*. Bantalan rotasi pada alat ini berhubungan langsung dengan motor penggerak pelontar peluru yang mana arah geraknya searah sumbu x. Selain itu terdapat motor penggerak yang dipasang pada sisi kiri pangkal laras yang berfungsi untuk menggerakkan laras naik dan turun (membentuk sudut elevasi). Secara garis besar alat pelontar peluru ini bergerak mengikuti target dengan memanfaatkan *input* gambar yang tertangkap oleh stereo kamera akan diproses dengan menggunakan program yang dihasilkan dari Tugas Akhir ini. *Input* yang dihasilkan oleh Tugas Akhir ini berupa koordinat x, y dan z dari objek.

3.3.1 Cara Kerja

Cara kerja program yang harus dibuat adalah pertama mendeteksi objek yang merupakan bola berwarna kuning dalam bentuk gambar 2D. Kemudian data gambar yang didapat dari stereo kamera ini dianalisa dan diproses menggunakan program C++ dari *library* openCV 2.1.0 dengan bantuan *Software* Visual Studio 2015. Setelah itu dilakukan proses kalibrasi kamera untuk mendapatkan nilai *focal length* dari kedua kamera. Dengan memanfaatkan nilai *focal length* yang didapat dari hasil kalibrasi, nilai sudut (α) antara objek dengan sumbu tengah kamera dapat ditentukan dengan menggunakan trigonometri. Dari nilai α serta jarak antar kamera (d), maka akan didapat nilai z dengan memanfaatkan trigonometri. Setelah mendapat nilai z, nilai x dan y dapat ditentukan dengan menggunakan sistem triangulation. Hasil koordinat x, y dan z diserialkan agar bisa diaplikasikan pada *prototype* alat pelontar peluru.

3.3.2 Proses dan Pengerjaan

Ada beberapa tahapan dalam proses dan pengerjaan perancangan program, yaitu pendektesian objek, penghitungan jarak objek terhadap 2 kamera (koordinat z) dan penentuan koordinat x dan y objek terhadap kedua kamera.

1. Kalibrasi stereo kamera

   Program kalibrasi kamera dibuat untuk mengetahui nilai intrinsik tiap kamera. Data – data yang didapat yaitu *focal length* (*f*), *imaging 2D coordinate system* (*u, v*), titik gambar 2D kamera kanan dan kiri, dan matriks kamera kanan dan kiri (opsional).

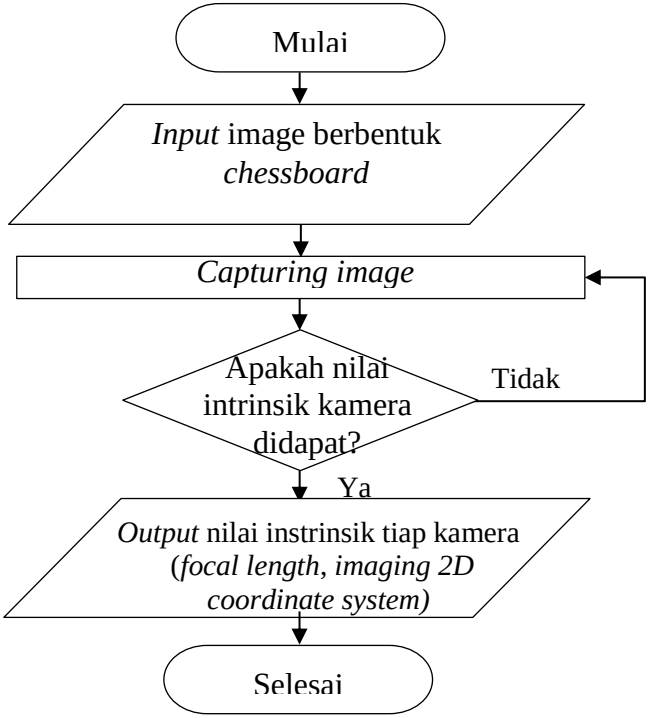
2. Pendektesian dan Penetuan Koordinat x, y dan z Objek

   Pendektesian objek dilakukan dengan cara mengatur satu nilai warna sesuai yang diinginkan, yang mana dalam penelitian ini menggunakan warna kuning, pada program agar pada saat program dijalankan objek dengan warna tersebut dapat dideteksi. Langkah – langkah yang perlu dilakukan untuk mendeteksi objek yaitu mengambil gambar yang didalamnya terdapat objek tersebut. Kemudian dilakukan proses *image processing* untuk mengubah gambar yang sebelumnya memiliki format RGB ke bentuk format HSV. Gambar yang sudah berupa HSV dikerjakan proses *thresholding* untuk menghasilkan citra biner dimana warna kuning menjadi warna hitam dan warna lainnya menjadi warna putih, seperti yang ditunjukkan pada gambar 3.3.

Gambar 3.3 Gambar asli dengan gambar hasil *threshold*

Penentuan koordinat x, y dan z objek terhadap 2 kamera dilakukan dengan memanfaatkan nilai intrinsik hasil dari kalibrasi kamera. Kalibrasi cukup dilakukan sekali selama fokus lensa tiap kamera tidak diubah. Jika koordinat z sudah diketahui, koordinat x dan y bisa ditentukan juga dengan menggunakan persamaan trigonometri. Nantinya koordinat x, y dan z dari objek terhadap 2 kamera diserialkan agar bisa diproses untuk menjadi data masukan pada alat pelontar peluru.

3.3.3 *Flowchart* Perancangan Program Kalibrasi Stereo Kamera

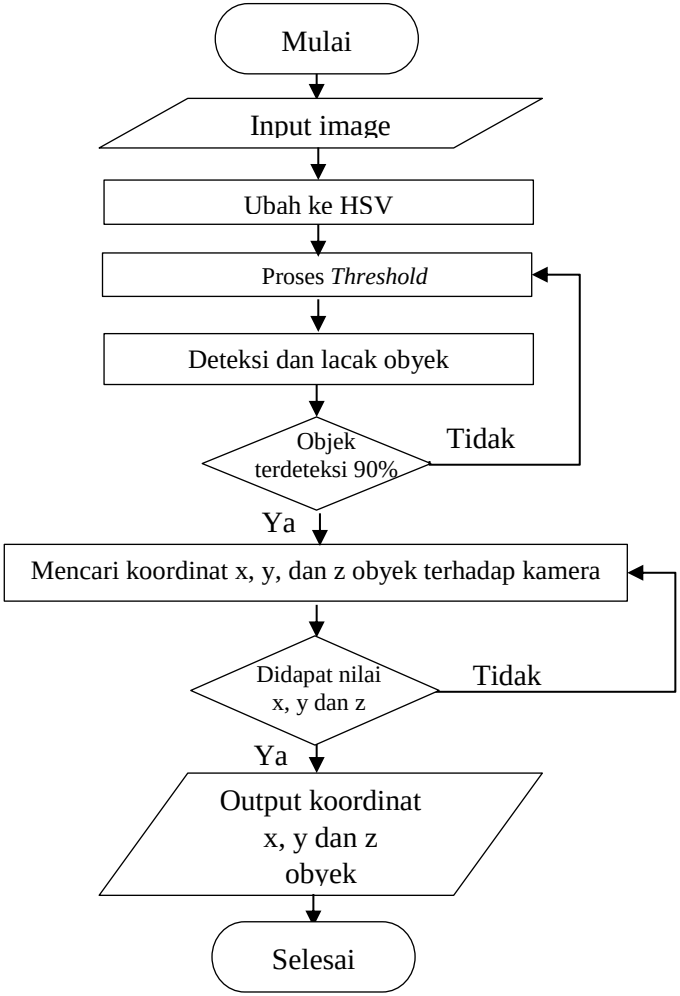


Gambar 3.4 Diagram alir rancangan program Kalibrasi Stereo Kamera

Seperti yang dapat dilihat pada gambar 3.4, proses kalibrasi ini digunakan untuk mengetahui nilai instrinsik tiap kamera. Hasil dari kalibrasi ini berupa matrik 3x3 yang berisi nilai *focal length* (*f*) serta titik pusat (u_0, v_0) tiap kamera.

$$K = \begin{matrix} f & 0 & u_0 \\ 0 & f & v_0 \\ 0 & 0 & 1 \end{matrix} \quad \text{.....(10)}$$

3.3.4 *Flowchart* Perancangan Program Pendeteksi dan Penentu Koordinat



Gambar 3.5 Diagram alir rancangan program Pendeteksi Objek

Pada diagram alir yang ditunjukkan pada gambar 3.5 dapat dilihat bahwa proses yang pertama kali dilakukan yaitu melakukan *image processing*. Gambar *input* yang memiliki format RGB diubah terlebih dahulu diubah menjadi gambar berformat HSV. Kemudian dari gambar tersebut dilakukan proses *thresholding* untuk merubah gambar menjadi bentuk biner, dimana objek diubah menjadi warna hitam dan latar belakang diubah menjadi warna putih. Dari proses tersebut objek akan dapat terdeteksi. Hasil pendeteksian tersebut dapat dimanfaatkan untuk memperoleh koordinat 3 dimensi dari objek.

3.4 Tahap Implementasi

Tahap implementasi merupakan tahap pengimplementasian *software* yang telah dirancang pada alat pelontar peluru. Dilakukan *running* program dengan berbagai variasi posisi objek. Kemudian dari hasil deteksi objek tersebut didapatkan data berupa koordinat x, y dan z objek dari berbagai posisi. Data ini kemudian dijadikan bahan analisa untuk menentukan persamaan hubungan antara koordinat objek terhadap 2 kamera dengan koordinat objek terhadap *prototype* alat pelontar peluru.

(Halaman ini sengaja dikosongkan)

BAB 4
PERANCANGAN DAN PEMBUATAN PROGRAM

4.1 Implementasi Program dalam Sistem

Program pendeteksian dan penentuan koordinat 3 dimensi objek dibuat dengan menggunakan teknologi *computer vision*. Perangkat lunak yang digunakan dalam pembuatan program adalah Visual Studio Community 2015 dan OpenCV 2.1. Mesin pengolah yang digunakan untuk menjalanan program adalah *personal computer* dengan spesifikasi sebagai berikut:

a. Sony VAIO Fit E14 model SVF143A1YW
b. *Processor* Intel® Core™ i5-4200U CPU @ 1.6GHz
c. *Memory* (RAM) 8 GB
d. *Operating System* Windows 8.1 (64-bit)

Untuk mengetahui performa dari program pendeteksian dan penentuan koordinat 3 dimensi objek yang dibangun maka perlu dilakukan pengujian. Pengujian dilakukan pada kondisi tertentu. Objek yang digunakan adalah bola tenis berdiameter 6,5 cm. Pengambilan data dilakukan di siang dan sore hari secara *real-time*.

4.2 Konstruksi Program

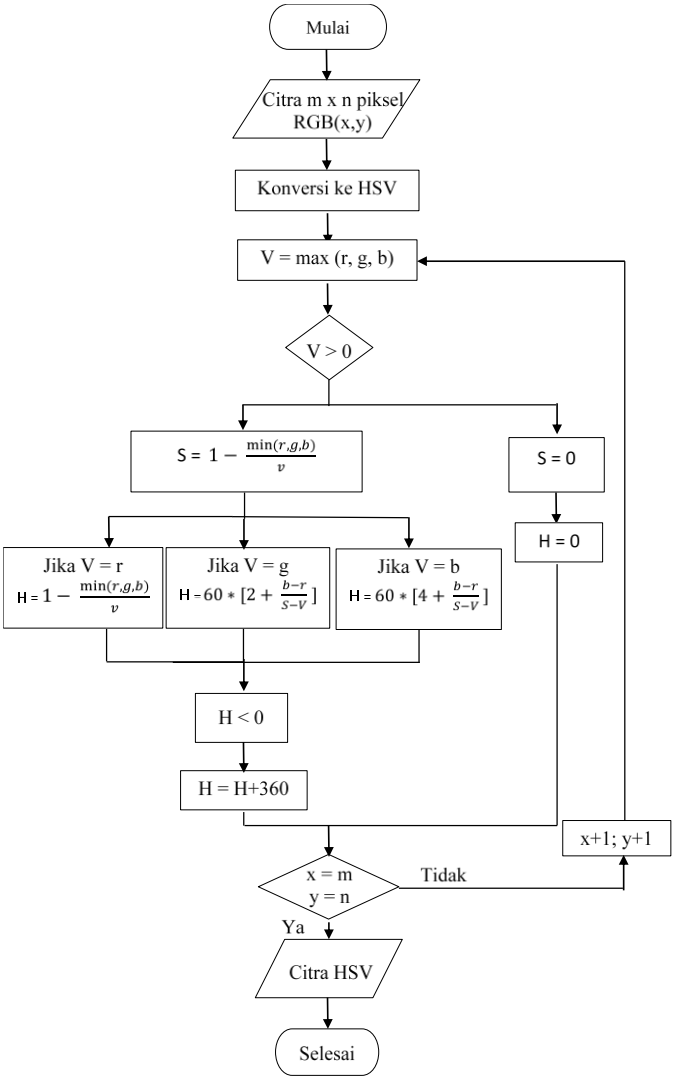
Pada tahap konstruksi program dibahas mengenai tahap-tahap penerapan metode yang digunakan dalam pembuatan program pendeteksian dan penentu koordinat 3 dimensi objek. Proses konstruksi pogram dimulai dari tahap pendeteksian objek. Tahap selanjutnya dilakukan penentuan jarak objek terhadap kamera. Tahap terakhir adalah penentuan koordinat x serta y dari objek. Konstruksi program dibangun ke dalam kode program menggunakan perangkat yang telah ditentukan. Program dibangun dalam bentuk kode program dengan menggunakan Visual Studio Community 2015, *library* OpenCV, serta bahasa pemrograman C++.

4.2.1 Program Pendeteksian Objek

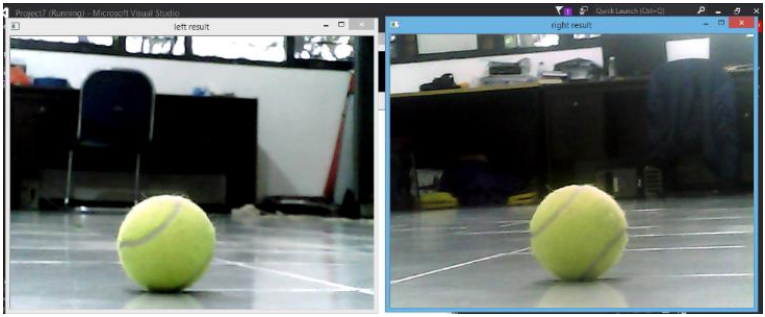
Kode program 1 : *RGB to HSV*

```
IplImage *HSV=cvCreateImage(cvSize(frame->width,frame->height),8,3);
IplImage *HSV2=cvCreateImage(cvSize(frame2->width,frame2->height),8,3);
cvCvtColor(frame,HSV,CV_BGR2HSV);
cvCvtColor(frame2,HSV2,CV_BGR2HSV);
```

Pada tahap pendeteksian objek, proses pertama yang dilakukan adalah mengubah gambar masukan hasil tangkapan kamera dari bentuk RGB menjadi bentuk HSV. Hal ini dilakukan karena hasil *thresholding* yang diperoleh dari gambar HSV lebih baik dibandingkan dengan hasil *thresholding* yang diperoleh dari gambar RGB.



Gambar 4.1 Diagram alir proses pengolahan dari gambar RGB ke HSV



Gambar 4.2 Gambar RGB sebagai gambar masukan *frame* kiri dan kanan

Frame hasil dari tangkapan kamera USB berbentuk format gambar RGB (*Red Gren Blue*) dengan resolusi 640 x 480 *pixel* seperti yang terlihat pada Gambar 4.2. Hasil dari *frame* tersebut akan diproses menjadi citra HSV.

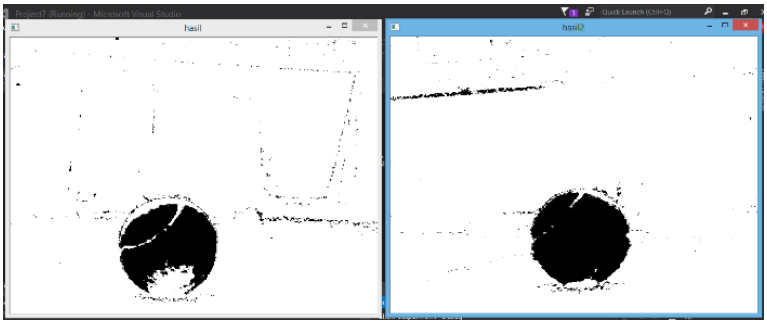
Persamaan (3) digunakan untuk merubah dari gambar RGB ke HSV secara manual. Gambar 4.1 adalah diagram alir dari proses pengolahan citra RGB ke HSV, pada gambar tersebut dapat dilihat proses perubahan dari citra RGB menjadi citra HSV.

Kode Program 2 : *Threshold*

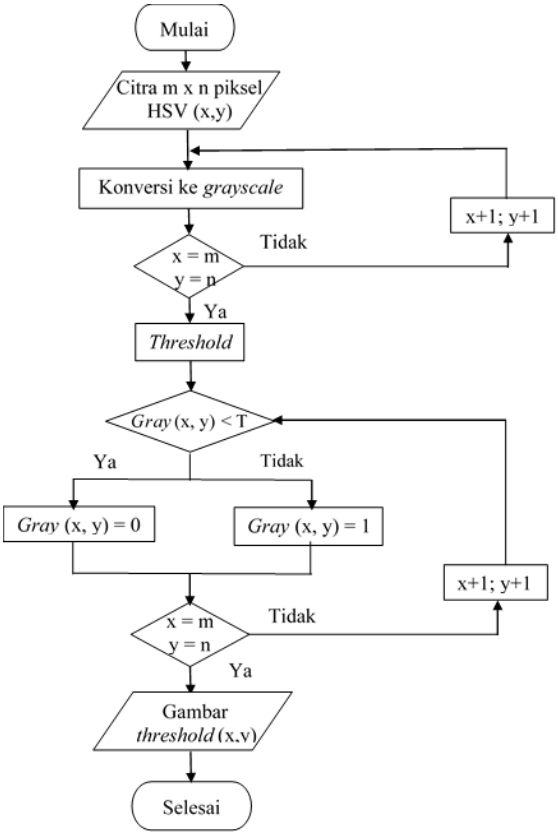
```
IplImage *thres=cvCreateImage(cvSize(frame->width,frame->height),8,1);
IplImage *thres2=cvCreateImage(cvSize(frame2->width,frame2->height),8,1);
cvInRangeS(HSV, cvScalar(hl,sl , vl), cvScalar(hh, sh, vh), thres);
cvInRangeS(HSV2, cvScalar(hl,sl , vl), cvScalar(hh, sh, vh), thres2);
```

Proses kedua dari tahap pendeteksian objek adalah *thresholding* pada gambar HSV. Tipe *threshold* yang digunakan adalah *binary threshold* dengan *range* nilai HSV minimum (0, 0, 0) - maksimum (179, 255, 255) baik untuk HSV *low* dan HSV *high*. Pada gambar 4.4 dapat dilihat diagram alir dari proses *threshold* pada gambar HSV.

Hasil perubahan gambar HSV menjadi gambar biner dengan tipe *binary threshold* dapat dilihat pada gambar 4.3. Gambar hasil *threshold* hanya terdapat warna hitam dan putih seperti terlihat pada gambar 4.3. Warna hitam merepresentasikan objek yang akan dideteksi sedangkan warna putih merepresentasikan latar belakang dari objek. Namun terkadang hasil yang diperoleh kurang sempurna. Bentuk hasil *threshold* dari gambar bola tenis tampak kurang sempurna seperti terlihat pada gambar 4.3.



Gambar 4.3 Gambar *Threshold*

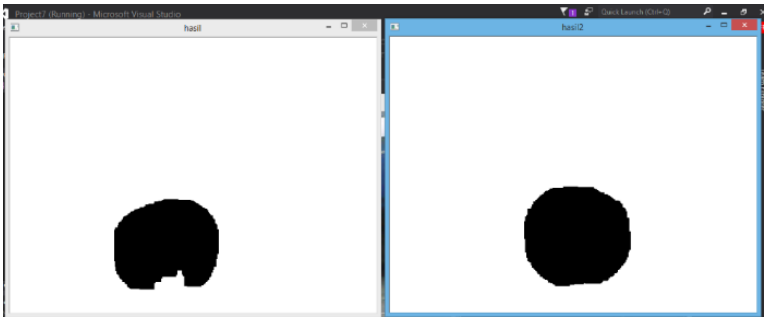


Gambar 4.4 Diagram alir proses *threshold*

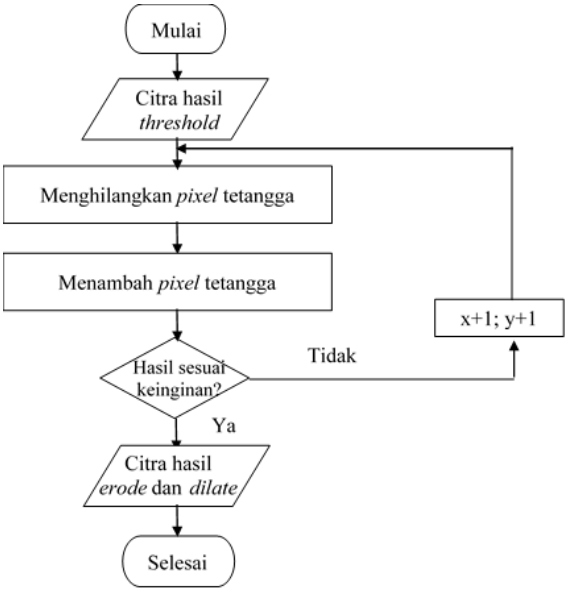
Kode Program 3 : *Erode and Dilate*

```
cvErode(thres,thres,element,ero1);
cvDilate(thres,thres,element,dil1);
cvErode(thres,thres,element,ero2);
cvDilate(thres,thres,element,dil2);
```

Proses *noise filtering* juga diperlukan pada proses *thresholding*. Diagram alir dari proses *noise filtering* dapat dilihat pada Gambar 4.6. Proses ini dilakukan karena pada proses *threshold* biasanya terdapat gangguan/*noise* pada hasil gambar yang diperoleh. *Noise filtering* bertujuan untuk membuang atau mengurangi gangguan tersebut agar gambar *threshold* yang diperoleh sesuai dengan yang diinginkan. Proses *noise filtering* terdiri dari *erode* dan *dilate*, dimana *erode* bertujuan untuk mengurangi *pixel* sedangkan *dilate* bertujuan untuk menambah *pixel*.

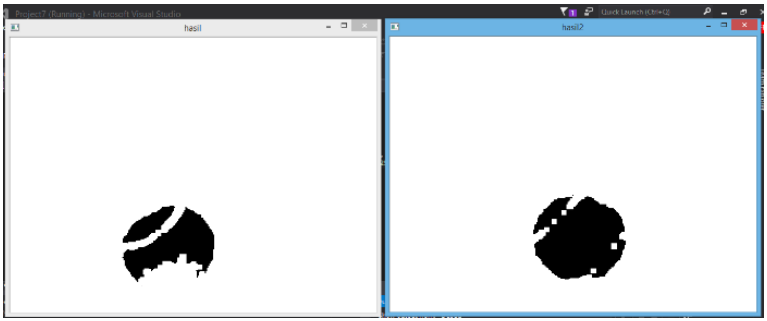


Gambar 4.5 Gambar hasil *threshold* yang telah diproses *noise filtering*

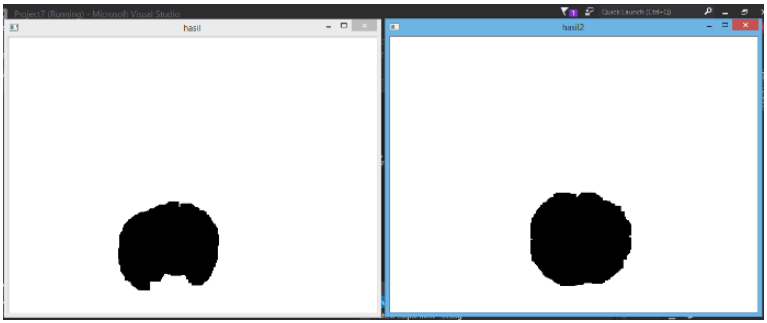


Gambar 4.6 Diagram alir proses *erode* dan *dilate*

Perbedaan pada gambar biner sebelum dan sesudah dilakukan proses *noise filtering* dapat dilihat pada gambar 4.7 dan gambar 4.8. Perubahan pada gambar *threshold* yang dilakukan proses *erode* dapat dilihat pada gambar 4.7. *Erode* dilakukan dengan pengecilan 1 *pixel* tetangga. Gambar *threshold* yang sudah di-*erode* kemudian dilakukan dilate dengan tiap 2 *pixel* tetangga dari *pixel* terluar. Hasil dari *dilate* tersebut dapat dilihat seperti pada Gambar 4.8.



Gambar 4.7 Gambar *threshold* yang telah dilakukan proses *erode*



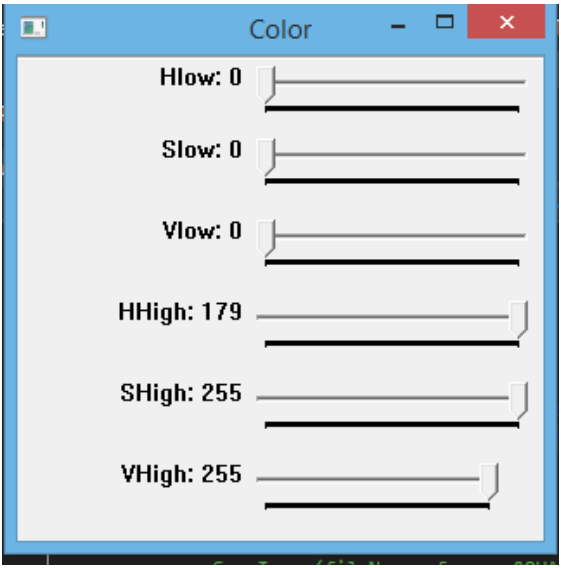
Gambar 4.8 Gambar *threshold* yang dilakukan proses *dilate*

Kode Program 4 : Mencari Range Warna Threshold

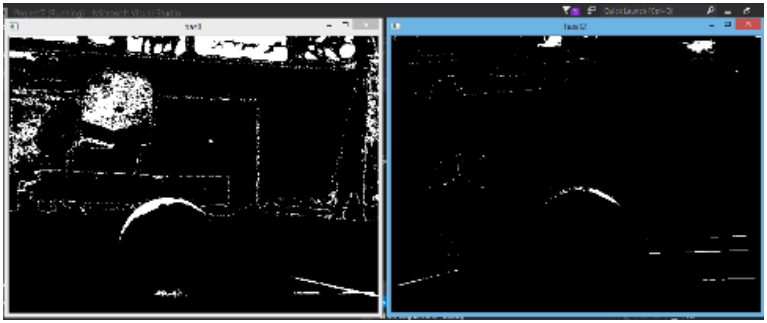
```
cvNamedWindow("Color",1);
cvCreateTrackbar("Hlow","Color", &hl,179,0);
cvCreateTrackbar("Slow","Color", &sl,255,0);
cvCreateTrackbar("Vlow","Color", &vl,255,0);
cvCreateTrackbar("HHigh","Color", &hh,179,0);
cvCreateTrackbar("SHigh","Color", &sh,255,0);
cvCreateTrackbar("VHigh","Color", &vh,255,0);
```

Proses berikutnya adalah mencari nilai *range* HSV untuk warna objek yang akan dideteksi pada gambar yang telah berformat HSV. *Trackbar* yang digunakan untuk mencari nilai *range* HSV untuk mengubah gambar HSV menjadi gambar *threshold* dapat dilihat pada gambar 4.9.

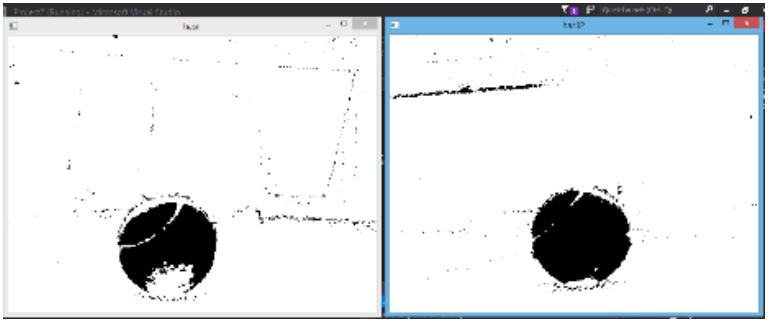
Gambar biner yang belum dilakukan proses *threshold* dapat dilihat pada gambar 4.10 (a). Kemudian, perubahan gambar biner dimana bentuk objek telah terlihat berwarna hitam dan latar belakangnya berwana putih setelah diubah parameter HSV-nya terlihat pada gambar (b). Proses selanjutnya adalah *noise filtering* dengan melakukan proses *erode* seperti pada gambar (c) dan *dilate* seperti pada gambar (d).



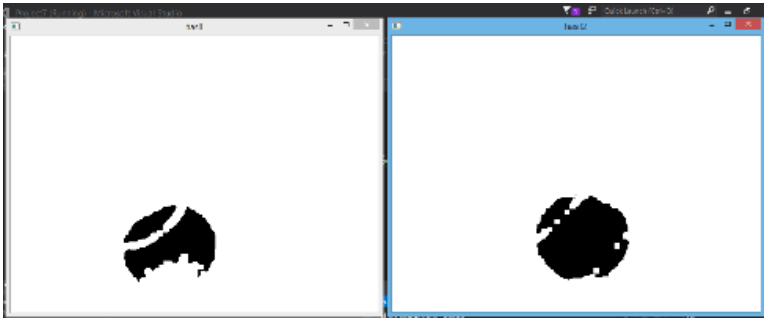
Gambar 4.9 *Trackbar* yang digunakan untuk mencari perubahan gambar HSV menjadi gambar *threshold*



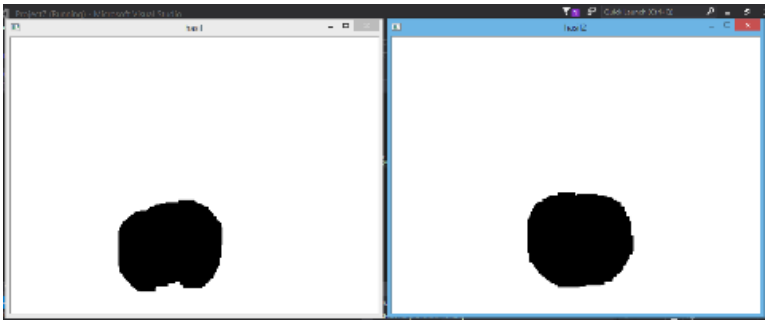
(a)



(b)



(c)



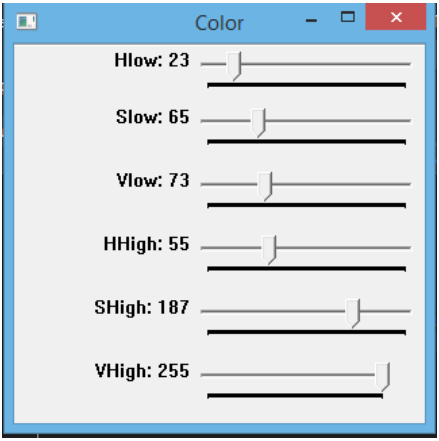
(d)

Gambar 4.10 Proses *threshold* dan *noise filtering*

Nilai *range* HSV yang diperoleh untuk mendeteksi objek dapat dilihat pada gambar 4.11. Nilai *range* HSV tersebut adalah sebagai berikut:

- H low : 23
- S low : 65
- V low : 73
- H high : 55
- S high : 187
- V high : 255

Nilai *range* HSV yang diperoleh kemudian dimanfaatkan untuk proses pendeteksian objek. Penentuan *range* HSV perlu dilakukan dengan beberapa percobaan untuk mendapatkan hasil yang terbaik karena gambar hasil proses *threshold* sangat mempengaruhi proses pendeteksian dari objek. Kondisi lingkungan seperti intensitas cahaya saat dilakukan proses pendeteksian juga sangat berpengaruh, sehingga nilai *range* HSV yang diperoleh juga dapat berubah ketika intensitas cahaya pada lingkungan tempat pengujiannya berubah.



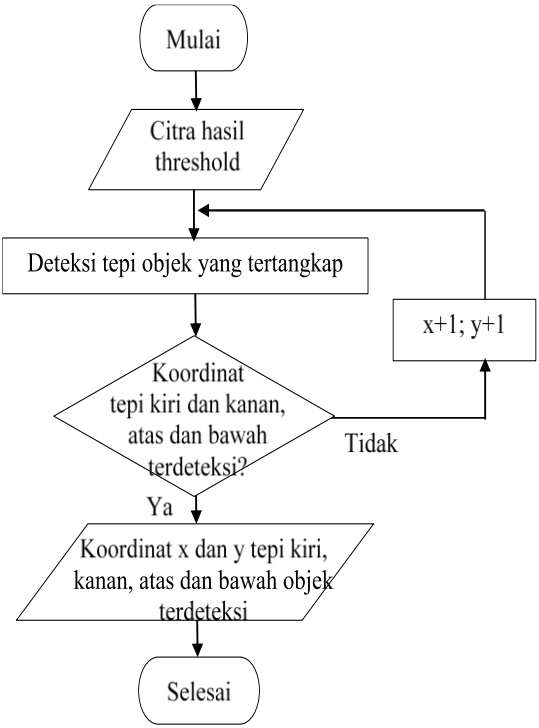
Gambar 4.11 Informasi hasil pencarian nilai HSV dengan *trackbar*

4.2.2 Pencarian Variabel Pendeteksian Objek

Kode Program 5 : Mendeteksi Objek

```
for(int x=0;x<thres->width;x=x++)
for (int y=0;y<thres->height;y=y++)
        {
        if(hasil->imageData[hasil->widthStep*y+x*hasil-
>nChannels]==0)
                {
                ukur.x= x;
                }
        }
for (int y=0;y<thres->height;y=y++)
for(int x=0;x<thres->width;x=x++)
        {
        if(hasil->imageData[hasil->widthStep*y+x*hasil-
>nChannels]==0)
                {
                ukur.y= y;
                }
```

Pendeteksian objek yang ditentukan berdasarkan warna objeknya dapat dilakukan dengan memanfaatkan bentuk gambar biner yang sebelumnya telah dilakukan proses *threshold*. Kemudian dengan menggunakan program, koordinat tepi objek dapat diperoleh dari gambar hasil *threshold*. Koordinat x dan y tepi kiri dan tepi kanan objek yang diperoleh dari program akan digunakan untuk menghitung koordinat titik tengah dari objek tersebut. Proses dari pencarian koordinat tepi objek dapat dilihat pada Gambar 4.12.

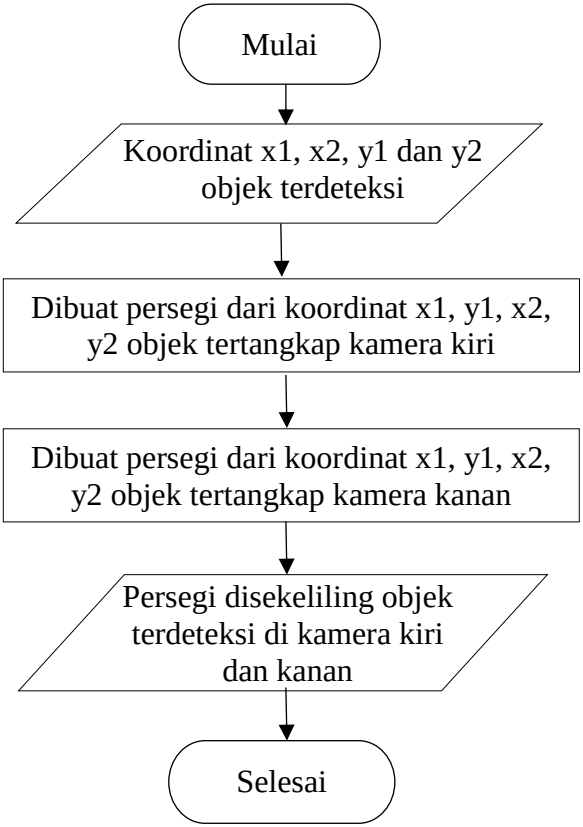


Gambar 4.12 Diagram alir pencarian koordinat tepi objek

Kode program 6 : Menunjukkan objek yang terdeteksi

```
cvRectangle( frame,cvPoint( ukur.x,
ukur.y),cvPoint(  ukur1.x , ukur1.y),cvScalar( 0, 0,
255, 0 ), 2, 0, 0 );
cvRectangle( frame2,cvPoint( ukur2.x,
ukur2.y),cvPoint(  ukur3.x , ukur3.y),cvScalar( 0, 0,
255, 0 ), 2, 0, 0 );
```

Program yang dapat membuat sebuah persegi di sekeliling objek dibuat setelah titik koordinat tepi kiri dan kanan objek diperoleh. Persegi ini bertujuan untuk menunjukkan posisi dari objek yang dideteksi. Persegi ini juga akan deprogram untuk muncul di dalam *frame* gambar hasil tangkapan kamera yang masih memiliki format RGB. Ketika program dijalankan, persegi ini juga akan otomatis menandai dan mengikuti objek yang dideteksi ketika objeknya tertangkap oleh kamera. Langkah-langkah pembuatan persegi ini dapat dilihat melalui diagram pada Gambar 4.13.



Gambar 4.13 Diagram alir pembuatan persegi

4.2.3 Program Penentuan Koordinat x, y, dan z

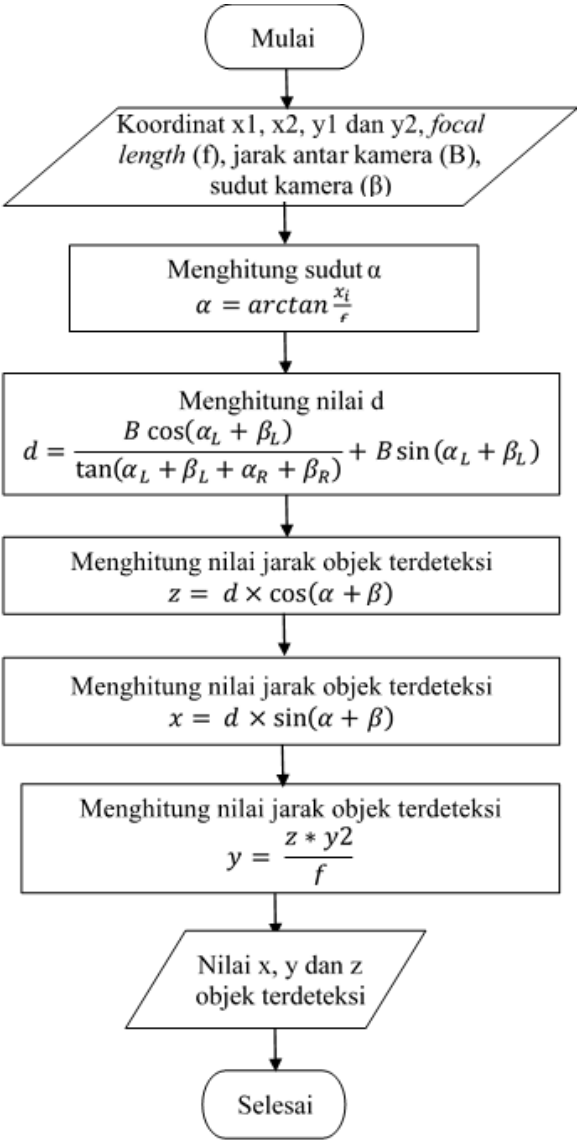
Kode Program 7 : Menentukan koordinat z, x, dan y

```
int f = 6.2215281079953297e+002;
int beta = (((double)15 / (double)180) * PI);
int B = 300;

int xfix = ((ukur.x - ukur1.x) / 2) + ukur1.x;
int xi = (xfix - 320);
int yfix = ((ukur.y - ukur1.y) / 2) + ukur1.y;
int xfix2 = (((ukur2.x - ukur3.x) / 2) + ukur3.x);
int xi2 = (320 - xfix2);
int yfix2 = (((ukur2.y - ukur3.y) / 2) + ukur3.y);

float alphal = (atan((double)xi / (double)f));
float alphar = (atan((double)xi2 / (double)f));
int d = (B * cos((double)alphal + (double)beta)) /
(tan((double)alphal + (double)beta + (double)alphar
+ (double)beta)) + (B * sin((double)alphal +
(double)beta));
int v = d * cos((double)alphal + (double)beta);
int u = d * sin((double)beta + (double)alphal);
int t = (yfix * v) / f;
```

Pada tahap penentuan koordinat, langkah pertama yang dilakukan adalah mencari nilai sudut α dimana sudut α merupakan sudut yang dibentuk antara garis sumbu kamera dengan objek yang dideteksi. Sudut α dapat dicari dengan menggunakan persamaan (4). Kemudian nilai d dicari, dimana d adalah jarak antara lensa kamera dengan objek yang dideteksi. Nilai d dapat diperoleh dari persamaan (5).



Gambar 4.14 Diagram alir penentuan koordinat x, y, dan z objek terhadap kamera

Langkah selanjutnya adalah mencari nilai z atau jarak antara kedua kamera dengan titik tengah objek yang dideteksi. Di dalam program, nilai z dituliskan dalam bentuk notasi v. Persamaan yang digunakan untuk mencari nilai v adalah sebagai berikut:

$$z = d \times \cos(\alpha_L + \beta)$$

Perubahan penggunaan persamaan (6) menjadi persamaan di atas bertujuan untuk mempermudah dalam pengambilan data nilai jarak z.

Setelah mendapatkan nilai v, proses dilanjutkan dengan mencari nilai koordinat x dan y. Di dalam program, nilai x ditulis dalam bentuk notasi u. Persamaan yang digunakan untuk mencari nilai u adalah sebagai berikut:

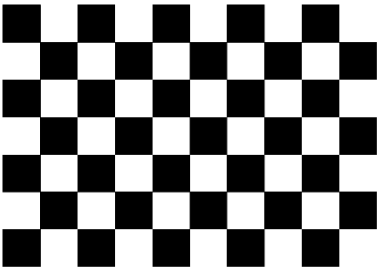
$$x = d \times \sin(\alpha_L + \beta)$$

Sama halnya dengan nilai jarak z, perubahan penggunaan persamaan (7) menjadi persamaan di atas bertujuan untuk mempermudah dalam pengambilan data nilai jarak x.

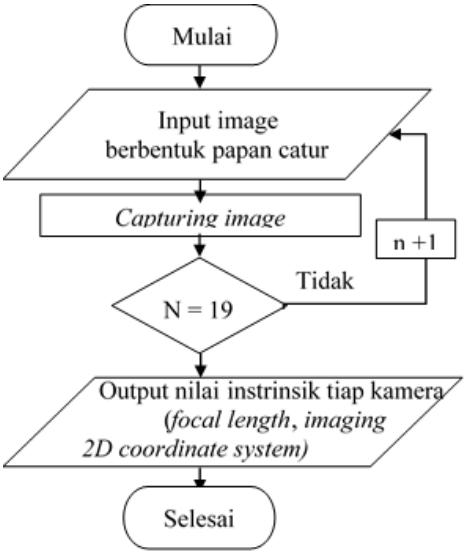
Kemudian koordinat y yang ditulis dalam notasi t dalam program dicari dengan menggunakan persamaan (8). Aliran proses penentuan koordinat objek dapat dilihat pada gambar 4.14.

4.3 Proses Kalibrasi

Proses kalibrasi dilakukan terlebih dahulu sebelum pengujian dilakukan. Program kalibrasi stereo kamera sendiri telah disediakan oleh OpenCV namun perlu diberi penyesuaian dengan data yang digunakan. Data yang digunakan dalam proses kalibrasi adalah gambar kotak hitam dan putih seperti papan catur dengan ukuran 10 kotak x 7 kotak. Ukuran tiap kotaknya adalah 2,5cm x 2,5cm. Gambar papan catur yang digunakan dapat dilihat pada gambar 4.15, sedangkan untuk diagram alir dari proses kalibrasi dapat dilihat pada gambar 4.16.



Gambar 4.15 Papan catur yang digunakan untuk proses kalibrasi

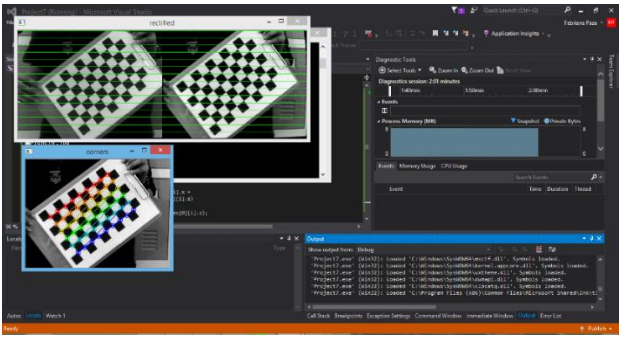


Gambar 4.16 Diagram Alir proses kalibrasi *stereo vision*

Proses pengambilan data kalibrasi dapat dilihat pada gambar 4.17. Hasil dari proses kalibrasi adalah diperolehnya nilai *focal length* dan sistem koordinat 2D dari kedua kamera. Hasil kalibrasi dipeoleh dalam bentuk matriks seperti berikut:

$$M1 = \begin{bmatrix} 6.2215281079953297e+002 & 0 & 1.4766261256446211e+002 \\ 0 & 6.2215281079953297e+002 & 1.3253156514508069e+002 \\ 0 & 0 & 1 \end{bmatrix}$$

$$M2 = \begin{bmatrix} 6.2215281079953297e+002 & 0 & 1.3952657644359141e+002 \\ 0 & 6.2215281079953297e+002 & 1.3252808044018661e+002 \\ 0 & 0 & 1 \end{bmatrix}$$



Gambar 4.17 Proses kalibrasi *stereo vision*

Nilai dari *f*, u_0, dan v_0 yang diperoleh dari proses kalibrasi dapat diketahui berdasarkan bentuk matriks M sebagai berikut:

$$M = \begin{matrix} f & 0 & u_0 \\ 0 & f & v_0 \\ 0 & 0 & 1 \end{matrix}$$

Berdasarkan bentuk matriks M tersebut dapat diketahui bahwa nilai *f* dari kedua kamera adalah 622,15281. Nilai *f* tersebut nantinya akan digunakan dalam proses pengujian. Nilai f akan digunakan dalam persamaan (4) dan (8), dimana kedua persamaan tersebut akan mempengaruhi penentuan nilai koordinat 3 dimensi dari objek.

(Halaman ini sengaja dikosongkan)

BAB 5
PENDETEKSIAN OBJEK DAN PENENTUAN PERSAMAAN KOMPENSASI

5.1 Pendeteksian Objek

Pengujian deteksi objek tuggal yang berupa bola tenis dilakukan dengan menjadikan objek bergerak seperti bandul. Proses pengujian deteksi bola tenis dilakukan pada siang hari. Pertama, bola tenis diikatkan pada seutas benang dengan panjang 30cm, kemudian digantung dengan jarak 70cm dari kamera. Jarak 70cm diambil sebagai jarak pengujian karena pada jarak inilah gambar yang tertangkap oleh kamera paling lebar. Langkah selanjutnya adalah mengayunkan bola tenis dengan sudut awal 25°. Tepat saat bola tenis diayunkan, proses perekaman tiap *frame*-nya dijalankan. Proses ini dilakukan hingga diperoleh jumlah *frame* sebanyak 1000. Kemudian dari 1000 *frame* tersebut dicari dan dihitung jumlah frame dimana bola tenis tidak terdeteksi dengan sempurna. Salah satu *frame* hasil pendeteksian dapat dilihat pada gambar 5.1.

Gambar 5.1 Hasil pendeteksian objek bola tenis

Berdasarkan data yang diperoleh, ada beberapa *frame* dimana terjadi kesalahan pendeteksian pada objek bola tenis. Kesalahan pendeteksian yang ditemukan adalah pendeteksian yang melebihi objek bola tenis dan pendeteksian dimana bola tenis tidak

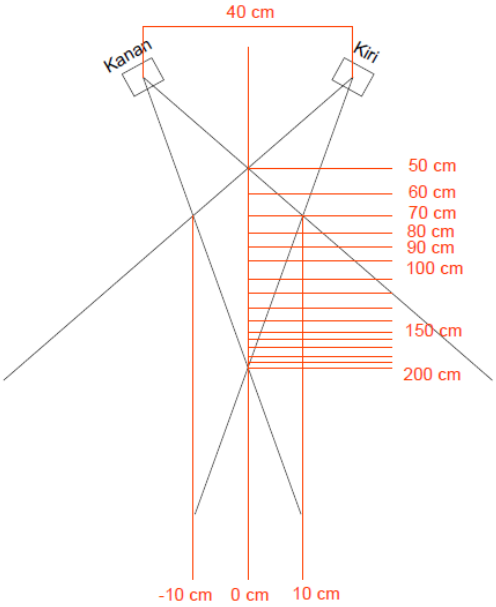
berada dalam kotak pendeteksi secara penuh. Salah satu kesalahan pendeteksian dapat dilihat pada gambar 5.2. Kesalahan pendeteksian yang terjadi dapat disebabkan oleh hasil *threshold* yang kurang sempurna sehingga dapat mendeteksi objek lain. Penyebab lainnya adalah perbedaan intensitas cahaya yang diterima oleh kedua kamera. Kesalahan-kesalahan pendeteksian yang terjadi dipengaruhi oleh kualitas 2 kamera yang berbeda.

Gambar 5.2 Hasil kesalahan pendeteksian objek bola tenis

Berdasarkan data yang diperoleh, ada 54 *frame* yang memiliki kesalahan pendeteksian dari 1000 *frame*. Dari data yang diperoleh, tingkat keberhasilan pendeteksian objek dapat dikalkulasikan dengan menggunakan program yang telah dibangun sebesar 96,4%.

5.2 Pendeteksian Jarak (z)

Pengujian program penentu jarak objek terhadap kamera (z) dilakukan dengan meletakkan objek pada jarak tertentu dimana jarak tangkap minimum dan maksimum kamera telah diketahui sebelumnya. Pengambilan data dimulai pada jarak terdekat 60cm hingga jarak terjauh 190cm dengan pertambahan jarak 10cm. Pada saat pengambilan data, objek tidak bergerak tetapi hanya berubah posisi. Skema pengambilan data dapat dilihat pada gambar 5.3.

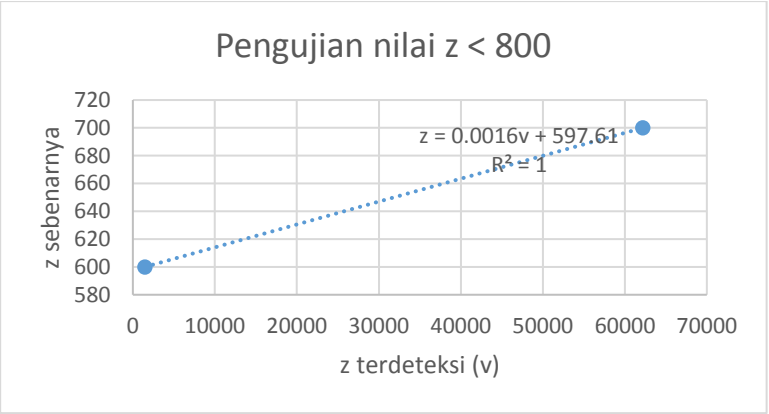


Gambar 5.3 Skema pengambilan data z dan x

Pengambilan data z diambil pada jarak 60, 70, 80, 90, 100, 110, 120, 130, 140, 150, 160, 170, 180, 190 cm. Kemudian untuk pengambilan data koordinat x diambil pada posisi -10, -5, 0, 5, 10 cm untuk setiap data jarak z. Selanjutnya untuk data koordinat y diambil pada posisi -3, 2, 7, 12, 17, 22 cm untuk setiap data jarak z, dimana titik 0-nya berada pada pusat kamera. Data-data hasil pengujian koordinat x, y, dan z dapat dilihat pada lampiran.

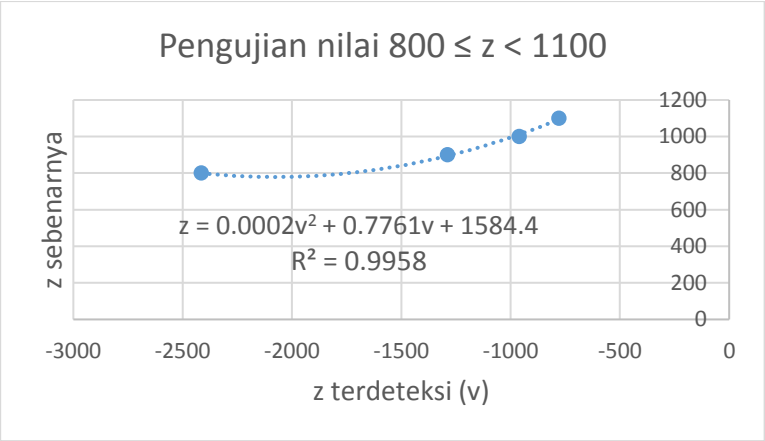
Pada tabel 1 yang tertera dalam lampiran, nilai z yang diperoleh dari program berbeda dari nilai z yang sebenarnya. Adanya perbedaan nilai ini dapat disebabkan oleh perbedaan antara perubahan nilai z yang sebenarnya dengan yang tertangkap oleh kamera. Suatu persamaan garis yang dapat mengurangi kesalahan pendeteksian nilai z dibutuhkan untuk mengatasi hal ini. Aplikasi

Microsoft Excel digunakan untuk menemukan persamaan kompensasi tersebut. Persamaan kompensasi diperoleh dari perbandingan nilai z yang diperoleh dari program (v) dengan nilai z yang sebenarnya. Ada beberapa persamaan kompensasi yang dibuat berdasarkan *range* jarak v yang diperoleh dalam pengujian. *Range* pertama adalah v > 0. Pada *range* ini, persamaan kompensasi sebesar z = (0,0016*v) + 597,61 diperoleh. Dari persamaan tersebut, nilai R^2 yang bernilai 1 diperoleh. R^2 adalah persentase kesesuaian data dengan model persamaan. Grafik persamaan ini dapat dilihat pada gambar 5.4.



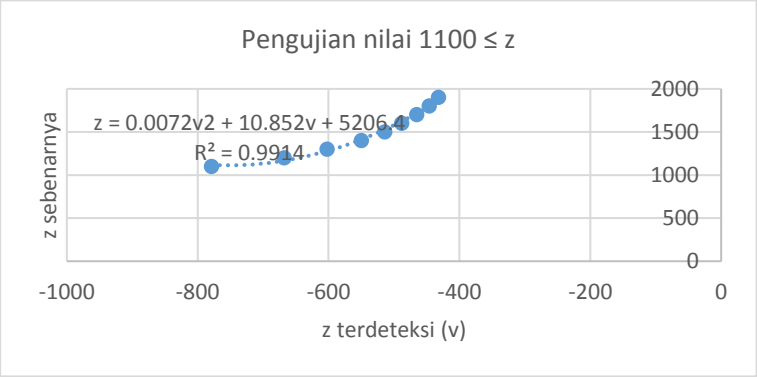
Gambar 5.4 Grafik persamaan kompensasi nilai z untuk z < 800

Range kedua yang digunakan dalam pengujian adalah 800 ≤ z < 1100. Persamaan kompensasi yang diperoleh pada *range* ini adalah z = 0,0002v^2 + 0,7761v + 1584,4. Grafik persamaan kompensasi untuk *range* 800 ≤ z < 1100 dapat dilihat pada gambar 5.5. Pada *range* ini nilai R^2-nya adalah 0,9958 atau dapat dikatakan bahwa tingkat kesesuaian data dengan model persamaan adalah 99,58%.



Gambar 5.5 Grafik persamaan kompensasi nilai z untuk 800 ≤ z < 1100

Range yang terakhir untuk pengujian nilai z adalah 1100 ≤ z. Dalam *range* ini, persamaan kompensasi yang diperoleh adalah $z = 0{,}0072v^2 + 10{,}852v + 5206{,}4$. Grafik persamaan kompensasi ini dapat dilihat pada gambar 5.6. Pada *range* ini nilai R^2-nya adalah 0,9914 atau dapat dikatakan bahwa tingkat kesesuaian data dengan model persamaan adalah 99,14%.



Gambar 5.6 Grafik persamaan kompensasi nilai z untuk 1100 ≤ z

Persamaan kompensasi yang diperoleh dari perangkat lunak Microsoft Excel kemudian diterapkan ke dalam program untuk mengkalkulasi ulang nilai z. Nilai z yang dideteksi yang baru diperoleh setelah dilakukan pengujian program ulang dengan menggunakan persamaan kompensasi.

5.3 Pendeteksian Koordinat x

Tabel 3 pada lampiran adalah data nilai x yang diperoleh dari pengujian program. Pada tabel hasil pengujian nilai koordinat x, nilai koordinat x yang dideteksi (u) juga berbeda jauh dengan nilai yang sebenarnya. Perbedaan ini dapat dikurangi dengan menggunakan persamaan kompensasi seperti pada pengujian jarak z. Persamaan kompensasi dapat diperoleh dengan menggunakan perangkat lunak Microsoft Excel. Persamaan kompensasi diperoleh dari perbandingan nilai x yang dideteksi (u) dengan nilai x yang sebenarnya. Persamaan kompensasi dibuat berdasarkan *range* z yang digunakan dalam pengujian.

Range pertama yang digunakan dalam pengujian nilai koordinat x adalah $z \leq 600$. Persamaan kompensasi yang diperoleh pada *range* ini adalah $x = 0{,}0000001*u^2 + 0{,}0022*u - 0.8258$. Grafik persamaan kompensasi ini dapat dilihat pada gambar 5.7. Pada *range* ini, nilai R^2-nya adalah 1 atau dapat dikatakan bahwa tingkat kesesuaian data dengan model persamaan adalah 100%.

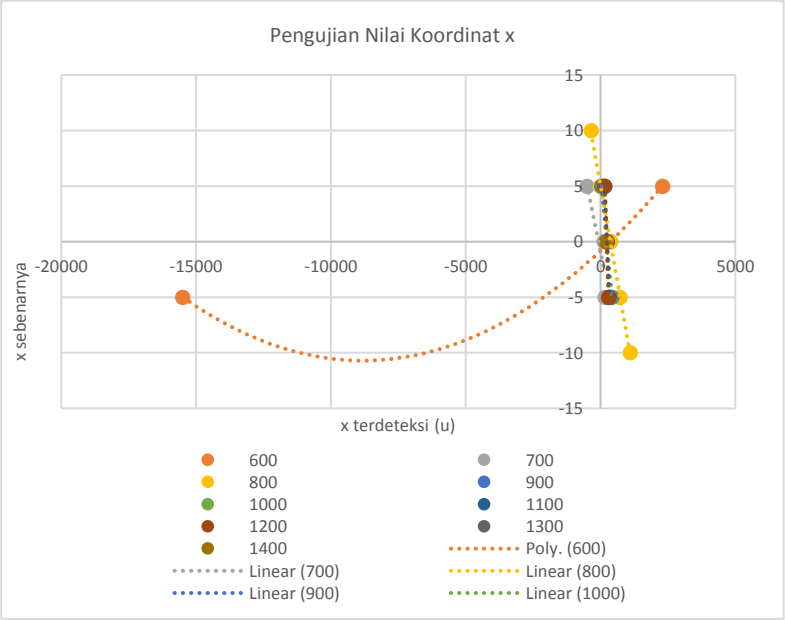
Range kedua yang digunakan dalam pengujian nilai koordinat x adalah $600 < z \leq 700$. Persamaan kompensasi yang diperoleh pada *range* ini adalah $x = (-0{,}0119*u) – 0{,}8577$. Grafik persamaan kompensasi ini dapat dilihat pada gambar 5.7. Pada *range* ini nilai R^2-nya adalah 0,7956 atau dapat dikatakan bahwa tingkat kesesuaian data dengan model persamaan adalah 79,56%.

Range ketiga yang digunakan dalam pengujian nilai koordinat x adalah $700 < z \leq 800$. Persamaan kompensasi yang diperoleh pada *range* ini adalah $x = (-0{,}0659*u) + 7{,}0291$. Grafik persamaan kompensasi ini dapat dilihat pada gambar 5.7. Pada

range ini nilai R^2-nya adalah 0,9997 atau dapat dikatakan bahwa tingkat kesesuaian data dengan model persamaan adalah 99,97%.

Range keempat yang digunakan dalam pengujian nilai koordinat x adalah 800 < z ≤ 900. Persamaan kompensasi yang diperoleh pada *range* ini adalah x = (-0,0587*u) + 8,6327. Grafik persamaan kompensasi ini dapat dilihat pada gambar 5.7. Pada *range* ini nilai R^2-nya adalah 1 atau dapat dikatakan bahwa tingkat kesesuaian data dengan model persamaan adalah 100%.

Range terakhir yang digunakan dalam pengujian nilai koordinat x adalah 900 < z. Persamaan kompensasi yang diperoleh pada *range* ini adalah x = (-0,0806*u) + 11,716. Grafik persamaan kompensasi ini dapat dilihat pada gambar 5.7. Pada *range* ini nilai R^2-nya adalah 0,9999 atau dapat dikatakan bahwa tingkat kesesuaian data dengan model persamaan adalah 99,99%.



Gambar 5.7 Grafik Persamaan Kompensasi Hasil Pengujian Nilai Koordinat x

Persamaan kompensasi yang diperoleh dari perangkat lunak Microsoft Excel kemudian diterapkan ke dalam program untuk mengkalkulasi ulang nilai x. Nilai x yang dideteksi yang baru diperoleh setelah dilakukan pengujian program ulang dengan menggunakan persamaan kompensasi.

5.4 Pendeteksian Koordinat y

Tabel 5 pada lampiran adalah data nilai y yang diperoleh dari pengujian program. Pada tabel hasil pengujian nilai koordinat y, nilai koordinat y yang dideteksi (t) juga berbeda jauh dengan nilai yang sebenarnya. Perbedaan ini dapat dikurangi dengan menggunakan persamaan kompensasi seperti pada pengujian jarak z dan x. Persamaan kompensasi dapat diperoleh dengan menggunakan perangkat lunak Microsoft Excel. Persamaan kompensasi diperoleh dari perbandingan nilai y yang dideteksi (t) dengan nilai y yang sebenarnya. Persamaan kompensasi dibuat berdasarkan *range* z yang digunakan dalam pengujian.

Range pertama yang digunakan dalam pengujian nilai koordinat y adalah $z \leq 600$. Persamaan kompensasi yang diperoleh pada *range* ini adalah y = -0,037t + 9,1113. Grafik persamaan kompensasi ini dapat dilihat pada gambar 5.8. Pada *range* ini nilai R^2-nya adalah 0,9815 atau dapat dikatakan bahwa tingkat kesesuaian data dengan model persamaan adalah 98,15%.

Range kedua yang digunakan dalam pengujian nilai koordinat y adalah $600 < z \leq 700$. Persamaan kompensasi yang diperoleh pada *range* ini adalah y = -0,0397t + 10,465. Grafik persamaan kompensasi ini dapat dilihat pada gambar 5.8. Pada *range* ini nilai R^2-nya adalah 0,9999 atau dapat dikatakan bahwa tingkat kesesuaian data dengan model persamaan adalah 99,99%.

Range ketiga yang digunakan dalam pengujian nilai koordinat y adalah $700 < z \leq 800$. Persamaan kompensasi yang diperoleh pada *range* ini adalah y = -0,0409t + 12,672. Grafik persamaan kompensasi ini dapat dilihat pada gambar 5.8. Pada

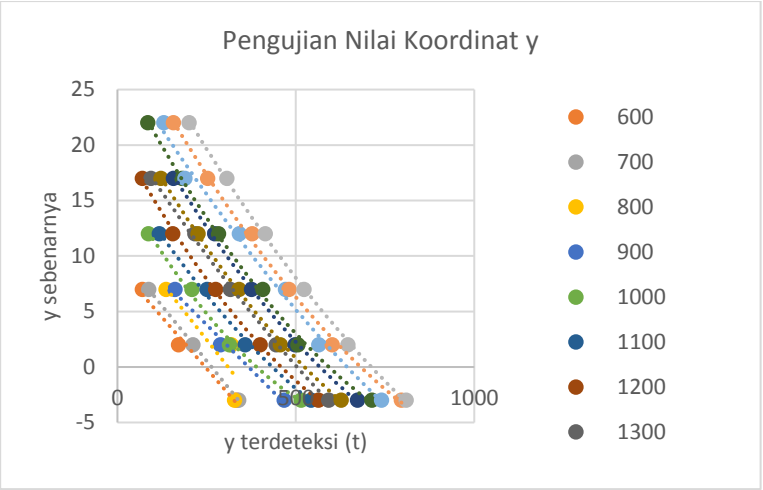
range ini nilai R^2-nya adalah 0,7851 atau dapat dikatakan bahwa tingkat kesesuaian data dengan model persamaan adalah 78,51%.

Range keempat yang digunakan dalam pengujian nilai koordinat y adalah $800 < z \leq 900$. Persamaan kompensasi yang diperoleh pada *range* ini adalah $y = -0{,}0324t + 11{,}901$. Grafik persamaan kompensasi ini dapat dilihat pada gambar 5.8. Pada *range* ini nilai R^2-nya adalah 0,9912 atau dapat dikatakan bahwa tingkat kesesuaian data dengan model persamaan adalah 99,12%.

Range kelima yang digunakan dalam pengujian nilai koordinat y adalah $900 < z \leq 1000$. Persamaan kompensasi yang diperoleh pada *range* ini adalah $y = 0{,}00004t^2 – 0{,}0575t + 16{,}898$. Grafik persamaan kompensasi ini dapat dilihat pada gambar 5.8. Pada *range* ini nilai R^2-nya adalah 0,9960 atau dapat dikatakan bahwa tingkat kesesuaian data dengan model persamaan adalah 99,60%.

Range keenam yang digunakan dalam pengujian nilai koordinat y adalah $1000 < z \leq 1100$. Persamaan kompensasi yang diperoleh pada *range* ini adalah $y = 0{,}00002t^2 – 0{,}0513t + 17{,}851$. Grafik persamaan kompensasi ini dapat dilihat pada gambar 5.8. Pada *range* ini nilai R^2-nya adalah 0,9954 atau dapat dikatakan bahwa tingkat kesesuaian data dengan model persamaan adalah 99,54%.

Range ketujuh yang digunakan dalam pengujian nilai koordinat y adalah $1100 < z \leq 1200$. Persamaan kompensasi yang diperoleh pada *range* ini adalah $y = 0{,}00003t^2 – 0{,}0591t + 20{,}806$. Grafik persamaan kompensasi ini dapat dilihat pada gambar 5.8. Pada *range* ini nilai R^2-nya adalah 0,9992 atau dapat dikatakan bahwa tingkat kesesuaian data dengan model persamaan adalah 99,92%.



Gambar 5.8 Grafik Persamaan Kompensasi Hasil Pengujian Nilai Koordinat y

Range kedelapan yang digunakan dalam pengujian nilai koordinat y adalah $1200 < z \leq 1300$. Persamaan kompensasi yang diperoleh pada *range* ini adalah $y = 0{,}00001t^2 - 0{,}0505t + 21{,}812$. Grafik persamaan kompensasi ini dapat dilihat pada gambar 5.8. Pada *range* ini nilai R^2-nya adalah 0,9986 atau dapat dikatakan bahwa tingkat kesesuaian data dengan model persamaan adalah 99,86%.

Range kesembilan yang digunakan dalam pengujian nilai koordinat y adalah $1300 < z \leq 1400$. Persamaan kompensasi yang diperoleh pada *range* ini adalah $y = 0{,}00001t^2 - 0{,}0585t + 23{,}859$. Grafik persamaan kompensasi ini dapat dilihat pada gambar 5.8. Pada *range* ini nilai R^2-nya adalah 0,9994 atau dapat dikatakan bahwa tingkat kesesuaian data dengan model persamaan adalah 99,94%.

Range kesepuluh yang digunakan dalam pengujian nilai koordinat y adalah $1400 < z \leq 1500$. Persamaan kompensasi yang diperoleh pada *range* ini adalah $y = 0{,}00002t^2 - 0{,}0595t + 25{,}978$.

Grafik persamaan kompensasi ini dapat dilihat pada gambar 5.8. Pada *range* ini nilai R^2-nya adalah 0,9984 atau dapat dikatakan bahwa tingkat kesesuaian data dengan model persamaan adalah 99,84%.

Range kesepuluh yang digunakan dalam pengujian nilai koordinat y adalah $1500 < z \leq 1600$. Persamaan kompensasi yang diperoleh pada *range* ini adalah $y = 0{,}00003t^2 – 0{,}0622t + 27{,}262$. Grafik persamaan kompensasi ini dapat dilihat pada gambar 5.8. Pada *range* ini nilai R^2-nya adalah 0,9977 atau dapat dikatakan bahwa tingkat kesesuaian data dengan model persamaan adalah 99,77%.

Range kesebelas yang digunakan dalam pengujian nilai koordinat y adalah $1600 < z \leq 1700$. Persamaan kompensasi yang diperoleh pada *range* ini adalah $y = 0{,}00001t^2 – 0{,}0521t + 27{,}701$. Grafik persamaan kompensasi ini dapat dilihat pada gambar 5.8. Pada *range* ini nilai R^2-nya adalah 0,9910 atau dapat dikatakan bahwa tingkat kesesuaian data dengan model persamaan adalah 99,10%.

Range keduabelas yang digunakan dalam pengujian nilai koordinat y adalah $1700 < z \leq 1800$. Persamaan kompensasi yang diperoleh pada *range* ini adalah $y = 0{,}00002t^2 – 0{,}0605t + 31{,}068$. Grafik persamaan kompensasi ini dapat dilihat pada gambar 5.8. Pada *range* ini nilai R^2-nya adalah 0,9980 atau dapat dikatakan bahwa tingkat kesesuaian data dengan model persamaan adalah 99,80%.

Range terakhir yang digunakan dalam pengujian nilai koordinat y adalah $1800 < z$. Persamaan kompensasi yang diperoleh pada *range* ini adalah $y = 0{,}00002t^2 – 0{,}0602t + 33{,}555$. Grafik persamaan kompensasi ini dapat dilihat pada gambar 5.8. Pada *range* ini nilai R^2-nya adalah 0,9994 atau dapat dikatakan bahwa tingkat kesesuaian data dengan model persamaan adalah 99,94%.

Persamaan kompensasi yang diperoleh dari perangkat lunak Microsoft Excel kemudian diterapkan ke dalam program untuk mengkalkulasi ulang nilai y. Nilai y yang dideteksi yang baru diperoleh setelah dilakukan pengujian program ulang dengan menggunakan persamaan kompensasi.

BAB 6
VERIFIKASI PROGRAM

6.1 Verifikasi Program untuk Jarak z

Data jarak z yang lebih mendekati nilai jarak sebenarnya diperoleh setelah program diberi persamaan kompensasi. Data hasil pengujian program yang telah diberi persamaan kompensasi dapat dilihat pada tabel 6.1.

Tabel 6.1 Hasil Pengujian Program Nilai z dengan Persamaan Kompensasi

z yang sebenarnya (mm)	z yang dideteksi (mm)	z yang dideteksi – z yang sebenarnya (mm)
600	601	1
700	730	30
800	833	33
900	921	21
1000	1006	6
1100	1133	33
1200	1244	44
1300	1336	36
1400	1455	55
1500	1537	37
1600	1644	44
1700	1746	46
1800	1848	48
1900	1913	13

Tabel 6.1 merupakan hasil pengujian program nilai z dengan persamaan kompensasi. Terdapat perbedaan nilai z yang

sebenarnya dengan nilai z yang terdeteksi. Selisih antara nilai yang terdeteksi dengan nilai sebenarnya menandakan masih adanya eror pada program. Selisih terbesar yang terdapat pada hasil pengujian program adalah 55 mm yang terdapat pada jarak 1400mm, sehingga nilai eror terbesar untuk pengujian program jarak z yaitu 3,92%.

6.2 Verifikasi Program untuk Koordinat x

Data jarak x yang lebih mendekati nilai jarak sebenarnya diperoleh etelah program diberi persamaan kompensasi. Data hasil pengujian program yang telah diberi persamaan kompensasi dapat dilihat pada tabel 6.2.

Tabel 6.2 merupakan hasil pengujian program nilai x dengan persamaan kompensasi. Terdapat perbedaan nilai x yang sebenarnya dengan nilai x yang terdeteksi. Selisih antara nilai yang terdeteksi dengan nilai sebenarnya menandakan masih adanya eror pada program. Selisih nilai x terdeteksi dengan nilai x sebenarnya dapat dilihat pada tabel 6.3. Selisih terbesar yang terdapat pada hasil pengujian program adalah 9 cm pada jarak z 1400 mm, sehingga nilai eror terbesar untuk pengujian program jarak x yaitu 0,64%.

Tabel 6.2 Hasil Pengujian Nilai x dengan Persamaan Kompensasi

Z (mm)	X = 10cm	X = 5cm	X = 0cm	X = -5cm	X = -10cm
600		10	7	-1	
700	14	9	1	-5	-13
800		7	1	-5	
900		4	0	-2	
1000		8	2	-3	
1100		1	0	-1	
1200		5	2	0	
1300			-8		
1400			9		
1500			9		
1600			-4		
1700			5		
1800			-4		

Tabel 6.3 Selisih x Terdeteksi dengan x Sebenarnya dengan Persamaan Kompensasi

Z (mm)	X = 10cm	X = 5cm	X = 0cm	X = -5cm	X = -10cm
600		5	7	4	
700	4	4	1	0	-3
800		2	1	0	
900		-1	0	3	
1000		3	2	2	
1100		-4	0	4	
1200		0	2	5	
1300			-8		
1400			9		
1500			9		
1600			-4		
1700			5		
1800			-4		

6.3 Verifikasi Program untuk Koordinat y

Data jarak y yang lebih mendekati nilai jarak sebenarnya diperoleh setelah program diberi persamaan kompensasi. Data hasil pengujian program yang telah diberi persamaan kompensasi dapat dilihat pada tabel 6.4.

Tabel 6.4 merupakan hasil pengujian program nilai y dengan persamaan kompensasi. Terdapat perbedaan nilai y yang sebenarnya dengan nilai y yang terdeteksi. Selisih antara nilai yang terdeteksi dengan nilai sebenarnya menandakan masih adanya eror pada program. Selisih nilai y terdeteksi dengan nilai y sebenarnya dapat dilihat pada tabel 6.5. Selisih terbesar yang terdapat pada hasil pengujian program adalah 35 cm pada jarak z 1400 mm,

sehingga nilai eror terbesar untuk pengujian program jarak y yaitu 2,5%.

Tabel 6.4 Hasil Pengujian Progam Nilai y dengan Persamaan Kompensasi

Z (mm)	y = -3cm	y = 2cm	y = 7cm	y = 12cm	y = 17cm	y = 22cm
600	-4	0	5			
700	-5	1	5			
800	-3	2	5			
900	-5	0	4			
1000	-6	0	6	10		
1100	-4	0	4	9		
1200	-7	0	4	11	17	
1300	-8	-1	4	8	14	
1400	-38	-23	-12	0	10	
1500	-36	-27	-9	-6	0	
1600	-10	-5	0	6	12	17
1700	-7	-2	1	8	12	16
1800	-6	-2	1	6	12	16
1900	-6	-2	0	6	12	17

Tabel 6.5 Selisih y Terdeteksi dengan y Sebenarnya dengan Persamaan Kompensasi

Z (mm)	y = -3cm	y = 2cm	y = 7cm	y = 12cm	y = 17cm	y = 22cm
600	-1	-2	-2			
700	-2	-1	-2			
800	0	0	-2			
900	-2	-2	-3			
1000	-3	-2	-1	-2		
1100	-1	-2	-3	-3		
1200	-4	-2	-3	-1	0	
1300	-5	-3	-3	-4	-3	
1400	-35	-25	-19	-12	-7	
1500	-33	-29	-16	-18	-17	
1600	-7	-7	-7	-6	-5	-5
1700	-4	-4	-6	-4	-5	-6
1800	-3	-4	-6	-6	-5	-6
1900	-3	-4	-7	-6	-5	-5

Berdasarkan hasil pengambilan data koordinat x, y, dan z diperoleh nilai error terbesar berada pada jarak 1400mm. Hal ini berhubungan karena untuk mendapatkan nilai x dan y membutuhkan nilai z, sehingga menyebabkan nilai error pada nilai koordinat x dan y muncul pada jarak z 1400mm yang memiliki nilai error terbesar.

BAB 7
KESIMPULAN DAN SARAN

7.1 Kesimpulan

Beberapa kesimpulan yang diperoleh pada penelitian tugas akhir ini adalah sebagai berikut:

1. Proses kalibrasi stereo kamera dilakukan dengan menggunakan papan catur berukuran 7 kotak x 10 kotak dengan ukuran masing-masing kotak 2,5cm x 2,5 cm. Nilai *focal length* yang diperoleh dari proses kalibrasi sebesar 622,15281 untuk kedua kamera.
2. Pendeteksian objek tunggal berupa bola tenis menunjukkan tingkat keberhasilan program sebesar 94,6%. Jarak terdekat objek dengan kamera dimana kedua kamera dapat menangkap gambar objek adalah 600mm, sedangkan jarak terjauh objek yang dapat ditangkap kedua kamera adalah 1900mm.
3. Pendeteksian nilai koordinat z atau jarak objek terhadap kamera yang sudah menggunakan program yang telah diberi persamaan kompensasi menunjukkan nilai error terbesar 3,92% pada jarak 1400mm.
4. Pendeteksian nilai koordinat x atau jarak objek terhadap kamera yang sudah menggunakan program yang telah diberi persamaan kompensasi menunjukkan nilai error terbesar 0,64% pada jarak 1400mm.
5. Pendeteksian nilai koordinat y atau jarak objek terhadap kamera yang sudah menggunakan program yang telah diberi persamaan kompensasi menunjukkan nilai error terbesar 2,5% pada jarak 1400mm.

7.2 Saran

Faktor yang mempengaruhi kesalahan pendeteksian yang diperoleh berdasarkan hasil pengujian program dan pembahasan

pada tugas akhir ini adalah bedanya kualitas dari kedua kamera yang menyebabkan perbedaan intensitas cahaya yang diterima oleh kedua kamera serta kurang sempurnanya persamaan kompensasi yang digunakan karena hanya dilakukan dengan pendekatan, maka dari itu muncul beberapa saran untuk pengembangan selanjutnya sebagai berikut:

- Pendeteksian objek sebaiknya menggunakan 2 kamera yang memiliki kualitas yang sama dan beresolusi tinggi karena akan sangat berpengaruh pada hasil pendeteksian.
- Posisi antar-kamera yang tidak sejajar dapat menyebabkan area tangkap kamera menjadi lebih kecil. Penggunaan kamera dengan *wide lens* akan mempengaruhi area pandang kamera sehingga dapat menambah luasan area tangkap dari kamera.

DAFTAR PUSTAKA

[1] Lutfi, Wardah C. (2015). "Kalibrasi Stereo Kamera dan Penentuan Koordinat 3 Dimensi Untuk Target Tunggal pada Sistem Pelontar Peluru *Autotracking*".

[2] Kuncorojati, A. (2015). "Rancang Bangun Pelontar Peluru Yang Dilengkapi Dengan Kamera Stereo Untuk Pendektesian Target Secara Otomatis".

[3] Imron, M.A,. (2015). "Pendeteksian dan Pelacakan 2D Multi Objek Secara Real – time dengan Menggunakan Kamera Tunggal".

[4] Shukla, Neha, Dr. Anurag Trivedi. (2015). "*Image Based Distance Measurement Technique for Robot Vision using New LBPA approach*".

[5] Tjandrasa, H. (2009). "Stereo Vision Untuk Pengukuran Jarak Objek Dengan Mendeteksi Tepi Subimage"

[6] Putra, D. (2010). "*Pengolahan Citra Digital*". Yogyakarta. ANDI.

[7] Gonzalez, R. C., Woods R. E. (2002), *Digital Image Processing, Second ed*. New Jersey: Prentice-Hall.

[8] Putra, D. (2010). "*Pengolahan Citra Digital*", Yogyakarta, Andi Offset.

[9] R. D. Kusumanto. (2011). "Klasifikasi Warna Menggunakan Pengolahan Warna HSV"

[10] Representasi Model Warna RGB Menggunakan HSL dan HSV. (2011). Dipetik pada 1 November 2015, dari https://mhstekkomp.wordpress.com/2011/05/07/representasi-model-warna-rgb-menggunakan-hsl-dan-hsv/

[11] OpenCV. (2011).Dipetik pada 9 November 2016, dari https://www.willowgarage.com/pages/software/opencv

[12] Sistem Koordinat Kartesius. (2015). Dipetik pada 9 November 2016, dari https://id.m.wikipedia.org/wiki/Sistem_koordinat_Kartesius

[13] Hauck, Alexa, Johanna Ruttinge, Michael Sorg, Georg Farber. (1999). "*Visual Determination of 3D Grasping Points on Unknown Objects with a Binocular Camera System*".

LAMPIRAN

Tabel 1. Hasil Pengujian Nilai z

z yang sebenarnya (mm)	z yang dideteksi (mm)
600	1454
700	62199
800	-2415
900	-1289
1000	-960
1100	-779
1200	-668
1300	-602
1400	-550
1500	-514
1600	-488
1700	-465
1800	-446
1900	-432

Tabel 2 Hasil Pengujian Nilai x

Z (mm)	X = 10cm	X = 5cm	X = 0cm	X = -5cm	X = -10cm
600		2299	361	-15499	
700		-506	124	165	
800	-348	-11	362	726	1096
900		40	230	418	
1000		121	237	356	
1100		155	244	319	
1200		156	226	296	
1300			224		
1400			225		
1500			219		
1600			218		
1700			218		
1800			218		
1900			218		

Tabel 3 Hasil Pengujian Nilai y (cm)

Z (mm)	y = -3cm	y = 2cm	y = 7cm	y = 12cm	y = 17cm	y = 22cm
600	335	171	70			
700	340	212	88			
800	328	319	136			
900	467	289	161			
1000	515	314	209	87		
1100	543	358	251	117		
1200	563	400	275	155	70	
1300	592	445	317	217	94	
1400	627	457	341	226	122	
1500	673	498	376	273	157	
1600	713	506	407	283	181	85
1700	740	564	471	342	189	130
1800	796	602	481	378	253	157
1900	809	646	523	414	307	201

Konstruksi Program

```
#include "cv.h"
#include "highgui.h"
#include "math.h"
#include <cv.h>
#include <highgui.h>
#include <math.h>
#include <cxtypes.h>
#include <iomanip>
#include <stdio.h>
#include <conio.h>
#include "tserial.h"
#include <string.h>

#define PI 3.14159265

const int QUALITY = 100;
char dataserial[8];
char fileName[13];
int counterFile = 0;
CvPoint mousep[7];
CvPoint statpos[17];
CvPoint MouseFilter(CvPoint mop);
CvPoint ukur, ukur1, ukur2, ukur3, tengah =
cvPoint(0, 0);
int sta_test(CvPoint pos);

IplConvKernel *element;
IplConvKernel *element2;
void GetDesktopResolution(int& horizontal, int&
vertical);
CvPoint pt = cvPoint(0, 0);
int add_pt = 0;
```

```
void on_mouse(int event, int x, int y, int
flags, void* param)
{
	if (event == CV_EVENT_MOUSEMOVE)
	{
		pt = cvPoint(x, y);
		add_pt = 1;

	}
}

int main()
{

	int x = 0, y = 0;
	int xs = 0, ys = 0, z = 0;
	double param, teta1, teta2;
	param = 1.0;
	int hl = 23, sl = 65, vl = 73, hh = 50, sh
= 187, vh = 255;
	CvCapture* capture = 0;;
	IplImage *frame;
	int data = 0, H = 0, S = 0, V = 0;
	capture = cvCaptureFromCAM(1);
	cvNamedWindow("left result", 1);
	int ero1 = 1, ero2 = 1, dil1 = 2, dil2 =
1;
	int hlr = 23, slr = 65, vlr = 73, hhr =
50, shr = 187, vhr = 255;
	CvCapture* capture2 = 0;;
	IplImage *frame2;
	int data2 = 0, Hr = 0, Sr = 0, Vr = 0;
	capture2 = cvCaptureFromCAM(2);
	cvNamedWindow("right result", 1);
```

```
	int ero1r = 1, ero2r = 1, dil1r = 2, dil2r
= 1;
	int k, j;


	//=======================================
=
	cvNamedWindow("Color", 1);
	cvCreateTrackbar("Hlow", "Color", &hl,
179, 0);
	cvCreateTrackbar("Slow", "Color", &sl,
255, 0);
	cvCreateTrackbar("Vlow", "Color", &vl,
255, 0);
	cvCreateTrackbar("HHigh", "Color", &hh,
179, 0);
	cvCreateTrackbar("SHigh", "Color", &sh,
255, 0);
	cvCreateTrackbar("VHigh", "Color", &vh,
255, 0);
	//=======================================
==
	cvNamedWindow("Filter", 1);
	cvCreateTrackbar("erode 1", "Filter",
&ero1, 20, 0);
	cvCreateTrackbar("dilate 1", "Filter",
&dil1, 20, 0);
	cvCreateTrackbar("erode 2", "Filter",
&ero2, 20, 0);
	cvCreateTrackbar("dilate 2", "Filter",
&dil2, 20, 0);
	//=======================================
==
	while (1)
	{
```

```
            frame = cvRetrieveFrame(capture);
            frame2 = cvRetrieveFrame(capture2);
            cvMirror(frame, frame, 90);
            cvMirror(frame2, frame2, 90);
            IplImage *thres =
cvCreateImage(cvSize(frame->width, frame-
>height), 8, 1);
            IplImage *thres2 =
cvCreateImage(cvSize(frame2->width, frame2-
>height), 8, 1);
            IplImage *HSV =
cvCreateImage(cvSize(frame->width, frame-
>height), 8, 3);
            IplImage *HSV2 =
cvCreateImage(cvSize(frame2->width, frame2-
>height), 8, 3);
            IplImage *hasil =
cvCreateImage(cvSize(frame->width, frame-
>height), 8, 1);
            IplImage *hasil2 =
cvCreateImage(cvSize(frame2->width, frame2-
>height), 8, 1);
            cvCvtColor(frame, HSV, CV_BGR2HSV);
            cvCvtColor(frame2, HSV2,
CV_BGR2HSV);
            cvInRangeS(HSV, cvScalar(hl, sl,
vl), cvScalar(hh, sh, vh), thres);
            cvInRangeS(HSV2, cvScalar(hl, sl,
vl), cvScalar(hh, sh, vh), thres2);
            element =
cvCreateStructuringElementEx(9, 9, 4, 4,
CV_SHAPE_RECT, NULL);
            cvErode(thres, thres, element,
ero1);
```

```
            cvDilate(thres, thres, element,
dil1);
            cvErode(thres, thres, element,
ero2);
            cvDilate(thres, thres, element,
dil2);
            cvErode(thres2, thres2, element,
ero1);
            cvDilate(thres2, thres2, element,
dil1);
            cvErode(thres2, thres2, element,
ero2);
            cvDilate(thres2, thres2, element,
dil2);
            //==================PROSES
THRESHOLDING DAN
PENENTUAN=======================================
============
            for (int x = 0; x<hasil->width; x =
x++)
                  for (int y = 0; y<hasil-
>height; y = y++)
                  {
                        if (thres-
>imageData[thres->widthStep*y + x*thres-
>nChannels] == 0)
                              hasil-
>imageData[hasil->widthStep*y + x*hasil-
>nChannels] = 255;
                        else
                              hasil-
>imageData[hasil->widthStep*y + x*hasil-
>nChannels] = 0;
                  }
```

```
            for (int x1 = 0; x1<hasil2->width;
x1 = x1++)
                  for (int y1 = 0; y1<hasil2-
>height; y1 = y1++)
                  {
                        if (thres2-
>imageData[thres2->widthStep*y1 + x1*thres2-
>nChannels] == 0)
                              hasil2-
>imageData[hasil2->widthStep*y1 + x1*hasil2-
>nChannels] = 255;
                        else
                              hasil2-
>imageData[hasil2->widthStep*y1 + x1*hasil2-
>nChannels] = 0;
                  }
            //==========MENDETEKSI
OBJEK=========================================
====//
            for (int x = 0; x<thres->width; x =
x++)
                  for (int y = 0; y<thres-
>height; y = y++)
                  {
                        if (hasil-
>imageData[hasil->widthStep*y + x*hasil-
>nChannels] == 0)
                        {
                              ukur.x = x;
                        }
                  }

            for (int x = 0; x<thres2->width; x =
x++)
```

```
                  for (int y = 0; y<thres2-
>height; y = y++)
                  {
                        if (hasil2-
>imageData[hasil2->widthStep*y + x*hasil2-
>nChannels] == 0)
                        {
                              ukur2.x = x;
                        }
                  }

            for (int y = 0; y<thres->height; y =
y++)
                  for (int x = 0; x<thres-
>width; x = x++)
                  {
                        if (hasil-
>imageData[hasil->widthStep*y + x*hasil-
>nChannels] == 0)
                        {
                              ukur.y = y;
                        }
                  }

            for (int y = 0; y<thres2->height; y
= y++)
                  for (int x = 0; x<thres2-
>width; x = x++)
                  {
                        if (hasil2-
>imageData[hasil2->widthStep*y + x*hasil2-
>nChannels] == 0)
                        {
                              ukur2.y = y;
                        }
```

```
                }

            for (int x = thres->width - 1; 0<x;
x = x--)
                for (int y = thres->height -
1; 0<y; y = y--)
                {
                      if (hasil-
>imageData[hasil->widthStep*y + x*hasil-
>nChannels] == 0)
                      {
                            ukur1.x = x;
                      }
                }

            for (int x = thres2->width - 1; 0<x;
x = x--)
                for (int y = thres2->height -
1; 0<y; y = y--)
                {
                      if (hasil2-
>imageData[hasil2->widthStep*y + x*hasil2-
>nChannels] == 0)
                      {
                            ukur3.x = x;
                      }
                }

            for (int y = thres->height - 1; 0<y;
y = y--)
                for (int x = thres->width - 1;
0<x; x = x--)
                {
```

```
                        if (hasil-
>imageData[hasil->widthStep*y + x*hasil-
>nChannels] == 0)
                        {
                              ukur1.y = y;
                        }
                  }

            for (int y = thres2->height - 1;
0<y; y = y--)
                  for (int x = thres2->width -
1; 0<x; x = x--)
                  {
                        if (hasil2-
>imageData[hasil2->widthStep*y + x*hasil2-
>nChannels] == 0)
                        {
                              ukur3.y = y;
                        }
                  }
            //====================ngotaki
disekeliling objek======================
            cvRectangle(frame, cvPoint(ukur.x,
ukur.y), cvPoint(ukur1.x, ukur1.y), cvScalar(0,
0, 255, 0), 2, 0, 0);
            cvRectangle(frame2, cvPoint(ukur2.x,
ukur2.y), cvPoint(ukur3.x, ukur3.y), cvScalar(0,
0, 255, 0), 2, 0, 0);

            sprintf(fileName, "gmb%d.jpeg",
counterFile);
            counterFile++;
            if (counterFile < 1001) {
                  cvSaveImage(fileName, frame,
&QUALITY);
```

```
            }

            sprintf(fileName, "gbr%d.jpeg",
counterFile);
            counterFile++;
            if (counterFile < 1001) {
                  cvSaveImage(fileName, frame2,
&QUALITY);
            }

            //==========MENCARI NILAI X,Y, dan
Z=========================================//
            int f = 6.2215281079953297e+002;
            int beta = (((double)15 /
(double)180) * PI);
            int B = 400;

            int xfix = ((ukur.x - ukur1.x) / 2)
+ ukur1.x;
            int xi = (xfix - 320);
            int yfix = ((ukur.y - ukur1.y) / 2)
+ ukur1.y;
            int xfix2 = (((ukur2.x - ukur3.x) /
2) + ukur3.x);
            int xi2 = (320 - xfix2);
            int yfix2 = (((ukur2.y - ukur3.y) /
2) + ukur3.y);

            float alphal = (atan((double)xi /
(double)f));
            float alphar = (atan((double)xi2 /
(double)f));
            int d = (B * cos((double)alphal +
(double)beta)) / (tan((double)alphal +
```

```
(double)beta + (double)alphar + (double)beta)) +
(B * sin((double)alphal + (double)beta));
            int v = (d * cos((double)alphal));
            int u = d * sin((double)beta +
(double)alphal);
            int t = (yfix * z) / f;

            //correction factor z
            if (int(v >= 0))
            {
                  z = (0.0016*v) + 597.61;
            }

            else if ((v >= -3000) && (v < -800))
            {
                  z = 0.0002*pow(v, 2) +
0.7761*v + 1584.4;
            }

            else if ((v < 0) && (v >= -800))
            {
                  z = 0.0072*pow(v,2) + 10.852*v
+ 5206.4;
            }

            //correction factor x
            if (int((z > 520) && (z <= 620)))
            {
                  x = 0.0000001*pow(u,2) +
0.0022*u - 0.8258;
            }

            else if ((z > 620) && (z <= 720))
            {
                  x = (-0.0119*u) - 0.8577;
```

```
}

else if ((z > 720) && (z <= 820))
{
      x = (-0.0659*u) + 7.0291;
}

else if ((z > 820) && (z <= 920))
{
      x = (-0.0587*u) + 8.6327;
}

else if (z > 920)
{
      x = (-0.0806*u) + 11.716;
}

//correction factor y
if (int((z > 520) && (z <= 620)))
{
      y = -0.037*t + 9.1113;
}

else if ((z > 620) && (z <= 720))
{
      y = -0.0397*t + 10.465;
}

else if ((z > 720) && (z <= 820))
{
      y = -0.0409*t + 12.672;
}

else if ((z > 820) && (z <= 920))
{
```

```
                y = -0.0324*t + 11.901;
            }

            else if ((z > 920) && (z <= 1020))
            {
                y = 0.00004*pow(t,2) -
0.0575*t + 16.898;
            }

            else if ((z > 1020) && (z <= 1120))
            {
                y = 0.00002*pow(t,2) -
0.0513*t + 17.851;
            }

            else if ((z > 1120) && (z <= 1220))
            {
                y = 0.00003*pow(t,2) -
0.0591*t + 20.806;
            }

            else if ((z > 1220) && (z <= 1320))
            {
                y = 0.00001*pow(t,2) -
0.0505*t + 21.812;
            }

            else if ((z > 1320) && (z <= 1420))
            {
                y = 0.00001*pow(t,2) -
0.0585*t + 23.859;
            }

            else if ((z > 1420) && (z <= 1520))
            {
```

```
                  y = 0.00002*pow(t,2) -
0.0595*t + 25.978;
            }

            else if ((z > 1520) && (z <= 1620))
            {
                  y = 0.00003*pow(t,2) -
0.0622*t + 27.262;
            }

            else if ((z > 1620) && (z <= 1720))
            {
                  y = 0.00001*pow(t,2) -
0.0521*t + 27.701;
            }

            else if ((z > 1720) && (z <= 1820))
            {
                  y = 0.00002*pow(t,2) -
0.0605*t + 31.068;
            }

            else if (z <= 1820)
            {
                  y = 0.00002*pow(t,2) -
0.0602*t + 33.555;
            }

            //==========Perubahan dari radian ke
derajat================//
            /*xs = (1 * x) + (0 * y) + (0 * z) +
0;
            xs= k+((-2.111*k)-0.0019);
            ys = (0 * x) + (1 * y) + (0 * z) -
14;
```

```
            ys= j-((-0.8714*j)+12.242);
            teta1 = (atan((double)ys /
(double)z) * (180 / PI)) + 90;
            teta2 = atan((double)xs / (double)z)
* (180 / PI) + 90;*/



      //=====================================
================================//
            system("cls");
            printf("xl : %d , yl : %d \n", xfix,
yfix);
            printf("xr : %d , yr : %d \n",
xfix2, yfix2);
            //printf("x (pixel): %d  \n",((xfix-
xfix2)/2)+xfix2);
            //printf("y (pixel): %d  \n",((yfix-
yfix2)/2)+yfix2);
            printf("x : %d  \n", x);
            printf("t : %d  \n", t);
            printf("u : %d  \n", u);
            printf("y : %d  \n", y);
            printf("v : %d  \n", v);
            printf("z : %d  \n", z);
            //printf("x senjata : %d  \n", xs);
            //printf("y senjata : %d  \n", ys);
            printf("alpha : %f  \n", alphal);
            //printf("teta1 : %f  \n", teta1);
            //printf("teta2 : %f  \n", teta2);

            cvShowImage("hasil", hasil);
            cvShowImage("left result", frame);
            cvShowImage("hasil2", hasil2);
            cvShowImage("right result", frame2);
```

```
            if (cvWaitKey(1) == 'q')
                  break;

            cvReleaseImage(&thres);
            cvReleaseImage(&thres2);
            cvReleaseImage(&HSV);
            cvReleaseImage(&HSV2);
            cvReleaseImage(&hasil);
            cvReleaseImage(&hasil2);
      }
      cvDestroyWindow("Filter");
      cvDestroyWindow("Color");
      cvReleaseImage(&frame);
      cvReleaseImage(&frame2);
      cvReleaseCapture(&capture);
      cvReleaseCapture(&capture2);
      cvDestroyWindow("left result");
      cvDestroyWindow("right result");
      return 0;
}
```

(Halaman ini sengaja dikosongkan)

BIODATA PENULIS

Febriana Pusparaniayu Pasa lahir di Surabaya pada 14 Februari 1994, merupakan anak kedua dari tiga bersaudara dari pasangan Djarot Harso Wahyudi dan Cysca Vera Polii. Penulis telah menempuh pendidikan formal yaitu TK Permai Surabaya (1998-2000), SDKr Petra 5 Surabaya (2000-2001), SDK St. Maria 2 Malang (2001-2006), SMPK St. Maria 2 Malang (2006-2009), SMAK St. Albertus Malang (2009-2012). Penulis melanjutkan pendidikan kuliah dengan mengikuti SNMPTN Jalur Undangan dan diterima di Jurusan Teknik Mesin, Institut Teknologi Sepuluh Nopember. Penulis terdaftar dengan NRP 2112100011.

Penulis mengambil Bidang Studi Manufaktur di Laboratorium Perancangan dan Pengembangan Produk Jurusan Teknik Mesin. Penulis berkegiatan dalam bidang akademik diantaranya menjadi asisten praktikum Pengukuran Teknik dan mata kuliah Gambar Mesin.

Dalam bidang kepanitiaan, penulis aktif menjadi panitia *event* yang diselenggarakan oleh mahasiswa jurusan teknik mesin seperti Mechanical City sie Media Partner (2013), sie Transportasi dan Akomodasi (2014 & 2015) serta panitia *event* Mechanical Competition sie Transportasi dan Akomodasi (2014). Penulis juga aktif mengikuti berbagai macam seminar dan pelatihan untuk meningkatkan *softskill*. Untuk semua informasi dan masukan dapat menghubungi penulis melalui email febrianap.ayupasa@gmail.com.