ROBOT PELACAK MANUSIA MENGGUNAKAN *WEBCAM* SEBAGAI SENSOR VISUAL

TUGAS AKHIR

Nama : Randiansyah Ramadhan Tjais

NIM : 10.41020.0051

Program : S1 (Strata Satu)

Jurusan : Sistem Komputer

SEKOLAH TINGGI

MANAJEMEN INFORMATIKA DAN TEKNIK KOMPUTER

SURABAYA

2014

ROBOT PELACAK MANUSIA MENGGUNAKAN *WEBCAM* SEBAGAI SENSOR VISUAL

TUGAS AKHIR

Diajukan sebagai salah satu syarat untuk menyelesaikan

Program Sarjana Komputer

Oleh:

Nama : Randiansyah Ramadhan Tjais

NIM : 10.41020.0051

Program : S1 (Strata Satu)

Jurusan : Sistem Komputer

SEKOLAH TINGGI

MANAJEMEN INFORMATIKA DAN TEKNIK KOMPUTER

SURABAYA

2014

UNIVERSITAS
Dinamika

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

ABSTRAK

Mobile robot merupakan robot yang bersifat bergerak dari satu titik ke titik lainnya. *Mobile robot* mendeteksi keberadaan objek di sekitar dan mengolahnya dengan *webcam*. Dengan kelebihan yang dimiliki *mobile robot* bisa memudahkan pekerjaan manusia. Salah satu perkerjaan manusia yang bisa dibantu oleh *mobile robot* adalah di bidang keamanan. Alat yang masih digunakan untuk keamanan hingga saat ini salah satunya adalah kamera pengintai yang dipasang pada setiap sudut tetapi kamera pengintai ini masih terdapat kekurangan, yaitu masih terdapat sisi yang tidak bisa dilihat atau yang biasa disebut *blind spot.* Dengan adanya *Mobile robot* ini memudahkan pekerjaan manusia untuk meningkatkan keamanan dengan mengurangi *blind spot* tersebut.

Dengan media *mobile robot* tersebut penulis membuat robot yang bisa mengikuti manusia pada saat berjalan. *Mobile Robot* yang digunakan adalah *Robotino*. Robot mengikuti manusia tersebut pada saat *webcam* telah mendeteksi badan manusia. *Webcam* dari Robot tersebut mengelola data gambar yang diterima agar bisa mendeteksi badan manusia dengan *image processing* dengan metode *Haar-like feature*. Robot tersebut mendekati target badan manusia yang telah terdeteksi menggunakan logika *fuzzy* untuk kendali pergerakan robot.

Robot penjejak badan manusia ini dapat melakukan deteksi badan manusia dengan jarak deteksi yang paling efektif yaitu 1 – 4 meter dari webcam pada robot. Robot tersebut juga dapat melakukan penjejakan untuk mendekati badan manusia dengan tingkat error sebesar 9.6515875 %.

KATA PENGANTAR

Puji dan syukur kepada Allah Swt atas segala kasih dan karunia di setiap langkah kehidupan penulis, sehingga penulis dapat menyelesaikan proyek Tugas Akhir ini yang berjudul "Robot pelacak manusia menggunakan *webcam* sebagai sensor visual ". Tugas Akhir ini disusun untuk menyelesaikan Program Studi S1 Sistem Komputer di Sekolah Tinggi Manajemen Informatika dan Teknik Komputer Surabaya.

Penulis juga mengucapkan terima kasih yang sebesar-besarnya kepada:

1. Mama , Papa , Mbah dan Adik-adik yang telah memberikan segalanya.
2. Bapak Harianto, S.Kom, M.Eng selaku Dosen Pembimbing I yang telah sabar membimbing, memberikan motivasi dan saran, terutama selama proses penyusunan Tugas Akhir ini.
3. Dr. Jusak., selaku Kepala Program Studi Sistem Komputer STIKOM Surabaya yang telah membantu serta mendukung setiap kegiatan sehingga pelaksanaan Tugas Akhir ini dapat berjalan dengan baik.
4. Para dosen Program Studi S1 Sistem Komputer yang sangat membantu dalam berbagai macam kesulitan sehingga penulis dapat termotivasi untuk terus berusaha hingga Tugas Akhir ini selesai sesuai dengan harapan.
5. Saudara-saudara seperjuanganku, mahasiswa S1-SK angkatan 2010 atas segala kebersamaan selama ini, baik di waktu senang maupun susah.
6. Rekan-rekan penghuni HIMA Prodi S1-SK yang selalu kubanggakan.
7. Seluruh pihak yang tidak dapat penulis tuliskan satu persatu yang telah membantu penulis secara langsung maupun tidak langsung.

Banyak hal dalam laporan Tugas Akhir ini yang masih perlu diperbaiki lagi. Oleh karena itu penulis mengharapkan saran dan kritik yang dapat membangun dari semua pihak agar dapat menyempurnakan penulisan ini kedepannya. Penulis juga memohon maaf yang sebesar-besarnya jika terdapat kata-kata yang salah serta menyinggung perasaan pembaca. Akhir kata penulis ucapkan banyak-banyak terima kasih yang sebesar-besarnya kepada para pembaca, semoga tulisan ini dapat bermanfaat bagi para pembaca.

Surabaya, Agustus 2014

Penulis

BAB I

PENDAHULUAN

1.1. Latar Belakang Masalah

Perkembangan yang pesat pada teknologi robotika seiring dengan kebutuhan robot cerdas yang mampu membantu dan menggantikan pekerjaan manusia. *Mobile Robot* merupakan robot yang bersifat bergerak dari satu titik ke titik lainnya sehingga membutuhkan salah satu elemen dasar pada sebuah *mobile robot*, yakni kamera untuk pengolahan citra yang akan dianggap seperti sepasang mata pada robot. *Mobile robot* akan mendeteksi keberadaan benda di sekitar dan mengolahnya dengan kamera sebagai sensor visual.

Elemen pengolahan citra pada *mobile robot* menggunakan sensor visual, sensor visual ini akan dianalogikan sebagai indera penglihatan pada manusia. Sensor visual memberikan informasi visual yang kemudian diolah menjadi deskripsi objek dalam lingkungan tempat robot tersebut. Sistem visual yang akan digunakan pada *mobile robot* ini diharapkan dapat melakukan beberapa fungsi yaitu pemetaan lingkungan, pengenalan objek penghalang, penjejakan objek, pembentukan lintasan, dan pemindahan objek. Masalah utama pada sistem visual pada *mobile robot* adalah pengenalan objek. Sepanjang perjalanannya *mobile robot* akan memetakan secara langsung lingkungan yang sedang dijelajahinya serta dapat mendeteksi dan memodelkan objek yang ada.

Dengan kelebihan yang dimiliki *mobile robot* tersebut akan bisa memudahkan pekerjaan manusia. Salah satu perkerjaan manusia yang bisa dibantu

oleh *mobile robot* adalah di bidang keamanan. Dengan *mobile robot* ini akan membantu manusia untuk melihat kondisi yang ada disekitar apakah ada seseorang yang berpotensi bisa melakukan kejahatan. Alat yang masih digunakan untuk keamanan hingga saat ini salah satunya adalah kamera pengintai yang dipasang pada setiap sudut tetapi kamera pengintai ini masih terdapat kekurangan yaitu masih terdapat sisi yang tidak bisa dilihat atau *blind spot.* Dengan adanya *mobile* robot ini akan memudahkan pekerjaan manusia untuk meningkatkan keamanan dengan mengurangi *blind spot* tersebut.

Ada beberapa jurnal yang berkaitan dengan penelitian ini. Menurut Chunhua Hu, XudongMa, XianzhongDai dan KunQian (2007) pada paper berjudul, *"Reliable people tracking approach for mobile robot in indoor environments"* yang berisi membuat robot pengintai dengan mendeteksi wajah dan badan manusia bagian depan. Sedangkan menurut Marin Kobilarov dan Gaurav Sukhatme (2008) pada paper berjudul "*People tracking and following with mobile robot using an omnidirectional camera and a laser*" yang berisi tentang membuat robot penjejak dengan kamera sebagai sensor visual untuk mendeteksi badan manusia dan *ultrasonic* sebagai sensor jarak, sehingga robot yang dibuat bisa menghitung jarak antara robot dan manusia.

Dengan *mobile robot* tersebut penulis akan membuat robot yang bisa mengikuti manusia pada saat berjalan. *Mobile robot* akan mengikuti manusia tersebut pada saat sensor visual telah mendeteksi badan manusia. *Mobile robot* tersebut akan mengikuti badan manusia yang telah terdeteksi tersebut kemanapun manusia itu akan berjalan sampai *mobile robot* tersebut telah mendekati target

badan manusia yang telah terdeteksi. Setelah mendekati target tersebut robot akan berhenti.

1.2. Perumusan Masalah

Dari latar belakang yang telah diuraikan di atas, maka dapat dirumuskan permasalahan sebagai berikut :

1. Bagaimana mendeteksi manusia melalui *webcam* dengan ukuran tubuh manusia , warna pakaian, intensitas cahaya dan keadaan lingkungan yang berbeda-beda
2. Bagaimana robot akan berjalan mendekati manusia sesuai dengan tempat dimana manusia tersebut berdiri

1.3. Pembatasan Masalah

Dalam perancangan dan pembuatan perangkat ini terdapat beberapa pembatasan masalah, antara lain :

1. Sensor yang dipakai pada mobile robot adalah sensor visual yaitu *webcam*
2. Robot yang dipakai untuk *mobile robot* adalah robotino
3. Aplikasi pemrograman yang dipakai adalah Microsoft C++ dan OpenCV
4. Robot akan selalu melacak badan manusia bagian atas
5. Robot akan diaplikasikan hanya pada dalam ruangan
6. Robotino dikendalikan oleh sinyal kontrol dari PC melalui jaringan *wireless*.
7. *Haar training* yang digunakan merupakan *haar training* dari OpenCV

1.4. Tujuan

Tujuan dari perancangan dan pembuatan perangkat ini adalah sebagai berikut :

1. Menghasilkan aplikasi pendeteksi badan manusia
2. Membuat *mobile robot* yang bisa mengikuti manusia
3. Membuat *mobile robot* yang bisa memantau pergerakan manusia

1.5 Sistematika Penulisan

Laporan penelitian tugas akhir ini tersusun atas beberapa bab dengan urutan sebagai berikut :

BAB I : Pendahuluan

Pada bab satu diuraikan mengenai latar belakang dari topik Tugas Akhir yang diambil, kemudian dirumuskan menjadi suatu permasalahan yang akan diselesaikan dalam tugas akhir ini, batasan-batasan masalah yang akan diteliti, tujuan dari penelitian tugas akhir ini, kontribusi yang dapat diberikan dari hasil penelitian ini terhadap perkembangan ilmu pengetahuan, serta sistematika penulisan buku Tugas Akhir.

BAB II : Landasan Teori

Bagian landasan teori menguraikan tentang teori-teori yang terkait dengan variabel-variabel penelitian termasuk uraian tentang pemilihan suatu teori yang diterapkan dalam menyelesaikan masalah. Teori yang akan diuraikan adalah tentang sistem yang digunakan yaitu Robotino, *webcam, library* OpenCV, *library*

OpenRobotinoAPI, metode pengolahan citra untuk mendeteksi badan yaitu *Haar- like feature* dan metode *logika fuzzy* untuk logika pergerakan robot untuk mengikuti pergerakan badan manusia

BAB III : Metode Penelitian

Dalam bab tiga diuraikan tentang metode penelitian yang digunakan dalam penelitian ini serta alasan dan penjelasan penggunaan metode tersebut dalam penelitian. Pada metode penelitian ini dimuat model sistem yang akan dibuat, perancangan aplikasi serta pembuatannya, yaitu proses integrasi Robotino dengan PC dan pengolahan citra sesuai yang diharapkan, dan model pengujian dan evaluasi sistem yang digunakan.

BAB IV : Pengujian dan Evaluasi Sistem

Dalam bagian pengujian dan evaluasi sistem, diuraikan tentang langkah-langkah pengujian, tujuan pengujian, prosedur pengujian dan hasil pengujian serta analisis hasil pengujian sistem secara keseluruhan.

BAB V : Penutup

Bagian penutup merupakan bagian akhir dari laporan penelitian tugas akhir ini yang menguraikan kesimpulan-kesimpulan yang diperoleh dari proses penelitian, serta saran-saran untuk pengembangan penelitian selanjutnya.

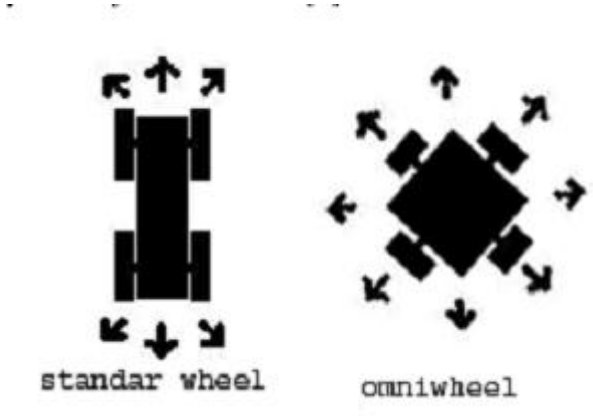
BAB II

LANDASAN TEORI

2.1. *Omni-Directional* Robot

Omni-directional robot adalah robot dengan dengan sistem pergerakan yang secara langsung dapat bergerak kesegala arah dengan konfigurasi apapun. Pada umumnya robot di desain dengan pergerakan yang sudah direncanakan terlebih dahulu. Pada sistem pergerakkan konvensional, pergerakan tidak mampu di kontrol pada setiap tingkat kebebasan dalam bergerak secara *independent*, sehingga hanya mampu bergerak ke beberapa arah yang sudah ditentukkan sebelumnya. Ini disebut kendala *non-holomic* yaitu pencegahan roda kemudi dari selip, meskipun pada umumnya mampu menjangkau setiap lokasi dan orientasi dalam ruang 2 dimensi, namun memerlukan manuver dan perencanaan jalan yang rumit dan kompleks (Doroftei,2007).

Keunggulan robot omni ini adalah pada roda yang berupa *omni directional poly roller wheel*. Pada Gambar 2.1 terlihat sebuah robot dengan *omniwheel* mampu melakukan gerakan yang kompleks untuk mencapai posisi tertentu. Dengan sistem pergerakkan ini maka robot akan memiliki 2 derajat kebebasan karena dapat bergerak pada aksis x ataupun y. Pada umumnya desain robot omni terdapat 2 jenis yaitu robot omni dengan 3 roda dan 4 roda. Pengaturan posisi *omniwheel* mempengaruhi pergerakan robot secara signifikan, jika dengan *omniwheel* standar semakin jauh jarak roda depan dengan roda belakang maka semakin cepat untuk memutar posisi robot (Syam,2011).



Gambar 2.1 Sistem Pergerakan Konvensional dan *Omni-Directional*
(Sumber : http://robotika.web.id/sistem-pergerakan-robot/)

Omniwheel terdiri dari roda inti besar dan sepanjang periferal ada terdapat banyak roda kecil tambahan yang mempunyai poros tegak lurus pada roda inti. *Omniwheel* merupakan roda *Mecanum* dengan cakram kecil di sekitar lingkar yang tegak lurus terhadap arah bergulir. Efeknya adalah bahwa roda akan berputar dengan kekuatan penuh, dan akan bergeser dengan sangat mudah. *Omniwheel* ini sering digunakan dalam sistem penggerak *holonomic*. Sistem pergerakkan ini sering digunakan dalam perlombaan seperti *RoboCup*, robot banyak menggunakan *omniwheel* ini karena memiliki kemampuan untuk bergerak ke segala arah. Roda omni sering digunakan sebagai kastor bertenaga untuk robot berkendara diferensial untuk membuat berputar lebih cepat (Syam,2011). Berikut beberapa jenis *omniwheel* seperti pada Gambar 2.2.

Gambar 2.2 Jenis *Omniwheel*
(Sumber : http://www.omnixtechnology.com/direct_components.html)

2.2. Robotino

Robotino adalah robot buatan Festo Didactic yang digunakan untuk edukasi dan penelitian serta kompetisi robot. Robotino memiliki fitur sistem gerak menggunakan *omni-directional drive, bumps sensors, infrared distance sensors,* dan *usb webcam*. Robotino didesain modular, sehingga dapat dengan mudah ditambahkan berbagai akesesoris pelengkap, seperti *sensor laser scanner, gyroscope,* dan *postioning system Northstar* dalam ruangan. (ROS, 2010). Gambar Robotino dapat dilihat pada Gambar 2.3.

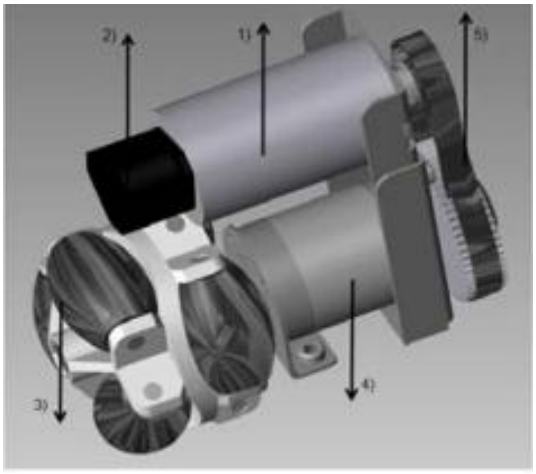
Gambar 2.3 Robotino
(Sumber : http://www.robotshop.com/ca/en/festo-robotino-mobile-robotic.html)

Robotino dapat bergerak maju, mundur dan menyamping ke segala arah, serta berputar di tempat, dengan menggunakan tiga roda. Robot ini dapat

diintegrasikan dan digunakan sebagai pilihan teknologi, misalnya untuk teknologi penggerak listrik, sensor, teknologi kontrol, pengolahan citra dan teknik pemrograman.

Robotino memiliki spesifikasi hardware sebagai berikut :

1. Diameter: 370 mm
2. Tinggi: 210 mm
3. Berat secara keseluruhan: 11 kg
4. Karet penjaga strip dengan perlindungan sensor tabrakan terpadu
5. 9 sensor jarak infra merah
6. Analog sensor induktif
7. 2 optik sensor
8. *Webcam* dengan antarmuka USB
9. *Wireless* LAN
10. Dapat diperluas dengan menggunakan dua busi 20-pin
11. 2 *Power supply* 12V akumulator gel
12. *Ethernet*, VGA dan USB untuk koneksi langsung ke *monitor* dan *keyboard*
13. *Driver Unit*, terdapat 3 unit yang terdiri dari komponen DC *motor, integrated planetary gear,omni-directional wheels,toothed belt with gear wheels*, dan incremental encoder. Kecepatan motor akan dikontrol melalui pengontrol PID yang diimplementasikan pada Atmel Microprocessor of the controller board of Robotino®. Gambar Driver Unit dapat dilihat pada Gambar 2.4.



Gambar 2.4 *Driver Unit*
(Sumber : http://www.festo-didactic.com/int-en/learning-systems/education-and-research-robots-robotino/drive-module-robotino-until-2013.htm)

2.3. *Webcam*

Web camera atau yang biasa dikenal dengan *webcam*, adalah kamera yang gambarnya bisa di akses menggunakan *world wide web* (www), program *instant messaging*, atau aplikasi komunikasi dengan tampilan *video* pada PC. *Webcam* juga digambarkan sebagai kamera *video* digital yang sengaja didesain sebagai kamera dengan resolusi rendah. *Webcam* dapat digunakan untuk sistem keamanan. Pada beberapa *webcam*, ada yang di lengkapi dengan *software* yang mampu mendeteksi pergerakan dan suara. Dengan *software* tersebut, memungkinkan PC yang terhubung ke kamera untuk mengamati pergerakan dan suara, serta merekamnya ketika terdeteksi. Hasil rekaman ini bisa disimpan pada komputer, e-mail atau di *upload* ke internet (Wibowo, 2010).

Webcam sangat bermanfaat dalam bidang telekomunikasi, bidang keamanan dan bidang industri. Sebagai contoh *webcam* digunakan untuk *video call chatting*, *surveillience camera*, dan sebagai *video conference* oleh beberapa *user*. Namun seiring perkembangan zaman, *webcam* sekarang dimanfaatkan

sebagai sensor pada robot yang digunakan sebagai mata robot yang akan menangkap gambar dan diolah dalam citra digital untuk menghasilkan gerakan yang diinginkan.

2.4. Citra Digital

Citra digital adalah citra dua dimensi yang dapat ditampilkan pada layar monitor komputer sebagai himpunan berhingga (diskrit) nilai digital yang disebut *pixel* (*picture elements*). *Pixel* adalah elemen citra yang memiliki nilai yang menunjukkan intensitas warna. Berdasarkan cara penyimpanan atau pembentukannya, citra digital dapat dibagi menjadi dua jenis. Jenis pertama adalah citra digital yang dibentuk oleh kumpulan *pixel* dalam array dua dimensi. Citra jenis ini disebut citra *bitmap* atau citra *raster*. Jenis citra yang kedua adalah citra yang dibentuk oleh fungsi-fungsi geometri dan matematika. Jenis citra ini disebut grafik vektor. Citra digital (diskrit) dihasilkan dari citra analog (*kontinu*) melalui digitalisasi. Digitalisasi citra analog terdiri s*ampling* dan *quantitazion* *Sampling* adalah pembagian citra ke dalam elemen-elemen diskrit (*pixel*), sedangkan *quantitazion* adalah pemberian nilai intensitas warna pada setiap *pixel* dengan nilai yang berupa bilangan bulat (G.W. Awcock,1996).

2.5. Pengolahan Citra Digital

Pengolahan citra merupakan teknik manipulasi citra secara digital yang khususnya menggunakan komputer, menjadi citra lain yang sesuai untuk digunakan dalam aplikasi tertentu. Agar mudah diinterpretasi oleh manusia atau komputer, pengolahan citra harus dilakukan dengan berbagai macam metode

untuk mencapai citra sesuai yang diinginkan. Operasi pengolahan citra digital umumnya dilakukan dengan tujuan memperbaiki kualitas suatu gambar sehingga dapat dengan mudah diinterpretasikan oleh mata manusia dan untuk mengolah informasi yang ada pada suatu gambar untuk kebutuhan identifikasi objek secara otomatis (Murinto, 2009).

Pengolahan citra adalah suatu metode yang digunakan untuk mengolah gambar sehingga menghasilkan gambar yang sesuai dengan keinginan kita. Pengambilan gambar bisa dilakukan dengan menggunakan *webcam* atau alat lain yang bisa digunakan untuk mentransfer gambar misalnya *scanner* atau kamera digital.

Bahasan kali ini berfokus pada pengambilan gambar menggunakan *webcam*. Sehingga citra yang dihasilkan sudah berbentuk sinyal digital dan mudah dikenali atau dibaca komputer. Citra digital adalah citra kontinyu yang sudah didiskritkan baik koordinat spasial maupun kecerahannya. Citra digital dianggap matrik dengan ukuran M x N dimana baris dan kolom menunjukkan titik-titiknya. Citra berwarna menggunakan metode RGB. Adapun masing masing warna dalam tabel memiliki 3 buah kombinasi angka yaitu R, G, dan B yang menentukan proporsi warna merah, warna hijau dan warna biru dari warna tersebut. RGB masing-masing memiliki range antara 0 hingga 63 sehingga jumlah warna yang dapat kita pilih untuk mengisi warna pada sebuah cell di tabel ialah 63x63x63 = 16 juta warna. Tetapi seluruh tabel hanya dapat diisi dengan 256 pilihan warna. Kita dapat mengubah intensitas warna dari sebuah warna pada tabel dengan cara menggunakan *interupt*. Untuk konversi warna RGB ke BGR hanya mengubah

proporsi warna dari ruang warna merah,hijau dan biru ke ruang warna biru, hijau dan merah.

2.5.1. Citra *Grayscale*

Pada dasarnya untuk membangun sebuah gambar dibutuhkan banyak sekali titik sebagai elemen dasar penyusunnya, contoh yang paling sederhana adalah garis. Jumlah maksimum titik yang membentuk segmen sgaris ditentukan oleh pengukuran resolusi piranti layarnya. Semakin tinggi resolusinya semakin banyak jumlah titik yang banyak ditampilkan dilayar sehingga tidak tampak oleh mata karena gambar yang dihasilkan sangat halus. Titik-titik kecil yang tampak dilayar komputer disebut piksel (*Picture Element*) piksel ini merupakan elemen terkecil dari tampilan layar yang dapat dikendalikan.

Dalam komputasi, suatu citra digital *grayscale* atau *greyscale* adalah suatu citra dimana nilai dari setiap piksel merupakan sample tunggal. Citra yang ditampilkan dari citra jenis ini terdiri atas warna abu-abu, bervariasi pada warna hitam pada bagian yang intensitas terlemah dan warna putih pada intensitas terkuat. Citra *grayscale* berbeda dengan citra ”hitam-putih”, dimana pada konteks komputer, citra hitam putih hanya terdiri atas 2 warna saja yaitu ”hitam” dan ”putih” saja. Pada citra *grayscale* warna bervariasi antara hitam dan putih, tetapi variasi warna diantaranya sangat banyak. Citra *grayscale* seringkali merupakan perhitungan dari intensitas cahaya pada setiap piksel pada spectrum elektromagnetik single band. Citra digital *grayscale* merupakan suatu larik dua dimensi atau suatu matrik yang elemen-elemennya menyatukan tingkat keabuan dari elemen gambar (Piksel). Sedangkan piksel itu sendiri merupakan bagian

terkecil citra yang berisi informasi tingkat keabuan suatu citra. Suatu citra digambarkan sebagai matrik ukuran N x M, dimana N menyatakan kolom dan M menyatakan baris. Letak koordinat awal citra dengan matrik piksel berbeda, koordinat x dan y suatu citra diawali pada pojok kiri bawah.

Citra *grayscale* disimpan dalam format 8 bit untuk setiap sampel piksel, yang memungkinkan sebanyak 256 intensitas. Format ini sangat membantu dalam pemrograman karena manupulasi bit yang tidak terlalu banyak. Pada aplikasi lain seperti pada aplikasi medical imaging dan remote sensing biasa juga digunakan format 10,12 maupun 16 bit.

Proses awal yang banyak dilakukan dalam Pengolahan Citra adalah mengubah citra berwarna menjadi citra *grayscale*, hal ini digunakan untuk menyederhanakan model citra. Pada awalnya citra terdiri dari 3 layer matrik yaitu R-layer, G-layer dan B-layer. Sehingga untuk melakukan proses-proses selanjutnya tetap diperhatikan tiga layer di atas. Bila setiap proses perhitungan dilakukan menggunakan tiga layer, berarti dilakukan tiga perhitungan yang sama. Sehingga konsep itu diubah dengan mengubah 3 *layer* di atas menjadi 1 *layer* matrik *grayscale* dan hasilnya adalah citra *grayscale*. Dalam citra ini tidak ada lagi warna, yang ada adalah derajat keabuan. Untuk mengubah citra berwarna yang mempunyai nilai matrik masing-masing r, g dan b menjadi citra *grayscale* dengan nilai s, maka konversi dapat dilakukan dengan mengambil rata-rata dari nilai r, g dan b.

Citra skala keabuan mempunyai nilai minimum (biasanya=0) dan nilai maksimum. Banyaknya kemungkinan nilai minimum dan maksimum bergantung pada jumlah bit yang digunakan (umumnya menggunakan 8 bit). Contohnya

untuk skala keabuan 4 bit, maka jumlah kemungkinan nilainya adalah $2^4 = 16$, dan nilai maksimumnya adalah $2^4-1 = 15$, sedangkan untuk skala keabuan 8 bit, maka jumlah kemungkinan nilainya adalah $2^8 = 256$, dan nilai maksimumnya adalah $2^8 – 1 = 255$.

Secara digital suatu *grayscale image* dapat direpresentasikan dalam bentuk array dua dimensi. Tiap elemen dalam array menunjukkan intensitas (*greylevel*) dari *image* pada posisi koordinat yang bersesuaian. Apabila suatu citra direpresentasikan dalam 8 bit maka berarti pada citra terdapat 28 atau 256 level *grayscale*, (biasanya bernilai 0 – 255), dimana 0 menunjukkan level intensitas paling gelap dan 255 menunjukkan intensitas paling terang. Tiap elemen pada array diatas disebut sebagai *picture* elemen atau sering dikenal sebagai piksel. Dengan melakukan perubahan pada intensitas pada masing-masing piksel maka representasi citra secara keseluruhan akan berubah. Citra yang dinyatakan dengan matrik M x N mempunyai intensitas tertentu pada piksel tertentu. Posisi *picture* elemen (i,j) dan koordinat (x,y) berbeda. Persamaan yang digunakan untuk mengkonversi citra berwarna menjadi citra skala keabuan (Basuki, A : 2005) `

Untuk mengubah citra berwarna yang mempunyai nilai matrik masing-masing r, g dan b menjadi citra gray scale dengan nilai s, maka konversi dapat dilakukan dengan mengambil rata-rata dari nilai r, g dan b sehingga dapat dituliskan menjadi berikut ini :

$$Gray = (R + G + B) / 3 \quad \text{.....(2.1)}$$

Konversi informasi suatu citra warna ke skala keabuan dapat juga dilakukan dengan cara member bobot pada setiap elemen warna (Achmad: 2005), sehingga persamaan diatas dimodifikasi menjadi rumus berikut ini :

$$Gray = {}_{WR}R + {}_{WG}G + {}_{WB}B \text{ .....} (2.2)$$

Dengan $w_{R,}$ $w_{G,}$ dan w_B masing-masing adalah bobot untuk elemen warna merah, hijau dan biru. NTSC (*National Television System Committee)* mendefinisikan bobot untuk konversi citra warna ke skala keabuan adalah sebagai berikut:

WR =0,299 WG = 0,587 WB = 0,114

Untuk citra berwarna nilai dari suatu piksel misal adalah X, maka untuk mendapat nilai ***Red, Green, Blue*** dapat menggunakan rumus berikut ini :

$$Blue = X/\ 2^{16} \text{.....} (2.3)$$

$$Green = (X - Blue * 2^{16})/2^{8} \text{.....} (2.4)$$

$$Red = X - Blue * 2^{16} - Green * 2^{8} \text{.....} (2.5)$$

2.6. Logika Fuzzy

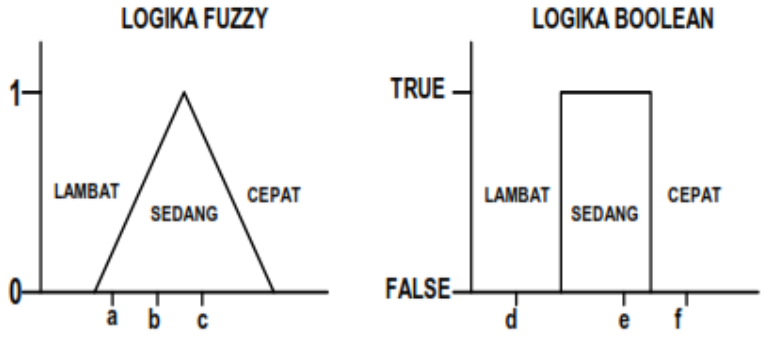
Logika *fuzzy* adalah cabang dari sistem kecerdasan buatan (*Artificial Intelegent*) yang mengmanipulasi kemampuan manusia dalam berpikir ke dalam bentuk algoritma yang kemudian dijalankan oleh mesin. Algoritma ini digunakan dalam berbagai aplikasi pemrosesan data yang tidak dapat direpresentasikan dalam bentuk biner. Logika *fuzzy* menginterpretasikan statemen yang samar menjadi sebuah pengertian yang logis.

Logika *Fuzzy* pertama kali diperkenalkan oleh *Prof. Lotfi Zadeh* seorang kebangsaan Iran yang menjadi guru besar di *University of California at Berkeley* pada tahun 1965 dalam papernya yang monumental. Dalam paper tersebut dipaparkan ide dasar *fuzzy set* yang meliputi *inclusion, union, intersection, complement, relation* dan *convexity*. Pelopor aplikasi *fuzzy set* dalam bidang

kontrol, yang merupakan aplikasi pertama dan utama dari *fuzzy set* adalah *Prof.* Ebrahim Mamdani dan kawan-kawan dari *Queen Mary College London*. Penerapan kontrol *fuzzy* secara nyata di industri banyak dipelopori para ahli dari Jepang, misalnya Prof. Sugeno dari *Tokyo Institute of Technology*, *Prof.Yamakawa* dari *Kyusu Institute of Technology*, *Togay* dan *Watanabe* dari *Bell Telephone Labs*.

2.6.1. Himpunan Fuzzy

Himpunan *fuzzy* merupakan suatu pengembangan lebih lanjut tentang konsep himpunan dalam matematika. Himpunan *Fuzzy* adalah rentang nilai-nilai. Masing-masing nilai mempunyai derajat keanggotaan (*membership*) antara 0 sampai dengan 1. Ungkapan logika *Boolean* menggambarkan nilai-nilai "benar" atau "salah". Logika *fuzzy* menggunakan ungkapan misalnya : "sangat lambat", "agak sedang", "sangat cepat" dan lain-lain untuk mengungkapkan derajat intensitasnya. Ilustrasi antara keanggotaan *fuzzy* dengan *Boolean set* dapat dilihat pada Gambar 2.5 :



Gambar 2.5 Ilustrasi antara logika fuzzy dan logika bolean
(Sumber :
http://file.upi.edu/Direktori/FPTK/JUR._PEND._TEKNIK_ELEKTRO/)

Logika *fuzzy* menggunakan satu set aturan untuk menggambarkan perilakunya. Aturan-aturan tersebut menggambarkan kondisi yang diharapkan dan hasil yang diinginkan dengan menggunakan *statemen IF… THEN*.

2.6.2 *Interferencing (Rule Base)*

Pada umumnya, aturan-aturan *fuzzy* dinyatakan dalam bentuk *"IF…THEN"* yang merupakan inti dari relasi *fuzzy*. Relasi *fuzzy*, dinyatakan dengan *R*, juga disebut implikasi *fuzzy*. Untuk mendapatkan aturan *"IF…..THEN"* ada dua cara utama :

1. Menanyakan ke operator manusia yang dengan cara manual telah mampu mengendalikan sistem tersebut, dikenal dengan *"human expert"*.
2. Dengan menggunakan algoritma pelatihan berdasarkan data-data masukan dan keluaran.

Dalam penalaran logika *fuzzy*, ada dua tipe utama untuk pengambilan keputusan *fuzzy* yaitu *Generalized Modus Ponens (GMP)* dan *Generalized Modus Tolens (GMT)*. *GMP* disebut juga dengan *direct reasoning*, sedangkan *GMT* disebut juga *indirect reasoning.* Jika himpunan *fuzzy* dinotasikan dengan *A, A', B, B'* dan variabel linguistik dinotasikan dengan *x* dan *y*, maka *GMP* dan *GMT* dapat dinyatakan sebagai berikut :

Generalized Modus Ponens (GMP) :

Pernyataan 1 (aturan) : *if x* is *A then y* is *B*

Pernyataan 2 (fakta) : *x* is *A*'

Penyelesaian : *y* is *A*'

Dalam hal ini penyelesaian *B*' dapat dinotasikan dengan :

$B' = A' \; o \; R$

Dengan *R* adalah relasi *fuzzy* dari implikasi *fuzzy* '*if A then B'*, tanda o adalah operator komposisi, dan *A*' adalah himpunan *fuzzy* yang mempunyai bentuk : sangat A, lebih atau kurang A, tidak A dan sebagainya.

Generalized Modus Tolens (GMT) :

Pernyataan 1 (aturan) : if *x* is *A* then *y* is *B*

Pernyataan 2 (fakta) : *x* is *B'*

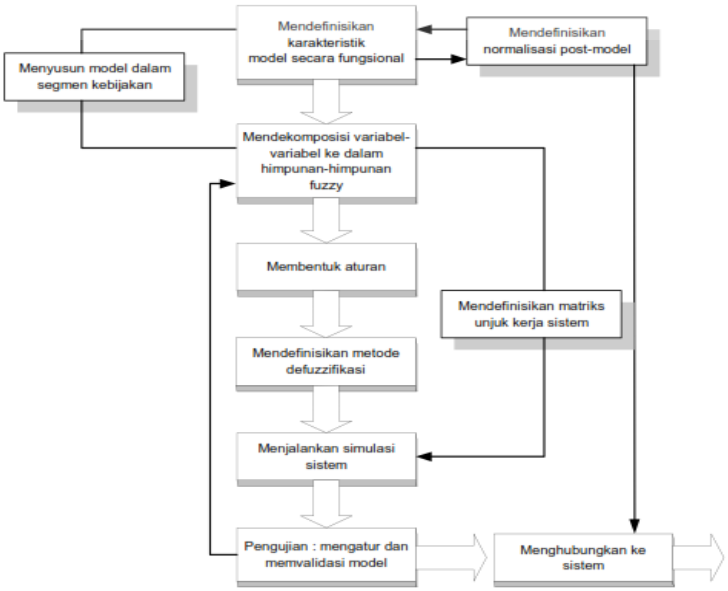
Penyelesaian : *y* is *A'*

Dalam hal ini penyelesaian *B'* dapat dinotasikan dengan :

$A' = R \; o \; B'$

2.6.3 Metodologi Desain Sistem *Fuzzy*

Secara garis besar untuk perancangan suatu sistem fuzzy perlu dilakukan beberapa tahapan yaitu pada Gambar 2.6:



Gambar 2.6 Metodologi Pengembangan Sistem Fuzzy
(Sumber : ttp://informatika.stei.itb.ac.id/~rinaldi.munir/MetNum/2011-2012/Pengantar%20Logika%20Fuzzy.pdf)

2.6.4 Fuzzyfikasi

Proses fuzzyfikasi merupakan proses untuk mengubah variabel *non fuzzy* (variable numerik) menjadi variabel *fuzzy* (variabel linguistik). Nilai masukan-masukan yang masih dalam bentuk variabel numerik yang telah dikuantisasi sebelum diolah oleh pengendali *fuzzy* harus diubah terlebih dahulu ke dalam variabel *fuzzy*. Melalui fungsi keanggotaan yang telah disusun maka nilai-nilai masukan tersebut menjadi informasi *fuzzy* yang berguna nantinya untuk proses pengolahan secara *fuzzy* pula. Proses ini disebut fuzzyfikasi.

Dengan kata lain fuzzyfikasi merupakan pemetaan titik-titik numerik *(crisp point)* ke himpunan fuzzy *A* di *U*. *U* adalah semesta pembicaraan.

2.6.5 Defuzzyfikasi

Keputusan yang dihasilkan dari proses penalaran masih dalam bentuk *fuzzy*, yaitu berupa derajat keanggotaan keluaran. Hasil ini harus diubah kembali menjadi varibel numerik *non fuzzy* melalui proses defuzzifikasi.

Model defuzzyfikasi metode Sugeno :

Dalam metode sugeno, output sistem tidak berupa himpunan *fuzzy*, melainkan berupa konstanta atau persamaan linier. Metode ini diperkenalkan oleh Takagi-Sugeno Kang pada tahun 1985.

Secara umum model *fuzzy* SUGENO terdiri dari dua jenis :

1. Model *fuzzy* SUGENO orde-nol :

IF input1 = x dan input2 = y, *THEN* Outputnya adalah z = k.

2. Model *fuzzy* SUGENO orde-satu :

IF input1 = x dan input2 = y, *THEN* Outputnya adalah z = ax+by+c

Defuzzyfikasi dilakukan dengan cara mencari rata-ratanya *(weight average/wtaver)* seperti pada Rumus berikut ini:

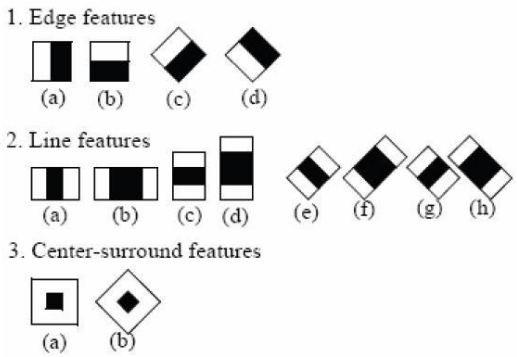
$$\text{Final Output} = \frac{\sum_{i=1}^{N} W_i z_i}{\sum_{i=1}^{N} Wi} \quad \text{..... (2.6)}$$

2.7 *Haar-like Feature*

Haar-like feature yang dikenal sebagai *Haar Cascade Classifier*. *Haar-like features* merupakan *rectangular features,* yang memberikan indikasi secara spesifik pada sebuah gambar atau *image*. Ide dari *Haar-like features* adalah untuk mengenali obyek berdasarkan nilai sederhana dari fitur tetapi bukan merupakan nilai piksel dari *image* obyek tersebut. Metode ini memiliki kelebihan yaitu komputasinya sangat cepat, karena hanya bergantung pada jumlah piksel dalam persegi bukan setiap nilai piksel dari sebuah *image*. Metode ini merupakan metode yang menggunakan *statistical model* (*classifier*). Pendekatan untuk mendeteksi objek dalam gambar menggabungkan empat konsep utama :

1. Training data
2. Fitur segi empat sederhana yang disebut fitur Haar.
3. Integral image untuk pendeteksian fitur secara cepat.
4. Pengklasifikasi bertingkat (Cascade classifier)

Haar-like Feature adalah fitur yang didasarkan pada *Wavelet Haar*. *Wavelet Haar* adalah gelombang tunggal bujur sangkar (satu interval tinggi dan satu interval rendah). Untuk dua dimensi, satu terang dan satu gelap. Selanjutnya kombinasi-kombinasi kotak yang digunakan untuk pendeteksian objek visual yang lebih baik. Setiap *Haar-like feature* terdiri dari gabungan kotak - kotak hitam dan putih. Seperti pada Gambar 2.7 berikut ini:



Gambar 2.7 Macam-macam variasi *feature* pada metode Haar
(Sumber :
http://docs.opencv.org/modules/objdetect/doc/cascade_classification.html)

3 tipe kotak(rectangular) feature:

1. Tipe *two-rectangle feature* (horisontal/vertikal)

2. Tipe *three-rectangle feature*

3. Tipe *four-rectangle feature*

Adanya fitur Haar ditentukan dengan cara mengurangi rata-rata piksel pada daerah gelap dari rata-rata piksel pada daerah terang. Jika nilai perbedaannya itu diatas nilai ambang atau *treeshold*, maka dapat dikatakan bahwa fitur tersebut ada. Nilai dari *Haar-like feature* adalah perbedaan antara jumlah nilai-nilai piksel *gray level* dalam daerah kotak hitam dan daerah kotak putih:

F(x) = Sum black rectangle – Sum White Rectangle ..... (2.7)

Dimana untuk kotak pada Haar-like feature dapat dihitung secara cepat menggunakan "*integral image*".

2.7.1 *Haar-training*

Haar training digunakan untuk menyiapkan data-data apa saja yang akan dideteksi dengan metode Haar. Ada dua aplikasi di OpenCV untuk training *classifier cascade* : opencv_haartraining dan opencv_traincascade. opencv_traincascade adalah versi yang lebih baru, yang ditulis dalam bahasa pemrograman C + + sesuai dengan *library* OpenCV 2.x API . Namun perbedaan utama antara dua aplikasi ini adalah bahwa opencv_traincascade bisa melakukan training Haar [Viola2001] dan LBP [Liao2007] (Pola Binary Lokal) fitur . Fitur LBP adalah bilangan bulat berbeda dengan Haar, sehingga kedua training dan dan proses deteksi dengan LBP beberapa kali lebih cepat dibandingkan dengan fitur Haar. Mengenai kualitas deteksi LBP dan Haar , itu tergantung pada prosese training(kualitas data yang digunakan untuk *training*.

File xml dibuat dengan suatu *training* yang dikenal dengan *Haar Training*. Proses *training* terdiri dari beberapa tahap berikut ini :

a. Persiapan DataSet

Dataset terdiri dari 2 buah sampel. Sample positif adalah gambar yang mengandung obyek yang akan dideteksi. Jika kita menginginkan wajah untuk dideteksi maka sampel positif berisi gambar – gambar wajah. Sample negatif adalah gambar yang tidak mengandung obyek yang akan dideteksi. Seperti gambar pegunungan, mobil dsb. Masukkan sample positif pada 1 direktori, misalnya positiveSample/rawdata. Sedangkan sample negatif, dimasukkan pada /negativeSample. Sampel paling minimal 10 buah untuk 1 wajah dalam berbagai pose. Catatan: *file* gambar harus *file* *.bmp.

b. Membuat Sampel *Negative*

Gunakan create_list.bat pada *folder* /negativeSample untuk mencatatkan nama *file* sample negative pada infofile.txt.

c. Membuat Sampel *Positive*

Kemudian jalankan program objectmarker.exe pada folder *positiveSample*. Ketika program ini dijalankan, maka akan muncul satu per satu file dari sample *positiveSample*. Kemudian tandai obyek yang dimaksud dari gambar tersebut dengan menggerakkan kursor *mouse* membentuk sebuah box persegi panjang. Kemudian tekan spasi untuk menambahkan *box* tersebut, lalu tekan *enter* untuk beralih pada *file* gambar berikutnya. Kalau berhasil, maka info.txt akan berisi data gambar

d. Membuat file vector.vec dari Sampel *Positive*

Lalu kita gunakan *tool* createsamples.exe untuk mengubah obyek gambar ke *file* vec. Jalankan perintah berikut pada dos command.

createsamples.exe -info positive/info.txt -vec data/vector.vec -num 18 -w 20 -h 20

Berikut penjelasan kode untuk membuat vector.vec pada Tabel 2.1 :

Tabel 2.1

Penjelasan Kode Untuk Membuat Vector.vec

Kode	Penjelasan
-info <nama *file*>	Mengambil data gambar
-num <Jumlah sampel>	Jumlah sampel yang akan diambil untuk *training*

-w < lebar sampel >	Lebar *pixel* pada sampel gambar yang akan digunakan
-h <tinggi sampel>	Tinggi *pixel* pada sampel gambar yang akan digunakan
-vec <nama file *vector*>	File *binary* yang menunjukan banyak sampel

e. Memulai *HaarTraining*

Setelah kita punya *file* vector.vec maka kita mulai *haartraining*.

Jalankan program haartraining.exe di dos command:

haartraining.exe -data data/cascade -vec data/vector.vec -bg negativeSample/infofile.txt -npos 12 -nneg 2 -mem 1000 -mode ALL -w 20 -h 20 –nonsym

lalu pada folder tools/temp/data/cascade maka akan muncul folder mulai dari 0 sampai N. Kemudian copy semua folder tersebut pada tools/cascade2xml/data.

f. Membuat file *.xml

Jalankan haarconv.exe pada folder /cascade2xml di *dos command* sebagai berikut:

haarconv.exe data output.xml 20 20

Berikut keterangan kode untuk membuat *file* .xml pada Tabel 2.2 :

Tabel 2.2

Keterangan Kode Membuat *file* .xml

Kode	Keterangan
- Data <direktori *file*>	Direktori *file* yang akan disimpan
- Vec <nama *file* vector>	Informasi sampel (.dat)
-npos < jumlah sampel positif >	Jumlah positif sampel
- Bg <nama *file background*>	Berisi sampel negative
-nneg <jumlah sampel negative>	Jumlah sampel negatif
-nstages <jumlah tahapan>	Jumlah tahapan dalam cascade selama proses trainin
-men <jumlah kapasitas memori (mb) >	Memori yang disediakan untuk proses
-nonsys /-sys	Tidak simetris / simetris
-w <lebar pada sampel>	Lebar sampel
-h <tinggi pada sampel>	Tinggi pada sampel

Jika berhasil, maka akan muncul *file* output.xml pada folder /cascade2xml.

2.8 *Computer Vision*

Computer Vision adalah pencitraan komputer dimana aplikasi tidak melibatkan manusia dalam proses pengulangan visual. Dengan kata lain, gambar yang diperiksa dan di olah oleh komputer. Meskipun orang yang terlibat dalam

pengembangan sistem aplikasi, akhirnya membutuhkan komputer untuk mengambil informasi visual secara langsung (Umbaugh, 1998).

Computer vision merupakan sebuah proses otomatis yang menintegrasikan sejumlah besar proses persepsi visual, seperti pengolahan citra, klasifikasi citra, pengenalan citra dan akusisi citra. *Computer vision* didefinisikan sebagai salah satu cabang ilmu pengetahuan yang mempelajari bagaimana komputer dapat mengenali obyek yang diamati atau diobservasi. Cabang ilmu ini bersama kecerdasan buatan (*Artificial Intelligence*) akan mampu menghasilkan sistem kecerdasan visual (*Visual Intelligence System*) (Munir, 2004).

Vision = Geometri + Measurement + Interpretatio……(2.8)

Proses-proses dalam *computer vision* dapat dibagi menjadi tiga aktivitas:

a. Memperoleh atau mengakuisisi citra digital.
b. Melakukan teknik komputasi untuk memperoses atau memodifikasi data citra.
c. Menganalisis dan menginterpretasi citra dan menggunakan hasil pemrosesan untuk tujuan tertentu, misalnya memandu robot, mengontrol peralatan, memantau proses manufaktur, dan lain-lain.

2.9 OpenCV

OpenCV (*Open Computer Vision*) adalah sebuah API (*Application Programming Interface*) *library* yang sudah sangat familiar pada pengolahan citra *computer vision*. *Computer vision* itu sendiri adalah salah satu cabang dari bidang ilmu pengolahan citra (*Image Processing*) yang memungkinkan komputer dapat melihat seperti manusia. Dengan *computer vision* tersebut komputer dapat mengambil keputusan, melakukan aksi, dan mengenali terhadap suatu objek.

Beberapa pengimplementasian dari c*omputer vision* adalah *face recognition, face detection, face/pbject tracking, road tracking*, dll.

OpenCV adalah *library open sourc*e untuk *computer vision* untuk C/C++, OpenCV didesain untuk aplikasi *real-time*, memiliki fungsi-fungsi akuisisi yang baik untuk *image/video*. OpenCV juga menyediakan *interface* ke *Integrated Performance Primitives* (IPP) Intel sehingga jika anda bisa mengoptimasi aplikasi *computer vision* anda jika menggunakan prosesor Intel (Syafi'i,2011).

Fitur yang dimiliki OpenCV antara lain :

1. Manipulasi data citra (*alocation, copying, setting, convert*).
2. Citra dan video I/O (*file* dan kamera *based input, image/video* file *output*).
3. Manipulasi Matriks dan Vektor beserta rutin-rutin aljabar linear (*products, solvers, eigenvalues*, SVD).
4. Data struktur dinamis (*lists, queues, sets, trees, graphs*).
5. Pemroses citra fundamental (*filtering, edge detection, corner detection, sampling and interpolation, color conversion, morphological operations, histograms, image pyramids*).
6. Analisis struktur (*connected components, contour processing, distance Transform, various moments, template matching, Hough Transform, polygonal approximation, line fitting, ellipse fitting, Delaunay triangulation*).
7. Kalibrasi kamera (*calibration patterns, estimasi fundamental matrix, estimasi homography, stereo correspondence*).
8. Analisis gerakan (*optical flow, segmentation, tracking*).
9. Pengenalan obyek (*eigen-methods*, HMM).

10. Graphical *User* Interface (display *image/video*, penanganan keyboard dan *mouse handling, scroll-bars*).

OpenCV terdiri dari 3 *library*, yaitu:

1. CV : Untuk algoritma *Image Processing* dan *Vision*
2. Highgui : Untuk GUI, *Image* dan *Video* I/O
3. CXCORE : Untuk struktur data, support XML dan fungsi-fungsi grafis.

2.10 OpenRobotino API

OpenRobotinoAPI (*Application Programming Interface*) adalah *library* aplikasi *programming* yang dibuat khusus untuk Robotino yang diciptakan untuk mempermudah *user* dalam membuat program pada Robotino. *Library* ini memungkinkan akses penuh terhadap sensor pada Robotino. Komunikasi antara Robotino dengan PC melalui jaringan TCP dan UDP, dan semuanya sangat transparan, meskipun program yang berjalan sudah tertanam pada Robotino ataupun secara *remote* .

BAB III

METODE PENELITIAN

3.1. Model Pengembangan

Tujuan dari tugas akhir ini akan membangun sebuah aplikasi untuk menjalankan robot yang akan melacak manusia yang ada disekitar robot tersebut. Robot yang akan digunakan adalah robot *omni-directional* dimana robot ini bisa bergerak dari satu titik ke titik lain sehingga mempermudah dalam melacak badan manusia. robot *omni-directional* ini akan ditambahkan dengan perangkat sensor visual yaitu webcam sehingga mempermudah robot ini untuk melacak badan manusia. Dari webcam tersebut akan keluar output berupa gambar, gambar tersebut akan diolah dengan menggunakan Microsoft Visual C++ 2008 dan openCV untuk bisa mendeteksi objek yang akan diharapkan yaitu badan manusia. Setelah mendapatkan objek badan manusia tersebut, dengan modul RobotinoAPI dan microsoft visual C++ 2008 akan memberikan instruksi pada robot untuk melacak pergerakan badan manusia yang telah terdeteksi.

3.2. Prosedur Penelitian

Prosedur penelitian yang dipakai dalam pengerjaan tugas akhir ini adalah:

1. Studi literatur

Pencarian data-data literatur dari masing-masing fungsi pada *library* Microsft visual C++ 2008,openCV ,OpenRobotinoAPI dan logika *fuzzy* melalui pencarian dari internet, dan konsep-konsep teoritis dari buku-buku penunjang serta metode yang digunakan untuk melakukan pengolahan citra.

2. Tahap perancangan dan pengembangan sistem

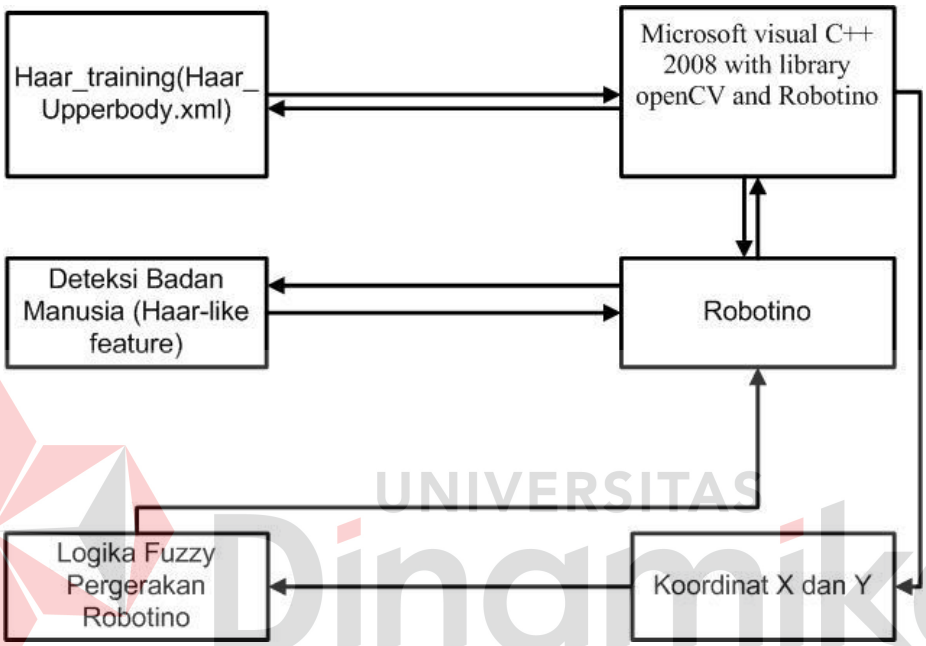
Dalam membuat pengembangan sistem, terdapat beberapa langkah rancangan sistem yang diambil antara lain:

a. Membuat *flowchart* pada proses sistem secara keseluruhan proses pengolahan citra, proses deteksi badan manusia menggunakan metode *Haar-like feature*, dan proses *tracking.*
b. Membuat aplikasi pendeteksi badan manusia bagian atas
c. Mengatur fungsi-fungsi yang digunakan pada aplikasi ini dan mengelompokkan fungsi tersebut pada beberapa *class*
d. Melakukan koneksi antara komputer dengan robotino.
e. Pengambilan gambar secara *real-time*
f. Melakukan proses pengolahan citra, yaitu konversi ke *BGR* dan *Grayscale.*
g. Menerapkan metode *Haar-like feature.*
h. Mengikuti pergerakan badan manusia yang telah terdeteksi.
i. Melakukan percobaan pada aplikasi ini untuk memastikan apakah aplikasi ini sudah dapat berjalan dengan baik atau belum.

3.3. Diagram Blok Sistem

Sistem ini terdiri dari 2 proses utama yaitu blok proses *detection* dan proses *tracking*. Proses deteksi terdiri dari haar training, Aplikasi *tracking*, robotino dan objek badan manusia. Aplikasi *tracking* akan melakukan proses pengolahan citra untuk mendeteksi objek yang sesuai dengan hasil proses *training haar-like feature.*

Sedangkan pada pada proses *tracking*, aplikasi t*racking* akan menerima hasil deteksi haar-like feature berupa koordinat X dan Y dimana koordinat ini akan dikelola dengan proses *fuzzy*, hasil dari proses *fuzzy* ini yang akan digunakan untuk arah pergerakan robotino. Untuk mempermudah dalam memahami sistem yang akan dibuat dapat dijelaskan melalui blok diagram pada Gambar 3.1.



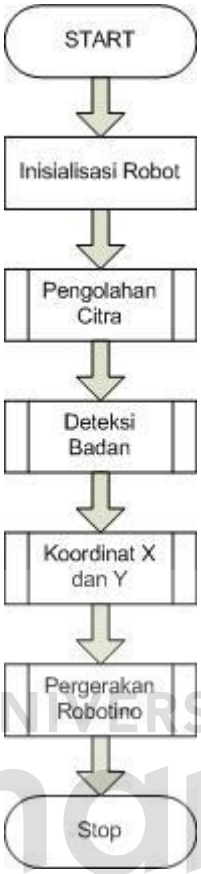
Gambar 3.1 Blok diagram sistem secara umum

3.4. Perancangan Perangkat Lunak

Dalam perancangan perangkat lunak, *compiler* yang digunakan adalah Microsoft Visual C++ 2008. Untuk *library* yang digunakan pada pengolahan citra yaitu *library* OpenCV v2.3 dan *library* OpenRobotinoAPI digunakan untuk mengintegrasikan Robotino dengan PC, sehingga seluruh modul-modul dan sensor di dalamnya dapat diakses.

Kemudian dalam penulisannya atau dalam pembuatan program, akan meliputi bagian-bagian penting dalam setiap langkah-langkah per bagian sesuai

dengan algoritma atau logika sekuensial dari awal sampai output. Berikut adalah algoritma program secara global.



Gambar 3.2 *Flowchart* sistem secara global

3.5. Inisialisasi Robot

Tahap-tahap inisialisai Robotino meliputi cara-cara *setting* koneksi Robotino, pergerakan Robotino, penerimaan data citra serta *streaming* citra Robotino. Untuk kendali Robotino digunakan OpenRobotinoAPI (*Application Programming Interface*) yaitu *library* aplikasi *programming* yang dibuat khusus untuk Robotino, yang diciptakan untuk mempermudah *user* dalam membuat program pada Robotino. *Library* ini memungkinkan akses penuh terhadap sensor dan *actuator* pada Robotino. Komunikasi antara Robotino dengan PC melalui

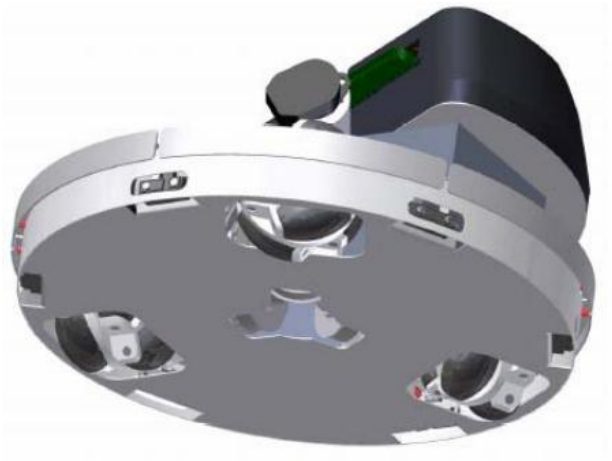
jaringan TCP atau UDP menggunakan media *wireless.* Untuk mengakses OpenRobotinoAPI pada program harus menggunakan *header* Robotino untuk bahasa *Microsoft Visual C++*, yaitu *using* rec.robotino.com.

3.5.1 Koneksi Robotino

Untuk menghubungkan koneksi *wireless* dari PC ke *access point* Robotino digunakan prototipe fungsi com.setAddress(const char* address) digunakan untuk memberikan alamat IP yang akan diakses. Untuk simulasi pada program Robotino® SIM dapat digunakan alamat IP 128.0.0.1:8080. Untuk koneksi langsung ke *access point* Robotino, secara *default* alamat IP yang digunakan adalah 172.26.1.1. Untuk mengakses fungsi tersebut dibutuhkan deklarasi *header* file yang terletak pada rec::Robotino::com:Com.h, dimana pada awal program harus dideklarasikan terlebih dahulu.

3.5.2 Pergerakan Robotino

Robotino memiliki sistem pergerakan *omni-directional drive* dengan menggunakan 3 roda yang terpasang, sehingga memungkinkan robot ini untuk bergerak maju, mundur, ke samping, dan ke segala arah serta berputar ditempat. Berikut gambar sistem *omni-directional drive* Robotino pada Gambar 3.13.



Gambar 3.3 *Omni-Directional Drive* Pada Robotino
(Sumber : http://www.ros.org/news/robots/mobile-robots/)

Untuk pergerakan Robotino digunakan fungsi setVelocity yang memiliki parameter vx, vy, dan omega atau (Z). Parameter vx adalah parameter kecepatan pada sumbu x dan vy adalah parameter kecepatan untuk sumbu y, dengan ketentuan parameter vx dan vy dalam satuan mm/s. Dan omega merupakan parameter kecepatan sudut dengan ketentuan parameter omega dalam satuan deg/s.

3.5.3 Penerimaan Data Citra

Setiap data citra yang dikirimkan dari *webcam* Robotino diakses dengan *pointer* bertipe *const unsigned char*. Karena resolusi *default* dari Robotino adalah 320x240 maka data untuk 1 citra yang dikirimkan adalah sebanyak 230400, dimana pada 1 piksel citra terdapat 3 *channel*, dan pada setiap *channel* berukuran 8bit. Ketika fungsi *setStreaming* bernilai *true*, fungsi *imageReceivedEvent* akan terus melakukan *streaming* citra hingga koneksi diputuskan atau saat *setStreaming* bernilai *false*. Berikut potongan program untuk menampilkan gambar hasil dari *webcam* pada Robotino:

```
void image(const unsigned char *data,unsigned int
dataSize,unsigned int width,unsigned int height)
{
IplImage *ima =
cvCreateImage(cvSize(width,height),IPL_DEPTH_8U,3);
          for (int i = 0; i < dataSize; i++)
          {
               ima->imageData[i] = *(data+i)
          }

          }

     img2=cvCloneImage(ima);

     img1=cvCloneImage(img2);
```

Selain menggunakan webcam robotino, program ini juga bisa mendeteksi badan melalui webcam dari *personal computer*. Untuk bisa mengakses webcam dibutuhkan fungsi kode pemrograman cvCapture* CvCaptureFromCAM(int device) dan cvQueryFrame(CvCapture* capture). Int device merupakan dari *webcam* yang melakukan perekaman gambar, jika ada hanya satu webcam maka berikan id 0. cvQueryFrame merupaka kode fungsi pemograman yang digunakan untuk menggabungkan fungsi kode pemrograman cvGrabFrame dan CvRetrieveFrame dalam satu fungsi kode pemrograman. Dalam cvQueryFrame tidak bisa untuk memodifikasi gambar yang telah diambil maka kita harus menduplikasi gambar yang telah diambil dengan fungsi kode pemrograman cvCloneImage(). Sedangkan untuk merubah ukuran frame yang digunakan menggunakan fungsi kode pemrograman cvResize(const CvArr* **src**, CvArr* **dst**, int **interpolation**=CV_INTER_LINEAR). Berikut potongan program untuk menampilkan gambar hasil dari webcam pada *personal computer* :

```
CvCapture *capture = cvCaptureFromCAM( 0 );

IplImage * webcam = cvQueryFrame(capture);
```

```
IplImage * clone = cvCloneImage(webcam);
IplImage*img2=cvCreateImage(cvSize(width,height),IPL
_DEPTH_8U,3);
cvResize(clone,img2,CV_INTER_LINEAR);
img1=cvCloneImage(img2);
```

Data citra yang ditangkap dari webcam robotino merupakan data citra dengan format BGR(Blue, Green, Red). Sedangkan format citra yang berasal dari webcam *personal computer* merupakan format citra RGB, format citra RGB ini akan di konversi ke dalam format citra BGR. Format citra ini dipakai karena format citra ini tidak banyak terpengaruhi oleh intensitas cahaya. Dari format BGR ini akan dikonversikan ke mode citra *Grayscale.* Hal ini dilakukan untuk bisa mendeteksi badan manusia bagian atas. Dari format citra Grayscale akan dikonversi ke bentuk histogram ekualisasi. Untuk merubah format citra dalam histogram ekualisasi digunakan kode fungsi pemrograman cvEqualizeHist(Int Src, Int dst). Berikut potongan program untuk menampilkan hasil gambar yang dikonversi ke berbagai bentuk format citra.

```
cvCvtColor(img2,img1, CV_BGR2RGB);
cvCvtColor(img1,thresholded, CV_BGR2GRAY);
cvEqualizeHist( thresholded, thresholded );
```

cvCvtColor merupakan sebuah fungsi kode pemrograman OpenCV untuk melakukan konversi ruang warna ke berbagai ruang warna lainnya. Berikut penjelasan dari fungsi kode pemrograman tersebut :

cvtColor(InputArray src**, OutputArray** dst**, int** code**)**

Keterangan :

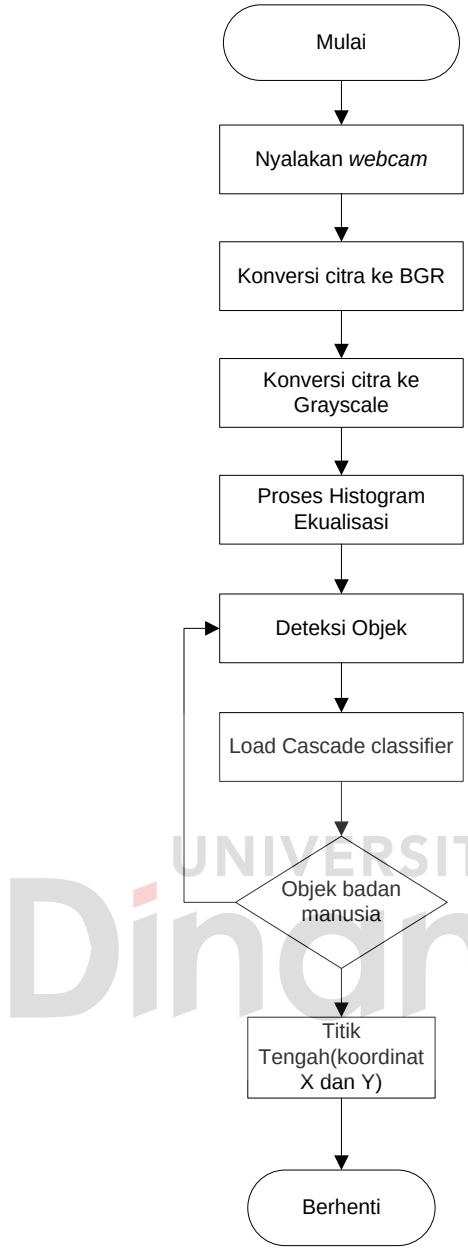
Src : Input gambar yang akan dikonversi

Dst : Output gambar yang telah dikonversi

Code : Kode ruang warna konversi

3.6 . Proses Deteksi Badan Manusia

Proses deteksi badan manusia bagian atas menggunakan metode *Haar-like feature* dimana proses deteksi akan membandingkan data perhitungan perbedaan *pixel* warna gelap dan warna terang pada hasil *training Haar-like feature*. Maka untuk melakukan proses deteksi badan manusia bagian atas ini diperlukan proses konversi ruang warna dari RGB ke *Threshold*. Untuk mendapatkan data gambar yang lebih jelas dilakukan proses histogram ekualisasi. *Haar Training* yang digunakan untuk pendeteksian badan manusia bagian atas ini menggunakan haar training yang disediakan oleh OpenCV. Berikut diagram alir dari proses deteksi badan manusia bagian atas yang digunakan pada sistem ini dapat dijelaskan pada Gambar 3.4



Gambar 3.4 Diagram alir proses deteksi badan manusia

Berikut penjelasan dari diagram alir porses deteksi badan manusia beserta kode pemrograman menggunakan metode deteksi *Haar-like feature* :

1. Proses menyalakan *webcam* dan melakukan konversi ke berbagai ruang warna seperti yang dijelaskan pada sub bab 3.5.3.

2. *Webcam* akan melakukan proses *scanning* objek pada frame gambar yang didapatkan
3. Mengambil data *training* *Haar cascade classifier* berupa *file* .xml pada program dengan fungsi kode pemrograman `CascadeClassifier::load`(const string& **filename**), Nama *file* yang dipanggil berupa *file* *.xml yang berada pada direktori proyek yang disimpan. Berikut penggalan kode pemrograman untuk mengambil data *training* :

```
String full_cascade_name = "haarcascade_upperbody.xml";
CascadeClassifier full_cascade;
```

4. Proses selanjutnya adalah membandingkan nilai *pixel* threshold pada frame objek yang tertangkap kamera dengan nilai *pixel* haar training. *Haar-Like feature* memproses gambar dalam wilayah kotak-kotak yang berisi beberapa *pixel* dari sebuah bagian gambar. Kemudian *pixel-pixel* dalam satu wilayah tersebut dijumlahkan dan dilakukan proses perhitungan sehingga didapatkan perbedaan dalam setiap wilayah kotak-kotak tersebut. Perbedaan inilah yang dapat dijadikan sebuah kode untuk menandai wilayah tersebut sehingga dapat memproses bagian gambar. Untuk dapat mendeteksi objek yang diinginkan digunakan kode fungsi pemrograman :

```
detectMultiScale(const Mat& image, CvHaarClassifierCascade* cascade, double scaleFactor=1.1, int minNeighbors=3, int flags=0, Size minSize=Size(), Size maxSize=Size()).
```

Keterangan :

Image : *Frame* gambar yang telah dikonversi ke dalam ruang warna *Grayscale*

Cascade : Haar training yang digunakan

Scale : Ukuran gambar yang akan dikurangi pada setiap skala

minNeighbors : Banyaknya objek yang akan terdeteksi oleh metode Haar

MinSize : Ukuran minimal objek yang akan dideteksi

MaxSize : Ukuran maksimal objek yang akan dideteksi

Berikut penggalan kode pemrograman untuk deteksi badan manusia bagian atas :

```
full_cascade.detectMultiScale(thresholded,full,1.1,1
,0|CV_HAAR_SCALE_IMAGE, Size(50,50) );
```

5. Proses pencarian titik tengah yaitu berupa koordinat x dan y bisa kita cari dengan fungsi kode pemrograman berikut :

```
x = full[i].x;
y= full[i].y;
```

Setelah mendapatkan koordinat x dan y, langkah selanjutnya adalah menandai daerah badan manusia yang telah terdeteksi dengan menandai disekitar area yang telah terdeteksi dengan bentuk *ellipse*. Untuk bisa menandai dengan bentuk *ellipse* kita harus menentukan titik tengah dari ellipse tersebut dengan kode pemrograman berikut :

```
Point center( full[i].x + full[i].width*0.5,
full[i].y + full[i].height*0.5 );
```

Titik tengah dari *ellipse* ini berguna sebagai acuan untuk membuat *ellipse* pada daerah yang diinginkan. Beriku fungsi kode pemrograman untuk membuat ellipse pada daerah yang diinginkan :

```
cvEllipse(CvArr* img, CvPoint center, CvSize axes, double angle, double start_angle, double end_angle, CvScalar color, int thickness=1, int line_type=8, int shift=0 )
```

Keterangan :

Img : *Frame* gambar

Center : Titik tengah dari *ellipse*

Axes : Setengah dari ukuran axis

Angle : Rotasi *ellipse* dalam hitungan derajat

StartAngle : Titik mula dari *ellipse* dalam hitungan derajat

EndAngle : Titik akhir dari *ellipse* dalam hitungan derajat

Color : Warna yang digunakan(RGB)

Thickness : Ketebalan garis *ellipse*

LineType : Tipe garis yang digunakan

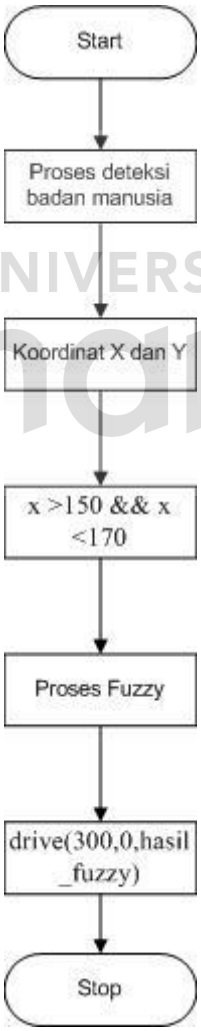
Shift : Jumlah bit fraksional dalam koordinat pusat dan nilai sumbu.

Berikut potongan program untuk membuat bentuk *ellipse* pada daerah yang terdeteksi badan manusia bagian atas :

```
cvEllipse( img1, center, Size( full[i].width*0.5, full[i].height*0.5), 0, 0, 360, Scalar( 255, 0, 255 ), 5, 8, 0 );
```

3.7 Proses *Tracking* Badan Manusia

Setelah robotino mendeteksi objek yang diinginkan yaitu badan manusia bagian atas, langkah selanjutnya adalah mengikuti pergerakan badan manusia tersebut. Robotino akan melakukan pergerakan untuk mengikuti pergerakan badan manusia sesuai dengan koordinat X . Dari koordinat X tersebut akan menentukan pergerakan dari robotino tersebut. . Berikut diagram alir dari proses deteksi badan manusia bagian atas yang digunakan pada sistem ini dapat dijelaskan pada Gambar 3.5



Gambar 3.5 Diagram alir proses penjejakan

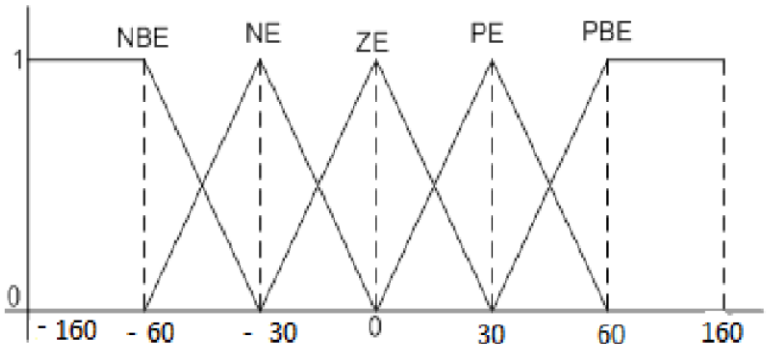
Robot akan bergerak mengikuti gerakan pergerakan manusia setelah badan manusia yang telah terdeteksi berada pada titik tengah *pixel* frame yang sebesar 320x320 dimana titik tengah *pixel* frame tersebut adalah (x>150 && x<170). Setelah berada pada titik tengah frame tersebut robot akan bergerak maju dan mengikuti arah pergerakan manusia tersebut. Pergerakan dari robot tersebut berdasarkan dari proses *fuzzy*.

3.7.1 Proses *Fuzzy*

Untuk menentukan pergerakan robotino dengan metode *fuzzy* terdapat tiga proses, yaitu:

1. Fuzzifikasi

   Pada proses fuzzifikasi ini terjadi pengambilan keputusan dengan cara memetakan *error* dan delta *error* ke dalam data *fuzzy* sesuai dengan tipe dan bentuk fungsi keanggotaan. Penentuan dari Fungsi Keanggotaan *Error* dan Delta *Error* . Fungsi keanggotan *Error* dan *Delta Error* pada maksimal adalah 160 untuk positif dan 160 negatif dikarenakan *pixel* resolusi frame yang digunakan adalah 320x320 sehingga jumlah maksimal untuk fungsi keanggotaan *Error* dan *Delta Error* adalah 320 .Penentuan dari Fungsi Keanggotaan *Error* dan Delta *Error* dapat dilihat di gambar 3.6.



Gambar 3.6 Pemetaan Derajat Keanggotaan *Error* dan *delta error*

Pada saat proses fuzzifikasi, data hasil didapatkan dari koordinat X dari proses deteksi badan manusia, selanjutnya data hasil pendeteksian tersebut akan dipetakan sesuai dengan fungsi keanggotaan *fuzzy* yang ada. Rumus yang ditentukan adalah sebagai berikut:

Error = Set Point – Present Value ..... (3.1)

Δ *Error* = *Error* Baru – *Error* Lama ..... (3.2)

Rumus (3.1) diperoleh setelah terdapat output koordinat dari proses deteksi badan manusia. Rumus (3.2) diperoleh setelah perhitungan *error* dilakukan kemudian dicari selisihnya dengan *error* yang baru.

Realisasi proses memperoleh nilai derajat keanggotaan dari masing – masing fungsi keanggotaan adalah sebagai berikut.

```
nb=fuzzyfikasi(-500,-500,-160,-80,selisih );
ns=fuzzyfikasi(-160,-80,-80,0,selisih);
z=fuzzyfikasi(-80,0,0,80,selisih);
pb=fuzzyfikasi(80,160,500,500,selisih);
ps=fuzzyfikasi(0,80,80,160,selisih);
dnb=fuzzyfikasi(-500,-500,-160,-80,dselisih);
dns=fuzzyfikasi(-160,-80,-80,0,dselisih);
dz=fuzzyfikasi(-80,0,0,80,dselisih);
dpb=fuzzyfikasi(80,160,500,500,dselisih);
dps=fuzzyfikasi(0,80,80,160,dselisih);
```

```
float fuzzyfikasi(float a,float b,float c,float d,float
x)
{
      float hasil;
      if (x<a)
            hasil=0;
      else if((x>=a)&&(x<b))
            hasil=((x-a)/(b-a));
      else if ((x>=b)&&(x<c))
            hasil=1;
      else if ((x>=c)&&(x<d))
            hasil=((d-x)/(d-c));
      else if (x>=d)
            hasil=0;
      return(hasil);

      }
```

2. *Rule Set*

Rule set / Evaluasi aturan adalah proses mengevaluasi derajat keanggotaan tiap-tiap fungsi keanggotaan himpunan *fuzzy* masukan ke dalam basis aturan yang telah ditetapkan. Tujuan dari evaluasi aturan ini adalah menentukan derajat keanggotaan dari keluaran *fuzzy*. Sebelum melakukan evaluasi aturan terlebih dahulu ditetapkan basis aturan. Basis aturan merupakan keseluruhan aturan dari kombinasi dua masukan yang mungkin. Secara lengkap, jumlah kombinasi yang mungkin dari dua himpunan *fuzzy* masukan dengan masing-masing lima fungsi keanggotaan sehingga jumlah aturannya adalah dua puluh lima aturan. Basis aturan yang dibuat berdasarkan *error* dan delta *error*. Pada bentuk yang lebih sederhana, dua puluh lima aturan kendali *fuzzy* dapat dilihat pada tabel 3.1 berikut :

Tabel 3.1 Tabel MacVicarWhelan

	Delta Error					
Error		NB	NS	Z	PS	PB
	NB	Kic	Ki	Ki	Ka	Kas
	NS	Ki	Ki	M	Ka	Kas
	Z	Ka	Ka	M	Ki	Kis
	PS	Ka	Ka	M	Ki	Ki
	PB	Kac	Ka	M	Ki	Kis

Keterangan :

NB	: *Negative Big*	Kis	: Kiri Cepat
N	: *Negative*	Ki	: Kiri
Z	: *Zero*	M	: Maju
P	: *Positive*	Kas	: Kanan Cepat
PB	: *Positive Big*	Ka	: Kanan

3. Defuzzifikasi

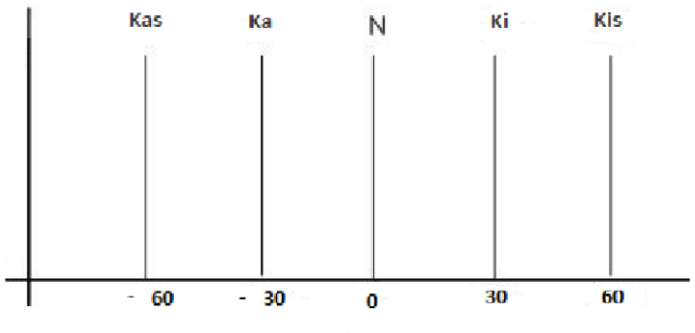
Defuzzifikasi adalah kebalikan dari proses fuzzifikasi, yaitu mengubah himpunan *fuzzy* keluaran menjadi keluaran tegas (*crisp*). Pengubahan ini diperlukan karena konstanta kendali *fuzzy* hanya mengenal nilai tegas sebagai variabel sinyal kontrol. Hasil keluaran *crisp output* akan dijadikan sebagai kendali robot. Realisasi proses pengambilan keputusan metode sugeno menjadi bentuk *crisp output* dalam bentuk rumus sebagai berikut :

$$Crisp\_out = \frac{\sum_i FuzzyOutput_i \times (PosisiSingletonXaxis_i)}{\sum_i FuzzyOutput_i} \quad \text{.....(3.3)}$$

Berikut kode untuk mendapatkan hasil keluaran *crisp output* yang akan dijadikan sebagai acuan untuk pergerakan dari robotino :

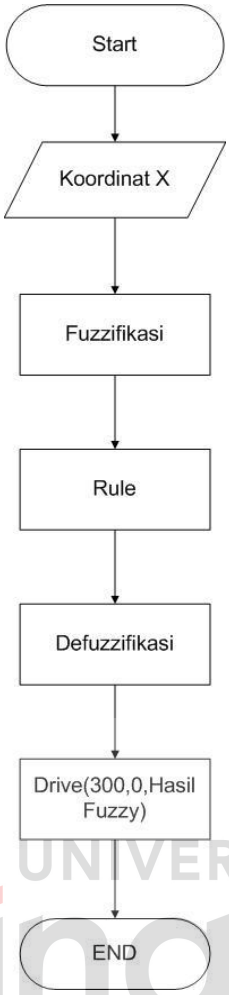
```
jtot[0]=(min[0]*jnb)+(min[1]*jnb)+(min[2]*jns)+(min[3]*jns)+(min[4
]*jz);
jtot[1]=(min[5]*jnb)+(min[6]*jnb)+(min[7]*jns)+(min[8]*jz)+(min[9]
*jps);
jtot[2]=(min[10]*jnb)+(min[11]*jns)+(min[12]*jz)+(min[13]*jps)+(mi
n[14]*jpb);
jtot[3]=(min[15]*jns)+(min[16]*jz)+(min[17]*jps)+(min[18]*jpb)+(mi
n[19]*jpb);
jtot[4]=(min[20]*jz)+(min[21]*jps)+(min[22]*jps)+(min[23]*jpb)+(mi
n[24]*jpb);
tot=0;
for (i=0;i<25;i++)
      {
            tot=tot+min[i];
      }
hasil=(jtot[0]+jtot[1]+jtot[2]+jtot[3]+jtot[4])/tot;.....
```

Di bawah ini adalah tampilan fungsi keanggotaan dari proses defuzzifikasi dimana arah pergerakan robot untuk bergerak memutar ke kanan berupa bilangan numerik negatif dikarenakan arah pergerakan memutar dari robotino untuk memutar ke kanan merupakan bilangan numerik negatif :



Gambar 3.7 Gambar fungsi keanggotaan

Berikut diagram alir dari proses *fuzzy* yang digunakan pada sistem ini dapat dijelaskan pada Gambar 3.6



Gambar 3.6 Diagram Alir Proses *fuzzy*

Hasil dari proses defuzzifikasi tersebut akan dijadikan sebagai acuan untuk pergerakan robot. Dalam pengendalian pergerakan robot memiliki paramaeter yaitu X,Y dan Z. X sebagai parameter untuk bergerak lurus dalam satuan mm/s, Y sebagai parameter untuk bergerak menyamping dalam satuan mm/s dan Z sebagai parameter untuk bergerak berputar dalam satuan degree/s. Berikut kode untuk mengatur pergerakan robotino :

```
Drive (300, 0 , Hasil_fuzzy)
```

3.8 Metode Pengujian dan Evaluasi Sistem

Untuk mengetahui apakah aplikasi yang dibuat dapat berjalan sesuai yang diharapkan, maka akan dilakukan pengujian dan evaluasi sistem untuk setiap tahapan-tahapan dalam pembuatan aplikasi. Dimulai dari pengambilan citra, *color filtering* menggunakan ruang warna RGB,BGR dan *thresholding*, koneksi Robotino, deteksi badan manusia menggunakan metode Haar-like feature dan penjejakan badan manusia bagian atas sesuai dengan pergerakan manusia,

3.8.1 Pengujian Koneksi Robotino

Pengujian koneksi dilakukan dengan cara melihat apakah Robotino dapat terintegrasi dengan PC, kemudian dilakukan pengujian dengan melihat *access point* dari Robotino apakah sudah terhubung pada Wi-Fi PC. Kemudian setelah terhubung secara *wireless*, langkah berikutnya yaitu menghubungkan koneksi dari *console programming* ke alamat IP yang telah ditentukan pada Robotino. Secara *default* alamat IP yang digunakan adalah 172.26.1.1.

3.8.2 Pengujian Pergerakan Robotino

Untuk pengujian pergerakan Robotino dilakukan dengan cara mengubah parameter pada fungsi setVelocity secara manual. Kemudian dari data tersebut didapatkan arah pergerakan Robotino. Data arah pergerakan Robotino selanjutnya digunakan untuk pergerakan badan manusia.

3.8.3 Pengujian *Streaming* Citra Melalui Kamera PC

Untuk mengetahui apakah data citra sudah dapat diakses langsung melalui kamera PC, maka dilakukan pengujian dengan cara menjalankan (*running*) program pemanggilan kamera dari Visual C++ 2008, yaitu untuk mengakses *console* kamera PC secara langsung dari program. Kemudian citra yang tampil akan diuji apakah dapat menampilkan data citra secara *streaming.*

3.8.4. Pengujian *Streaming* Citra Melalui Kamera Robotino

Untuk mengetahui apakah data citra sudah dapat diakses langsung melalui kamera Robotino, maka dilakukan pengujian dengan cara melihat data yang tampil pada PC apakah sudah sesuai dengan citra yang ditangkap oleh kamera Robotino. Kemudian citra yang tampil akan diuji apakah dapat menampilkan data citra secara *streaming.*

3.8.5 Pengujian Deteksi Badan Manusia

Untuk mengetahui apakah aplikasi dapat mendeteksi badan manusia bagian atas**.** Pengujian ini akan dilakukan berdasarkan jarak manusia dengan kamera ,intensitas cahaya dan banyaknya badan manusia yang tertangkap oleh kamera. Pengujian ini dinyatakan berhasil jika aplikasi sudah dapat mendeteksi badan manusia bagian atas yang tertangkap oleh kamera. Setiap badan yang tertangkap oleh kamera dan berhasil dideteksi akan ditandai dengan sebuah ellipse yang berada disekeliling badan dan aplikasi akan mengeluarkan output berupa koordinat X dan Y. Hal ini dilakukan untuk menandai area badan manusia yang

telah berhasil dideteksi dan titik koordinat akan dijadikan sebagai acuan untuk pergerakan robotino.

3.8.6 Pengujian Penjejakan Robotino

Tujuan Pengujian ini adalah untuk melihat efektifitas, kecepatan dan arah pergerakan robot untuk mengikuti pergerakan badan manusia sesuai dengan koordinat X dan Y yang telah terdeteksi oleh metode *Haar-like feature*. Pengujian ini dinyatakan berhasil jika robot bergerak mendekati badan manusia yang dituju.

3.8.7 Evaluasi Sistem Keseluruhan

Pengujian evaluasi sistem kesuluruhan adalah pengujian sistem secara keseluruhan dari awal hingga akhir, dimana pengujian dilakukan dengan menjalankan aplikasi secara keseluruhan. Mulai dari proses deteksi sampai proses penjejakan. Pertama-tama sistem ini akan melakukan pengolahan citra yaitu konversi warna dari RGB ke ruang warna BGR setelah itu ke ruang warna *grayscale*. Setelah proses pengolahan citra maka proses selanjutnya adalah deteksi objek badan manusia bagian atas, setelah objek badan manusia telah terdeteksi maka sistem akan mengeluarkan nilai koordinat X dan Y yang akan dijadikan sebagai acuan pergerakan robotino untuk melakukan penjejakan. Setelah mendapatkan nilai koordinat X dan Y akan diuji apakah robot bisa mengikuti pergerakan badan manusia yang telah terdeteksi tersebut dan mendekati badan manusia tersebut.

BAB IV

PENGUJIAN SISTEM

Pengujian sistem yang dilakukan merupakan pengujian terhadap Robotino dan aplikasi pada PC yang telah dibuat. Dimulai dari menghubungkan koneksi ke Robotino, menggerakan Robotino, pengambilan citra dari kamera PC dan kamera Robotino, deteksi badan manusia, melacak badan manusia hingga pergerakan robot mengikuti pergerakan badan manusia.

4.1. Pengujian Koneksi Robotino

Pengujian koneksi Robotino dilakukan dengan menguji jaringan *wireless* pada PC yang dihubungkan dengan *access point* Robotino. Hal tersebut untuk diketahui apakah dapat terhubung, dan saat program dijalankan apakah *console* sudah mampu mengakses Robotino.

4.1.1. Tujuan

Tujuan pengujian ini yaitu mengetahui apakah Robotino sudah mampu terhubung dengan komputer PC. Mengetahui apakah *console* yang digunakan sudah mampu mengakses Robotino, sehingga dapat disimpulkan apakah Robotino telah terintegrasi dengan PC.

4.1.2. Alat yang Digunakan

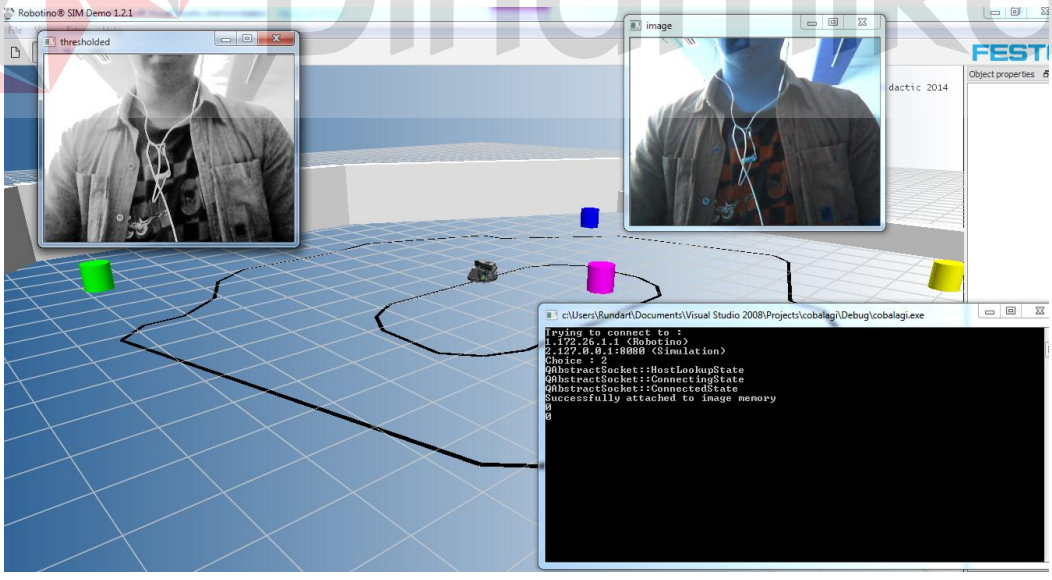
1. Robotino
2. Microsoft Visual C++ 2008
3. *Personal Computer* (PC)

4.1.3. Prosedur Pengujian

1. Menyalakan Robotino
2. Menghubungkan koneksi *wireless* dari PC ke *access point* Robotino.
3. Menjalankan program *console* pada Microsoft Visual C++ 2008
4. Menghubungkan *console* dengan Robotino sesuai IP yang ditentukan (128.0.0.1 untuk simulasi dan 172.26.1.1 untuk Robotino)

4.1.4. Hasil Pengujian

Robotino sudah dapat terhubung ke PC dengan media *wireless* dengan baik. Terlihat pada Gambar 4.1, saat program *console* dijalankan terdapat 2 pilihan koneksi yaitu langsung ke Robotino atau simulasi. Pada simulasi seluruh fungsi Robotino dapat digunakan seperti Robotino yang sesungguhnya, sehingga jika dihubungkan koneksi pada simulasi dan berhasil, maka sudah dapat dipastikan koneksi yang dihubungkan ke Robotino langsung berhasil.



Gambar 4.1 *Console* Terhubung Dengan Simulasi Robotino

Untuk pengujian ke Robotino secara langsung dapat dilihat pada LCD yang ditampilkan pada Robotino bahwa koneksi antara Robotino dengan PC sudah terhubung. Untuk lebih jelasnya terlihat pada Gambar 4.2.

Gambar 4.2 *Console* Terhubung Dengan Robotino

Dari hasil pengujian diatas Robotino sudah mampu terhubung dengan komputer PC, dan *console* yang digunakan sudah mampu mengakses Robotino, sehingga dapat disimpulkan Robotino telah terintegrasi dengan PC.

4.2. Pengujian Pergerakan Robotino

Pengujian pergerakan Robotino dilakukan dengan memanfaatkan *omni-directional drive* pada Robotino. Yaitu dengan menguji fungsi setVelocity saat diisi dengan parameter tertentu pada *console*, apakah Robotino akan bergerak sesuai dengan yang diharapkan.

4.2.1 Tujuan

Tujuan pengujian pergerakan Robotino memanfaatkan *omni-directional drive* Robotino adalah untuk mengetahui arah pergerakan Robotino dengan mengatur parameter untuk pergerakan Robotino apakah sesuai dengan yang

diinginkan. Maka hasil yang didapatkan akan menunjukan arah pergerakan Robotino.

4.2.2. Alat yang Digunakan

1. Robotino
2. Microsoft Visual C++ 2008
3. *Personal Computer* (PC)

4.2.3. Prosedur Pengujian

1. Menyalakan Robotino
2. Menghubungkan koneksi *wireless* dari PC ke *access point* Robotino
3. Menjalankan program *console* pada Microsoft Visual C++ 2008
4. Menghubungkan *console* dengan Robotino sesuai IP yang ditentukan (128.0.0.1 untuk simulasi dan 172.26.1.1 untuk Robotino)
5. Menjalankan program untuk *manual control*
6. Melihat arah pergerakan Robotino

4.2.4. Hasil Pengujian

Hasil dari pengujian ini adalah mengetahui arah pergerakan Robotino melalui parameter-parameter yang ada yaitu x,y dan omega. Untuk pergerakan Robotino dengan arah tertentu dilakukan konfigurasi parameter sesuai Tabel 4.1.

Tabel 4.1 Pergerakan Robotino

Button	**Parameter SetVelocity**			**Hasil Pergerakan Robotino**
	Vx	**Vy**	**Omega**	
w	200	0	0	Maju
s	-200	0	0	Mundur
d	0	-200	0	Kanan

a	0	200	0	Kiri
x	0	0	0	Berhenti
z	0	0	20	Berputar ke Kiri
c	0	0	-20	Berputar ke Kanan

Dengan demikian dapat disimpulkan bahwa *omni-directional drive* pada Robotino dengan parameter setVolocity yang diisi sesuai tabel dan pengujian di atas sudah dapat berjalan dengan baik dan sesuai dengan arah yang diharapkan.

4.3. Pengujian *Streaming* Citra Melalui kamera PC

Pengujian *streaming* ini dilakukan dengan mengintegrasikan Microsoft Visual C++ melalui library OpenCV. Yaitu untuk memanggil serta menjalankan *console* kamera PC tersebut.

4.3.1. Tujuan

Tujuan pengujian ini yaitu untuk mengetahui apakah aplikasi sudah mampu menampilkan data citra dari kamera PC ke aplikasi pada Visual C++ dan apakah dapat langsung diproses oleh program.

4.3.2. Alat yang Digunakan

1. Microsoft Visual C++ 2008
2. Personal Computer (PC) dengan kamera web

4.3.3. Prosedur Pengujian

1. Menjalankan program *console* pada Microsoft Visual C++ 2008
2. Menjalankan program untuk mengakses data citra pada kamera PC tersebut
3. Melihat hasil data citra pada *window*

4.3.4. Hasil Pengujian

Setelah melakukan pengujian sesuai dengan prosedur diatas berikut adalah gambar yang didapatkan dari kamera PC pada Gambar 4.3.

Gambar 4.3 Gambar *streaming* dengan kamera PC

4.4. Pengujian *Streaming* Citra Melalui kamera Robotino

Pengujian *streaming* gambar Robotino dilakukan dengan melihat data gambar yang tampil pada *window* apakah sesuai dengan yang terlihat pada kamera Robotino. Dan setelah itu melihat apakah data gambar yang dikeluarkan mampu diolah dan ditampilkan secara *streaming*.

4.4.1. Tujuan

Tujuan pengujian ini yaitu untuk mengetahui apakah aplikasi sudah mampu menampilkan data gambar dari *webcam* Robotino ke *window* pada PC, dan untuk gambar yang ditampilkan apakah sudah *streaming.*

4.4.2. Alat yang Digunakan

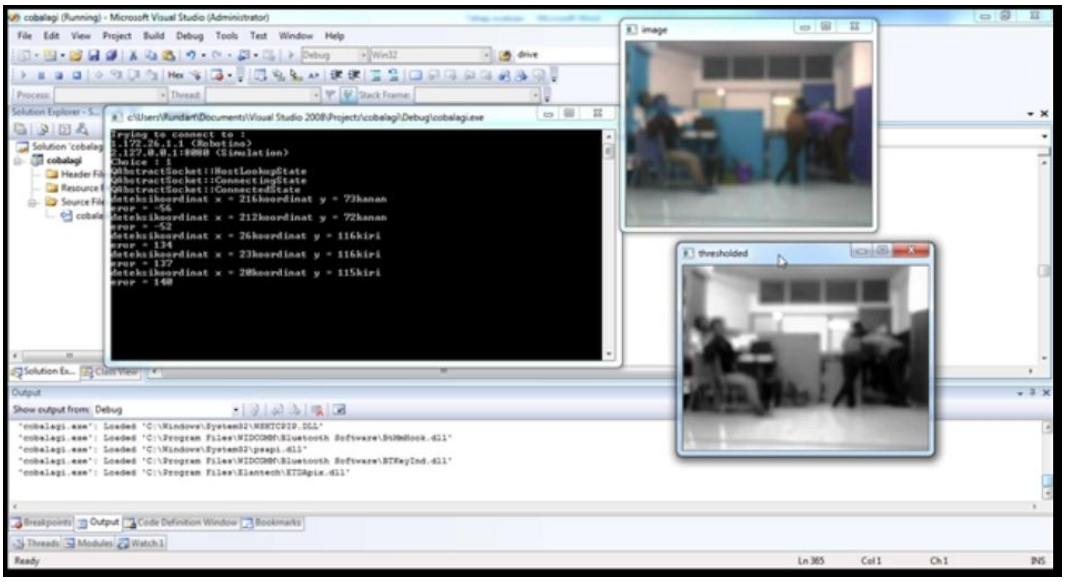
1. Robotino
2. *Webcam*
3. Microsoft Visual C++ 2008
4. Personal Computer (PC)

4.4.3. Prosedur Pengujian

1. Menyalakan Robotino
2. Menghubungkan koneksi *wireless* dari PC ke *access point* Robotino
3. Menjalankan program *console* pada Microsoft Visual C++ 2008
4. Menghubungkan *console* dengan Robotino sesuai IP yang ditentukan (128.0.0.1 untuk simulasi dan 172.26.1.1 untuk Robotino)
5. Menjalankan program untuk mengakses data gambar pada *webcam* Robotino
6. Melihat data gambar pada *window* dengan ketentuan sebagai berikut:
   a. *Streaming*: data gambar yang ditampilkan akan mengikuti pergerakan Robotino.
   b. Tidak *Streaming*: data gambar yang ditampilkan tidak mampu menampilkan saat Robotino bergerak.

4.4.4. Hasil Pengujian

Setelah melakukan pengujian sesuai dengan prosedur di atas berikut adalah gambar yang didapatkan dari *webcam* Robotino pada Gambar 4.4.



Gambar 4.4 Gambar *streaming* kamera Robotino

Dengan melihat hasil pengujian di atas, dapat disimpulkan bahwa aplikasi sudah mampu mendapatkan data citra pada *webcam* Robotino dan mampu menampilkannya secara *streaming* kepada PC.

4.5. Pengujian *Pre-Processing*

Pengujian berikut adalah pengujian data citra pada hasil pengolahan citra, dengan melihat tampilan data citra yang sudah dikonversikan ke ruang warna BGR. Kemudian dilakukan *color filtering* untuk mendeteksi badan manusia bagian atas menggunakan *thresholding*.

4.5.1. Tujuan

Pengujian ini bertujuan untuk menguji proses *color filtering* pada citra dengan cara melihat data citra yang sudah di konversikan ke ruang warna BGR, kemudian dari data gambar hasil konversi ruang warna BGR ke ruang warn*a Grayscale* dilihat apakah citra hasil proses *color filtering* sudah mampu menyaring warna sesuai filter yang ditentukan.

4.5.2. Alat yang Digunakan

1. *Webcam*
2. Microsoft Visual C++ 2008
3. *Personal Computer* (PC)
4. Objek Badan Manusia

4.5.3. Prosedur Pengujian

1. Menjalankan program *console* pada Microsoft Visual C++ 2008
2. Menjalankan program untuk mengakses *webcam*
3. Menjalankan program untuk proses *color filtering*
4. Melihat citra pada *window* dengan tampilan cita pada ruang warna BGR
5. Melihat citra pada *window* hasil proses *color filtering* dalam citra biner (*thresholding*)

4.5.4. Hasil Pengujian

Untuk proses *pre-processing* diperlukan konversi ruang warna RGB ke HSV digunakan baris perintah berikut:

```
cvCvtColor(img1,thresholded, CV_BGR2GRAY);
```

Citra hasil konversi dari baris perintah di atas dapat dilihat seperti pada Gambar 4.5.

Gambar 4.5 Hasil konversi ruang warna BGR ke *Thresholded*

Dengan melihat hasil seperti pada Gambar 4.5 dapat disimpulkan bahwa pengujian *color filtering* sudah berjalan sesuai ketentuan. Dari citra hitam putih yang ditampilkan, maka dapat disimpulkan aplikasi yang dibuat sudah mampu mendeteksi warna sesuai filter yang telah ditentukan sebelumnya.

4.6. Pengujian Deteksi Badan Manusia

Pengujian deteksi badan manusia bagian atas metode *Haar-like feature* dilakukan dengan melakukan deteksi badan manusia bagian atas dengan beberapa kondisi tertentu . Untuk mengetahui lokasi bagian objek badan manusia yang terdeteksi akan keluar output berupa koordinat X dan Y pada objek.

4.6.1. Tujuan

Tujuan Pengujian deteksi badan manusia bagian metode *Haar-like feature* ini untuk menguji metode ini untuk mendeteksi badan manusia yang tertangkap oleh kamera. Dengan berbagai model pengujian yang dilakukan

bertujuan untuk menguji efektivitas metode *Haar-like feature* dalam mendeteksi badan manusia bagian atas.

4.6.2. Alat yang Digunakan

1. Microsoft Visual C++ 2008
2. *Personal Computer* (PC)
3. Objek badan manusia
4. Robotino

4.6.3. Prosedur Pengujian

1. Menjalankan program *console* pada Microsoft Visual C++ 2008
2. Menjalankan program deteksi badan manusia
3. Koneksi ke robotino
4. Melakukan deteksi badan manusia dengan objek target badan manusia dan jarak deteksi yang berbeda-beda

4.6.4. Hasil Pengujian

Hasil dari pengujian ini adalah untuk mengetahui jarak maksimal dan minimal metode ini untuk melakukan deteksi badan manusia dan efektivitas metode ini untuk melakukan deteksi badan manusia terhadap kondisi pencahayaan yang berbeda-beda. Pengujian pertama akan dilakukan dengan cara melacak sebanyak 1 objek badan manusia dalam berbagai jarak, pengujian akan dilakukan dengan cara mengambil 25 sampel koordinat yang pertama yang terdeteksi. Hasil pengujian akan ditampilkan dalam Tabel 4.2 :

Tabel 4.2 Hasil pendeteksian badan manusia dalam berbagai jarak (1 objek badan manusia)

Percobaan	Jarak kamera dengan objek (Meter)	Jumlah Hasil Deteksi Objek	
		Terdeteksi	Tidak terdeteksi
1	0,5	0	25
2	1	25	0
3	2	25	0
4	2,5	25	0
5	3	25	0
6	3,5	25	0
7	4	25	0
8	4,5	22	3
9	5	23	2
10	5,5	19	6
11	6	14	11
12	6,5	0	25

Pengujian Kedua akan dilakukan dengan cara melacak sebanyak 2 objek badan manusia dalam berbagai jarak, pengujian akan dilakukan dengan cara mengambil 25 sampel koordinat yang pertama yang terdeteksi. Hasil pengujian akan ditampilkan dalam Tabel 4.3 :

Tabel 4.3 Hasil pendeteksian badan manusia dalam berbagai jarak (2 objek badan manusia)

Percobaan	Jarak kamera dengan objek (Meter)	Jumlah Hasil Deteksi Objek		
		Objek ke-1	Objek ke -2	Objek lain
1	2,5/ 3	21	4	0
2	3/3,5	20	1	0
3	4/4,5	8	10	7
4	4,5/5	23	2	0
5	5/5,5	21	0	4
6	5,5/6	25	0	0
7	6/6,5	25	0	0

Pengujian Ketiga akan dilakukan dengan cara melacak sebanyak 3 objek badan manusia dalam berbagai jarak, pengujian akan dilakukan dengan cara mengambil 25 sampel koordinat yang pertama yang terdeteksi. Hasil pengujian akan ditampilkan dalam Tabel 4.4 :

Tabel 4.4 Hasil pendeteksian badan manusia dalam berbagai jarak (3 objek badan manusia)

Percobaan	Jarak kamera dengan objek (Meter)	Jumlah Hasil Deteksi Objek			
		Objek ke-1	Objek ke - 2	Objek ke - 3	Objek Lain
1	2,5/ 3/3.5	2	22	1	0
2	3/3,5/4	2	21	2	0
3	3,5/4/4,5	14	8	3	0
4	4/4,5/5	6	18	1	0
5	4,55/5,5	0	25	0	0

Pada Tabel 4.6 adalah pengujian proses *recognition* dalam berbagai kondisi pencahayaan yang diukur melalui intensitas cahayanya. Untuk mengukur intensitas cahaya digunakan alat pengukur intensitas cahaya (*lux* meter/*light* meter). Pada *lux* meter akan diatur untuk menggunakan mode Ev. Dimana hasil pembacaan sensor akan dikonversi menjadi satuan intensitas cahaya (Lux) sesuai dengan Tabel 4.5.

Tabel 4.5 Tabel konversi nilai Ev ke *Lux* (Sekonic)

EV	Lux	EV	Lux	EV	Lux	EV	Lux
0.0	2.5	5.0	80	10.0	2600	15.0	82000
0.5	3.5	5.5	110	10.5	3600	15.5	120000
1.0	5.0	6.0	160	11.0	5100	16.0	160000
1.5	7.1	6.5	230	11.5	7200	16.5	230000
2.0	10	7.0	320	12.0	10000	17.0	330000
2.5	14	7.5	450	12.5	14000	17.5	460000
3.0	20	8.0	640	13.0	20000	18.0	660000
3.5	28	8.5	910	13.5	29000	18.5	930000
4.0	40	9.0	1300	14.0	41000	19.0	1300000
4.5	57	9.5	1800	14.5	58000	19.5	1900000

Tabel 4.6 Hasil pendeteksian badan manusia bagian atas dalam beberapa kondisi pencahayaan

Kondisi Pencahayaan		Hasil Deteksi Objek	
Ev	Lux	Objek badan manusia	Objek benda lain
6	160	30	0
8	640	26	4
10	2600	27	3
12	10000	28	2

Pada Tabel 4.6 dapat dilihat perbedaan intensitas cahaya yang diukur menggunakan *lux* meter pada proses deteksi tidak banyak mempengaruhi pada hasil proses deteksi badan.

Dengan melihat hasil pada Tabel 4.5 sampai dengan Tabel 4.6, dapat disimpulkan apakah metode *Haar like-feature* sudah dapat melakukan proses pengenalan dengan baik, dengan perbedaan kondisi pencahayaan optimal dari proses deteksi antara 160-10000 lux.

4.7. Pengujian Penjejakan Badan Manusia

Pengujian penjejakan badan manusia bagian atas menggunakan metode *Haar-like feature* dan logika *fuzzy* dilakukan dengan melakukan penjejakan badan manusia bagian atas dengan jarak antara manusia dengan robot yaitu 3 meter dan kondisi manusia menghadap robot dan menghadap berlawanan dengan robot. Untuk mengetahui lokasi bagian objek badan manusia yang terdeteksi akan keluar *output* berupa koordinat X dan Y pada objek. Lalu koordinat X akan dikelola dengan proses logika *fuzzy*, dimana hasil *fuzzy* tersebut akan dijadikan sebagai logika pergerakan rotasi robot untuk menjejaki badan manusia bagian atas untuk mendekati objek tersebut.

4.7.1. Tujuan

Tujuan Pengujian penjejakan badan manusia bagian atas menggunakan metode *Haar-like feature* dan logika *fuzzy* ini untuk menguji apakah aplikasi yang dibuat bisa untuk menjejaki badan manusia bagian atas beserta pergerakannya. Dengan berbagai model pengujian yang dilakukan bertujuan untuk menguji efektivitas metode *Haar-like feature* dalam mendeteksi badan manusia bagian atas.

4.7.2. Alat yang Digunakan

1. Microsoft Visual C++ 2008

2. *Personal Computer* (PC)
3. Objek badan manusia
4. Robotino

4.7.3. Prosedur Pengujian

1. Menjalankan program *console* pada Microsoft Visual C++ 2008
2. Menjalankan program deteksi badan manusia
3. Koneksi ke robotino
4. Melakukan penjejakan badan manusia dengan kondisi badan manusia dan pergerakan manusia yang berbeda-beda

4.7.4. Hasil Pengujian

Hasil dari pengujian ini adalah untuk mengetahui efektivitas metode ini untuk melakukan penjejakan badan manusia dengan melihat pergerakan robot yaitu berupa hasil logika *fuzzy*. Pengujian pertama akan dilakukan dengan cara melacak sebanyak 1 objek badan manusia dengan kondisi jarak antara manusia dan robot yaitu 4 meter dan kondisi badan manusia membelakangi dengan robot. Pengujian ini akan dilakukan sebanyak 3 kali. Hasil dari *fuzzy* akan ditampilkan dalam bentuk tabel dan grafik. Hasil *fuzzy* tersebut akan dijadikan sebagai pergerakan robot pada koordinat Z pada robotino.

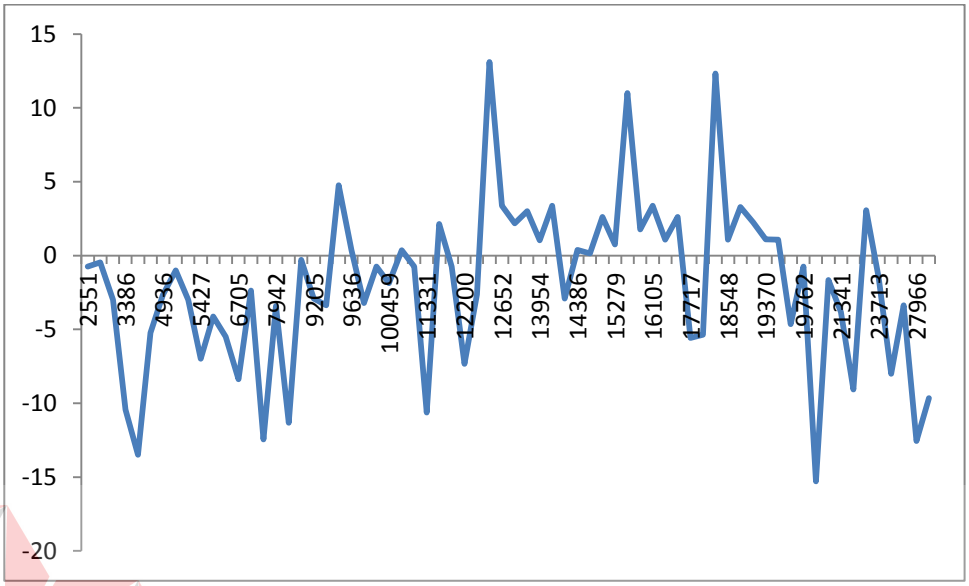
Tabel 4.7 Pergerakan Robot berdasarkan hasil pergerakan robot dengan Kondisi badan manusia membelakangi robot (Percobaan 1)

Error pergerakan (^{o})	Waktu (ms)
-0.75	2551
-0.47	2961
-3	2967
-10.43	3386
-13.48	3792
-5.23	4219
-2.66	4936
-1	5038
-3	5048
-6.976	5427
-4.125	5895
-5.48	6300
-8.372	6705
-2.38	7112
-12.44	7526
-3.4	7942
-11.33	8336
-0.3	8756
-2.92	9205
-3.375	9211
4.75	9628
0.37	9636
-3.21	100044

Error pergerakan (^{o})	Waktu (ms)
-0.75	100053
-1.875	100459
0.35	10895
-0.75	10912
-10.625	11331
2.14	11752
-0.75	11759
-7.33	12200
-2.62	12220
13.1	12645
3.37	12652
2.19	13068
3	13089
1.04	13954
3.375	13960
-2.92	14377
0.375	14386
0.15	14847
2.62	14854
0.75	15279
11	15686
1.78	16896
3.37	16105

Error pergerakan (^{o})	Waktu (ms)
1.07	16493
2.625	16504
-5.58	17717
-5.35	18144
12.33	18541
1.075	18548
3.29	18950
2.25	18960
1.09	19370
1.075	19378
-4.64	19754
-0.75	19762
-15.28	20159
-1.66	20553
-3.83	21341
-9.06	22184
3.06	23700
-1.5	23713
-8	24118
-3.375	24116
-12.55	27966

Berikut grafik dari percobaan 1 yang akan ditampilkan dalam Gambar 4.6 . Grafik tersebut merupakan arah pergerakan robot dimana koordinat Y adalah arah pergerakan robot dan koordinat 0 merupakan pergerakan lurus dari robotino :



Gambar 4.6 Grafik hasil pergerakan robot pada penjejakan 1

Keterangan :

Koordinat X : Waktu penjejakan (ms)

Koordinat Y : Arah pergerakan robot pada koordinat Z pada robotino (deg/s)

Tabel 4.8 Pergerakan Robot berdasarkan hasil pergerakan robot dengan Kondisi badan manusia membelakangi robot (Percobaan 2)

Error pergerakan (^{o})	Waktu (ms)
7.8	1759
-3.5	2210
-18.2	2636
-3.37	2643
-11.8	2083
-6	3507

Error pergerakan (^{o})	Waktu (ms)
-4.2	3901
-0.75	4351
8.47	4771
2.25	4783
-10.75	5594
-1.25	6391

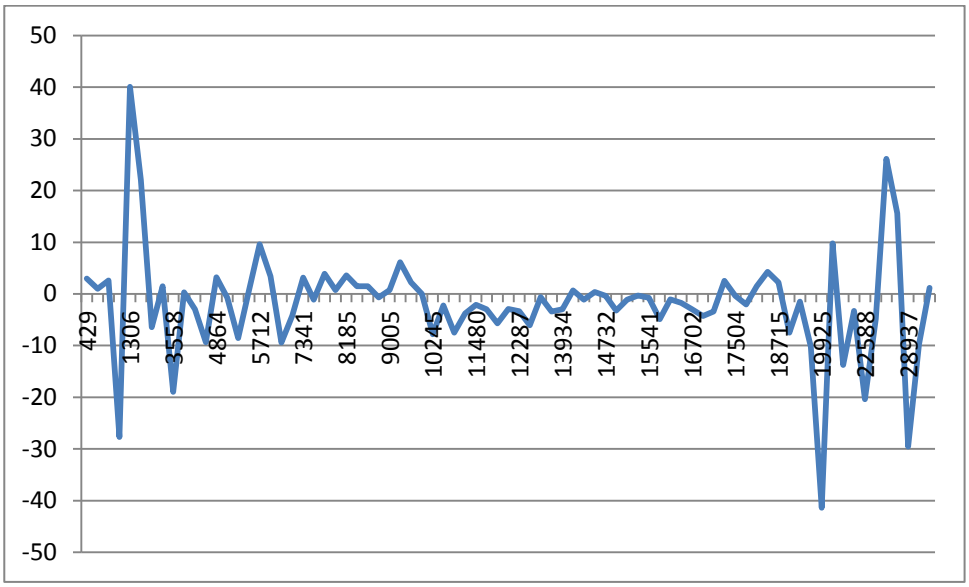
Error pergerakan (^{o})	Waktu (ms)
57.14	6403
-60	6807
5.62	7200
0	7213
2.62	8065
21.46	8868

Error pergerakan (^{o})	Waktu (ms)
`10.07	9269
-1.17	9675
-9.2	10162
-0.75	10169
-2.62	10611
-4.2	11024
-3	11031
-43.84	11437
-25.5	11867
4.68	12639
2.5	13031
-3	13038
-3	13461
-3	13469
2.19	13857
-0.375	13864
5.79	14262
1.5	14269

Error pergerakan (^{o})	Waktu (ms)
1.5	14667
1.5	14675
2.25	15083
-3.65	15502
-0.375	15510
12.3	15910
2.56	16317
3.375	16323
7.32	16728
-1.19	1711
2.25	17119
9.78	17552
-2.72	17948
1.5	17955
6.18	18364
3	18371
-4.3	18777
-0.3	18787

Error pergerakan (^{o})	Waktu (ms)
-9.25	19196
-3	19202
-13.46	19588
-17.94	19963
-20.4	20331
1.06	20734
-3.06	21104
-6.21	21513
-5.25	21892
20.82	28314
16.8	28700
2.68	29096

Berikut grafik dari percobaan 2 yang akan ditampilkan dalam Gambar 4.7. Grafik tersebut merupakan arah pergerakan robot dimana koordinat Y adalah arah pergerakan robot dan koordinat 0 merupakan pergerakan lurus dari robotino :



Gambar 4.7 Grafik hasil pergerakan robot pada penjejakan 2

Keterangan :

Koordinat X : Waktu penjejakan (ms)

Koordinat Y : Arah pergerakan robot pada koordinat Z pada robotino(deg/s)

Tabel 4.9 Pergerakan Robot berdasarkan hasil pergerakan robot dengan Kondisi badan manusia membelakangi robot (Percobaan 3)

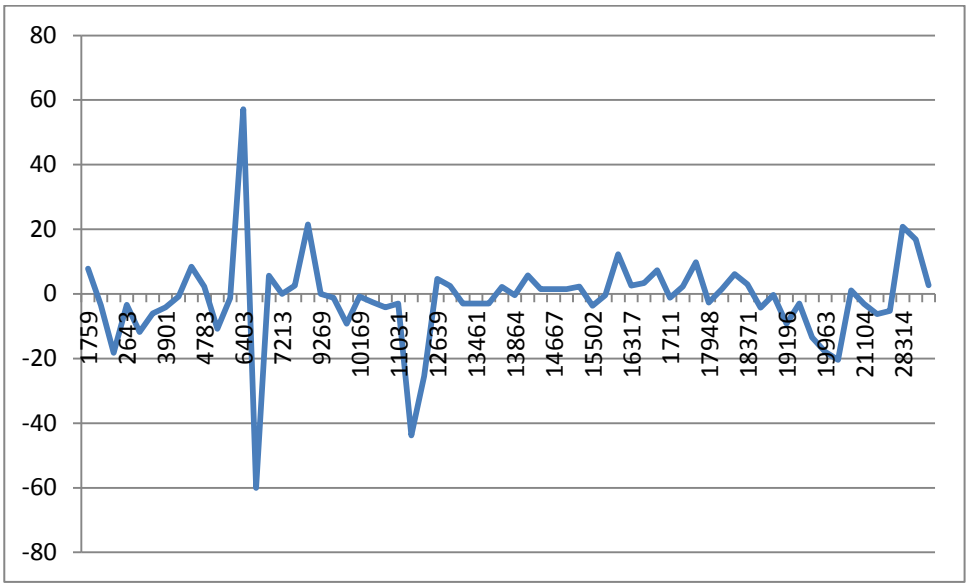
Error pergerakan (^{o})	Waktu (ms)	Error pergerakan (^{o})	Waktu (ms)	Error pergerakan (^{o})	Waktu (ms)
3	429	-9.33	4445	-1.12	7351
1.02	846	3.21	4864	3.92	7767
2.62	853	-0.75	4872	0.75	7776
-27.67	1293	-8.57143	5287	3.57	8185
40.07	1306	0.75	5293	1.5	8191
22.07	1744	9.57	5712	1.5	8689
-6.42	3124	3.375	5719	-0.71	8616
1.5	3128	-9.41	6123	0.75	9005
-18.94	3558	-4.18	6536	6.1	9426
0.31	3995	3.13	7341	2.25	9824

Error pergerakan (^{o})	Waktu (ms)
0	9830
-7.82	10245
-2.25	10455
-7.5	10673
-3.75	11069
-2.1	11480
-3	11487
-5.7	11888
-2.92	12277
-3.37	12287
-6.07	12700
-0.6	13497
-3.375	13505
-3	13934
0.69	14235
-1.12	14332
0.36	14726
-0.37	14732
-3.21	15145
-1.12	15152

Error pergerakan (^{o})	Waktu (ms)
-0.3	15532
-0.75	15541
-4.88	15925
-1.07	15933
-1.74	16693
-3	16702
-4.3	17112
-3.375	17119
2.56	17497
-0.375	17504
-2.04	17892
1.5	18311
4.28	18710
2.25	18715
-7.5	19114
-1.5	19120
-10.21	19520
-41.388	19925
9.81	19936
-13.72	20334
-3.26	20728

Error pergerakan (^{o})	Waktu (ms)
-20.4	22588
-4.83	23534
26.11	24120
15.625	28546
-29.54	28937
-9.75	29210
1.17	30086

Berikut grafik dari percobaan 2 yang akan ditampilkan dalam Gambar 4.8. Grafik tersebut merupakan arah pergerakan robot dimana koordinat Y adalah arah pergerakan robot dan koordinat 0 merupakan pergerakan lurus dari robotino :



Gambar 4.8 Grafik hasil pergerakan robotino pada penjejakan 3

Keterangan :

Koordinat X : Waktu penjejakan (ms)

Koordinat Y : Arah pergerakan robot pada koordinat Z pada robotino (deg/s)

Pada Tabel 4.5 – Tabel 4.7 telah ditampilkan hasil pergerakan robot dari 3 percobaan penjejakan, berikut nilai prosentase hasil pergerakan robot dari setiap percobaan penjejakan :

Prosentase Error Percobaan 1 = (Total error pergerakan / Total percobaan) * 100%

= (294.778 / 68)* 100%

= 4.34 %

Prosentase Error Percobaan 2=(Total error pergerakan / Total percobaan) * 100%

= (527.475/ 66)* 100%

= 7.99 %

Prosentase Error Percobaan 3= (Total error pergerakan / Total percobaan) * 100%

= (486.6244/ 79) * 100%

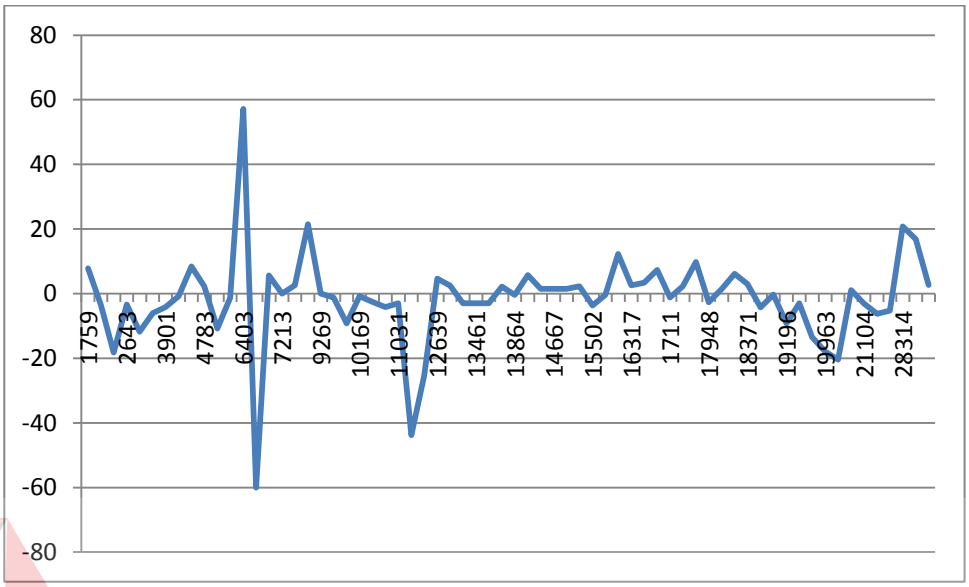
= 6.15 %

Pengujian kedua akan dilakukan dengan cara melacak sebanyak 1 objek badan manusia dengan kondisi jarak antara manusia dan robot yaitu 4 meter dan kondisi badan manusia berhadapan dengan robot. Pengujian ini akan dilakukan sebanyak 3 kali. Hasil dari *fuzzy* akan ditampilkan dalam bentuk tabel dan grafik. Hasil *fuzzy* tersebut akan dijadikan sebagai pergerakan robot pada koordinat Z pada robotino.

Tabel 4.10 Pergerakan Robot berdasarkan hasil Logika *fuzzy* dengan Kondisi badan manusia menghadap robot (Percobaan 1)

Error pergerakan (^{o})	Waktu (ms)	Error pergerakan (^{o})	Waktu (ms)	Error pergerakan (^{o})	Waktu (ms)
-25.21	0	-9.06	7121	6.81	14264
-12.07	392	-2.33	7525	0.34	14783
-13.57	835	2.85	7915	-1.07	15482
-2.3	1279	2.19	8318	11.63	15880
0.61	1752	0.375	8725	-3.65	16281
-2.56	2187	-0.375	9122	-15	17085
-5.71	2605	-13.84	9525	-2.44	17476
-17.54	3018	-2.85	9920	-2.14	17869
-4.3	3410	-11.08	10298	-13.46	19428
-11.59	3872	-4.2	10698	18.75	42155
-4.77	4275	-1.33	11098	27.82	42555
-13.69	4674	4.75	11494	23.57	42935
-5.45	5076	-18.87	12886	1.36	43393
-5.71	5509	-4.18	12698	11.78	44496
-4.18	5931	-5.62	13086	24.33	44889
-10.56	6324	6.951	13466	18.65	45322

Berikut grafik dari percobaan 1 yang akan ditampilkan dalam Gambar 4.9 . Grafik tersebut merupakan arah pergerakan robot dimana koordinat Y adalah arah pergerakan robot dan koordinat 0 merupakan pergerakan lurus dari robotino:



Gambar 4.9 Grafik hasil pergerakan robot pada penjejakan 4

Keterangan :

Koordinat X : Waktu penjejakan (ms)

Koordinat Y : Arah pergerakan robot pada koordinat Z pada robotino(deg/s)

Tabel 4.11 Pergerakan Robot berdasarkan hasil Logika *fuzzy* dengan Kondisi badan manusia menghadap robot (Percobaan 2)

Error pergerakan (^{o})	Waktu (ms)
-26.19	0
-11.625	8
-12.8	405
-5.74	1381
-5.79	4099
-6.95	19294

Error pergerakan (^{o})	Waktu (ms)
-8.78	19307
-9.14	20461
-3.21	21654
-19.36	23580
-12.14	23970
-8.57	24788

Error pergerakan (^{o})	Waktu (ms)
4.56	25179
-3.65	26360
1.07	26786
-19.13	27607
6.78	29636
3.87	31990

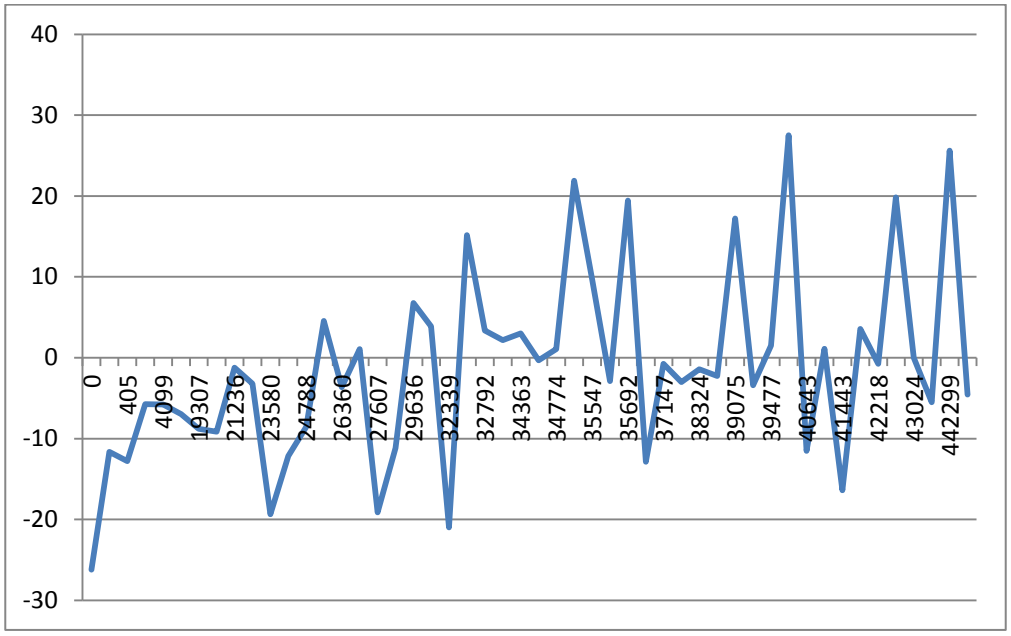
Error pergerakan (^{o})	Waktu (ms)
-21	32339
15.16	32876
3.37	32792
2.19	34355
3	34363
-0.3	34763
21.88	35178
9.75	35547
-2.88	35562
19.44	35692
-12.85	37138
-0.75	37147
-3	37553

Error pergerakan (^{o})	Waktu (ms)
-1.42	38324
-2.25	38331
17.22	39075
-3.4	39740
27.5	39875
-11.51	40643
1.12	40660
-16.38	41443
3.57	42212
-0.75	42218
19.83	42639
0	43024
-5.5	43411

Error pergerakan (^{o})	Waktu (ms)
25.62	442299

Berikut grafik dari percobaan 2 yang akan ditampilkan dalam Gambar 4.10 .

Grafik tersebut merupakan arah pergerakan robot dimana koordinat Y adalah arah pergerakan robot dan koordinat 0 merupakan pergerakan lurus dari robotino:



Gambar 4.10 Grafik hasil pergerakan robot pada penjejakan 6

Keterangan :

Koordinat X : Waktu penjejakan (ms)

Koordinat Y : Arah pergerakan robot pada koordinat Z pada robotino(deg/s)

Tabel 4.12 Pergerakan Robot berdasarkan hasil Logika *fuzzy* dengan Kondisi badan manusia menghadap robot (Percobaan 3)

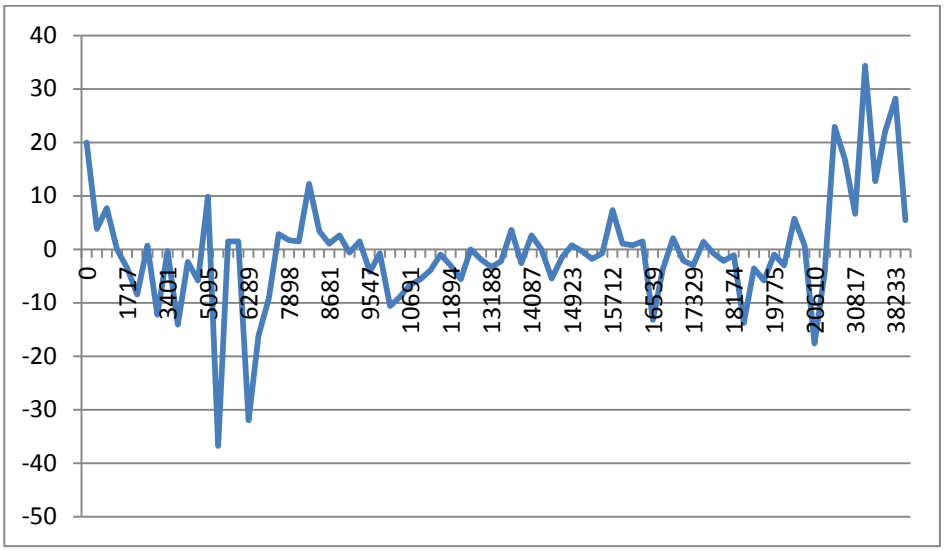
Error pergerakan (^{o})	Waktu (ms)	Error pergerakan (^{o})	Waktu (ms)	Error pergerakan (^{o})	Waktu (ms)
20	0	-14.1	3843	-9.25	7082
3.83	403	-2.33	4256	2.83	7486
7.68	814	-5.85	4669	1.7	7898
0	1224	9.88	5095	1.5	7905
-3.57	1717	-36.78	5501	3.37	8306
-8.47	2168	1.5	5509	1.07	8681
0.71	2591	1.5	5904	2.62	8687
-12.24	2983	-32	6289	-0.6	9048
-0.32	3401	-16.12	6681	1.5	9091

Error pergerakan (^{o})	Waktu (ms)
-4.2	9547
-0.75	9555
-10.62	9948
-8.86	10308
-6.58	10691
-5.62	11095
-3.92	11487
-1	11886
-3	11894
-5.61	12300
0	12732
-1.875	12742
-3.29	13188
-2.25	13195
3.65	13651
-2.625	13669
2.625	14087
-5.45	14496
-1.5	14583
0.73	14923

Error pergerakan (^{o})	Waktu (ms)
-0.375	14930
-1.82	15319
-0.75	15326
7.33	15712
1.07	15719
0.73	16129
1.5	16137
-13.2	16539
-3.75	16930
2.1	17322
-2.14	17322
-3	17329
1.42	17448
-0.75	17756
-2.19	18167
-1.12	18174
-13.82	18588
-3.57	19085
-5.85	19834
-3	19782

Error pergerakan (^{o})	Waktu (ms)
5.71	20193
0.75	20287
-17.65	20610
-4.28	29200
22.88	30004
6.6	30817
34.322	33842
12.76	34261
22.14	34564
28.16	38233

Berikut grafik dari percobaan 3 yang akan ditampilkan dalam Gambar 4. . Grafik tersebut merupakan arah pergerakan robot dimana koordinat Y adalah arah pergerakan robot dan koordinat 0 merupakan pergerakan lurus dari robotino :



Gambar 4.11 Grafik hasil pergerakan robot pada penjejakan

Keterangan :

Koordinat X : Waktu penjejakan (ms)

Koordinat Y : Arah pergerakan robot pada koordinat Z pada robotino(deg/s)

Pada Tabel 4.10 – Tabel 4.12 telah ditampilkan hasil pergerakan robot dari 3 percobaan penjejakan, berikut nilai prosentase hasil pergerakan robot dari setiap percobaan penjejakan :

Prosentase Error Percobaan 1= (Total error pergerakan / Total percobaan) * 100%

= (421.001 / 50)* 100%

= 8.42 %

Prosentase Error Percobaan 2= (Total error pergerakan / Total percobaan) * 100%

= (440.505/ 50)* 100%

= 8.81%

Prosentase Error Percobaan 3 = (Total error pergerakan / Total percobaan) * 100%

= (532.732/ 8200) * 100%

= 6.64%

Pengujian ketiga akan dilakukan dengan cara melacak sebanyak 1 objek badan manusia dengan kondisi jarak antara manusia dan robot yaitu 4 meter dan kondisi badan manusia membelakangi dengan robot lalu manusia akan bergerak menjauhi robot. Pengujian ini akan dilakukan sebanyak 1 kali. Hasil dari *fuzzy* akan ditampilkan dalam bentuk tabel dan grafik beserta nilai prosentase. Hasil *fuzzy* tersebut akan dijadikan sebagai pergerakan robot pada koordinat Z pada robotino

Tabel 4.13 Pergerakan Robot berdasarkan hasil Logika *fuzzy* dengan Kondisi badan manusia bergerak menjauhi robot

Error pergerakan (^{o})	Waktu (ms)
10.42	0
-50.47	11
14.6	863
-40.21	872
-20.2	974
40.15	1321
-3.42	1783
-50.76	1996
37.66	2242
-5.13	2677
-51.27	2843
38.18	3124
3.4	3546
46.6	3989
4.75	4418
-50.2	4851

Error pergerakan (^{o})	Waktu (ms)
45.25	4860
-3.04	5277
5.2	5688
-50.63	5699
27.07	6120
-0.375	6126
4.53	6531
1.12	6538
-54.3	6989
14.28	7805
-41.42	8235
41.81	8245
-3.52	850
-55.71	8660
41.25	9864

Error pergerakan (^{o})	Waktu (ms)
-3.52	8650
-55.71	8660
41.25	9864
-16.36	9476
-1.5	9648
29.21	9880
-0.75	10303
3	10315
4.39	10712
3.37	10717
-10.76	11121
-2.25	11127
-12.85	11518
-11.66	11926
-6.56	12333

Error pergerakan (^{o})	Waktu (ms)
-6.21	12736
4.18	13145
-1.12	13152
-1.5	13564
-6.81	13978
-3	13986
-6.97	14405
-11.41	14809
-12.33	15201
-12.95	15619
-14.31	16407
-7.66	16815
1.76	17227
1.02	17427
-2.62	17856
-5.35	18068
-3.37	18075
-7.32	18471
3.65	18872
-0.375	18879
-29.02	25539
-6.12	25371
1.44	26147
24.76	26556
16.27	26945
0	27362

Error pergerakan (^{o})	Waktu (ms)
-18	27755
-17.54	28158
-0.9	28541
-14.06	28934
2.23	29319
-2.62	29934
-18.75	30897
-30.78	30476
5.78	30875
14.36	31291
2.62	31294
-7.14	32180
0.75	32107
-26.14	32497
-23.2	33951
17.19	34349
0.375	34352
15	34740
24.04	35164
11.7	35605
-5.56	36046
-24.67	36436
-33.87	36815
18.13	37998
-1.12	38004
18.68	38408

Error pergerakan (^{o})	Waktu (ms)
15.31	38827
11.78	39250
-2.59	39467
-7.41	40037
-1.12	40043
-21.31	40806
-34.21	41200
-25.35	41594
-17.28	41995
-3.09	42004
8.67	42436
-1.07	42440
25.29	42829
-60	43237
40.83	43426
9.37	43634
-5.31	44042
3	44047
-43.84	44341
-5	45558
-6	45939
-13.57	46326
-5.17	47092
0	47477
11.93	47872
1.5	47877

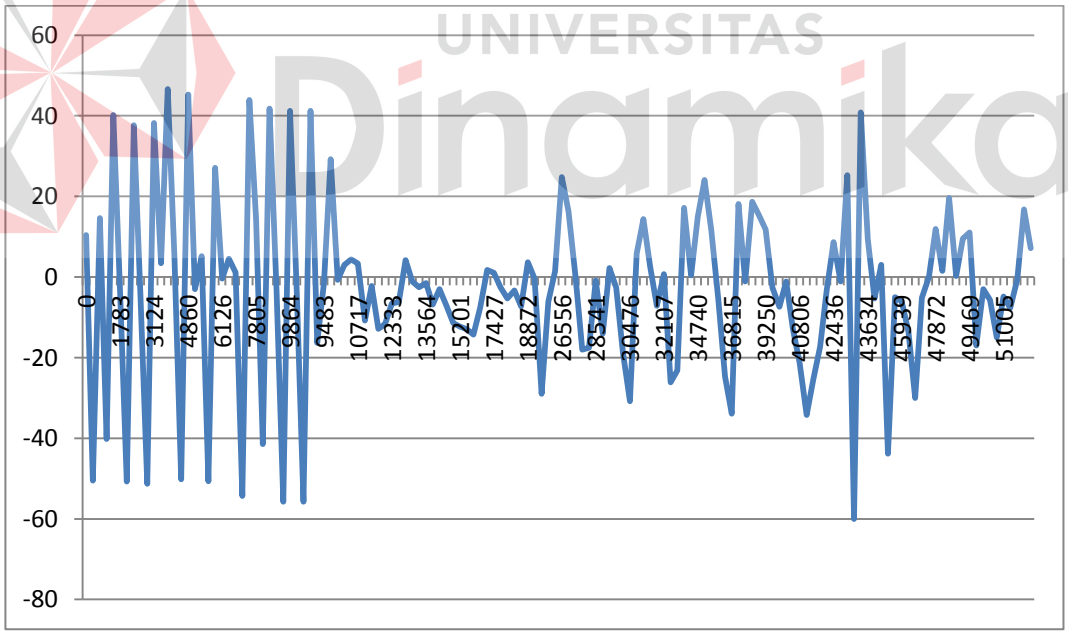
Error pergerakan (^{o})	Waktu (ms)
19.65	48283
0.25	48686
9.51	49067
11.07	49469
-17.01	49830

Error pergerakan (^{o})	Waktu (ms)
-3	49834
-5.71	50222
-15	50618
-4.88	51005
-7.68	51417

Error pergerakan (^{o})	Waktu (ms)
-0.625	52868
16.73	53242
7.14	56819

Berikut grafik dari percobaan 1 yang akan ditampilkan dalam Gambar 4.12.

Grafik tersebut merupakan arah pergerakan robot dimana koordinat Y adalah arah pergerakan robot dan koordinat 0 merupakan pergerakan lurus dari robotino:



Gambar 4.12 Grafik hasil pergerakan robot pada penjejakan 8

Keterangan :

Koordinat X : Waktu penjejakan (ms)

Koordinat Y : Arah pergerakan robot pada koordinat Z pada robotino(deg/s)

Pada Tabel 4.13 telah ditampilkan hasil pergerakan robot dari 1 percobaan penjejakan, berikut nilai prosentase hasil pergerakan robot dari 1 percobaan penjejakan :

Prosentase Error Percobaan 1 = (Total error pergerakan / Total percobaan) * 100%

= (2174.2/ 140)* 100%

= 15.53%

Pengujian keeempat akan dilakukan dengan cara melacak sebanyak 1 objek badan manusia dengan kondisi jarak antara manusia dan robot yaitu 3 meter dan kondisi badan manusia berhadapan dengan robot lalu manusia akan bergerak mendekati robot. Pengujian ini akan dilakukan sebanyak 1 kali. Hasil dari *fuzzy* akan ditampilkan dalam bentuk tabel dan grafik beserta nilai prosentase. Hasil *fuzzy* tersebut akan dijadikan sebagai pergerakan robot pada koordinat Z pada robotino.

Tabel 4.14 Pergerakan Robot berdasarkan hasil Logika *fuzzy* dengan Kondisi badan manusia bergerak mendekati robot

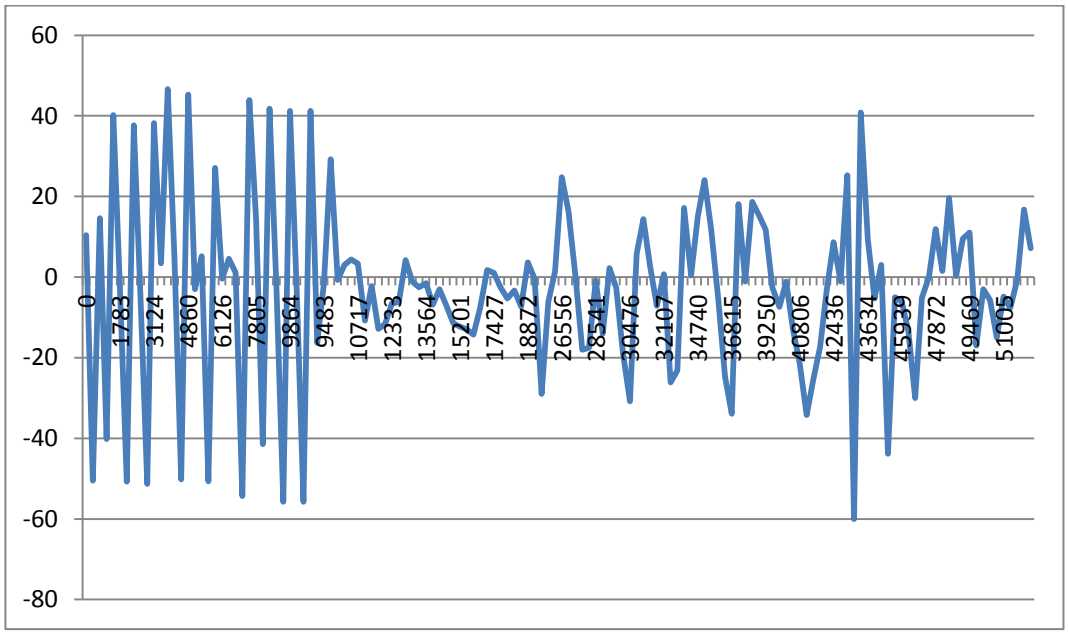
Error pergerakan (^{o})	Waktu (ms)	Error pergerakan (^{o})	Waktu (ms)	Error pergerakan (^{o})	Waktu (ms)
8.93	0	38.29	3880	-7.82	7137
4.02	417	-0.29	4296	-2.25	7144
-53.02	852	2.62	4384	9.78	7554
-26.7	12725	6.62	4740	2.25	7563
21.13	1716	-19.82	5543	-1.04	7969
-1.5	1721	-0.57	5944	1.12	7976
40.34	2608	9.28	6322	1.42	8771
27.44	3048	7.5	6727	3	8783
-50.76	3866	0	6736	4.39	9172

Error pergerakan (^{o})	Waktu (ms)
3.37	9178
-2.5	9585
0.75	9598
11.63	9993
-6.78	10397
-0.75	10404
-1.5	10833
10.21	11250
2.62	11260
12.18	11672
-5.12	12093

Error pergerakan (^{o})	Waktu (ms)
0.375	12098
5.79	12499
1.07	12508
-4.64	12959
-0.75	12966
48.83	13481
-45.71	13413
1.5	13420
48.377	13862
-46.125	13784
0	13881

Error pergerakan (^{o})	Waktu (ms)
48.51	14290
-3.57	14743
0.75	14750
48.64	15153
-46.29	15162
0.375	15186
-44.29	15596
57.39	16813
-45.43	16827
58.6	16648
-46.47	16458

Berikut grafik dari percobaan 1 yang akan ditampilkan dalam Gambar 4.13. Grafik tersebut merupakan arah pergerakan robot dimana koordinat Y adalah arah pergerakan robot dan koordinat 0 merupakan pergerakan lurus dari robotino :



Gambar 4.13 Grafik hasil pergerakan robot pada penjejakan 9

Keterangan :

Koordinat X : Waktu penjejakan (ms)

Koordinat Y : Arah pergerakan robot pada koordinat Z pada robotino(deg/s)

Pada Tabel 4.14 telah ditampilkan hasil pergerakan robot dari 1 percobaan penjejakan, berikut nilai prosentase hasil pergerakan robot dari 1 percobaan penjejakan :

Prosentase Error Percobaan 1 = (Total error pergerakan / Total percobaan) * 100%

= (1305.22/ 67)* 100%

= 19.48 %

4.8 Evaluasi Sistem secara Keseluruhan

Pengujian evaluasi sistem kesuluruhan adalah pengujian sistem secara keseluruhan dari awal hingga akhir, dimana pengujian dilakukan dengan menjalankan aplikasi secara keseluruhan. Mulai dari proses deteksi sampai proses penjejakan. Pertama-tama sistem ini akan melakukan pengolahan citra yaitu konversi warna dari RGB ke ruang warna BGR setelah itu ke ruang warna *grayscale*. Setelah proses pengolahan citra maka proses selanjutnya adalah deteksi objek badan manusia bagian atas, setelah objek badan manusia telah terdeteksi maka sistem akan mengeluarkan nilai koordinat X dan Y yang akan dijadikan sebagai acuan pergerakan robotino untuk melakukan penjejakan. Setelah mendapatkan nilai koordinat X dan Y akan diuji apakah robot bisa mengikuti pergerakan badan manusia yang telah terdeteksi tersebut dan mendekati badan manusia tersebut.

4.7.5. Tujuan

Tujuan evaluasi sistem ini adalah untuk mengetahui sistem pada aplikasi apakah sudah dapat berjalan sesuai dengan yang diharapkan. Apakah sistem dapat menemukan wajah dari orang yang dicari, sesuai dengan inputan dari *user*.

4.7.6. Alat yang Digunakan

1. Microsoft Visual C++ 2008
2. *Personal Computer* (PC)
3. Objek badan manusia
4. Robotino

4.7.7. Prosedur Pengujian

1.Menjalankan program *console* pada Microsoft Visual C++ 2008

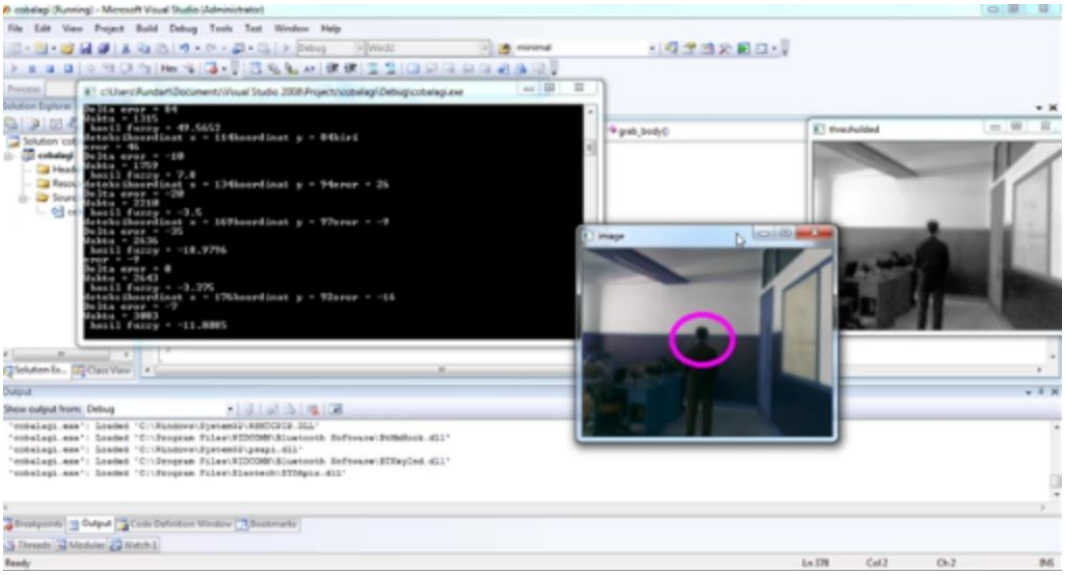
2.Menjalankan program deteksi badan manusia

3.Koneksi ke robotino

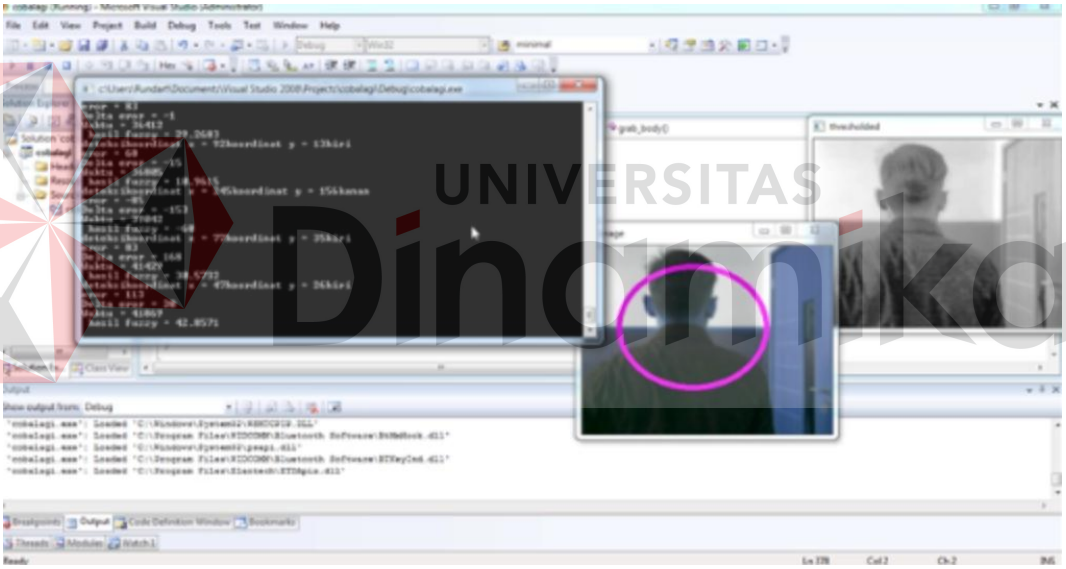
4.Melakukan deteksi dan penjejakan badan manusia

4.7.8. Hasil Pengujian

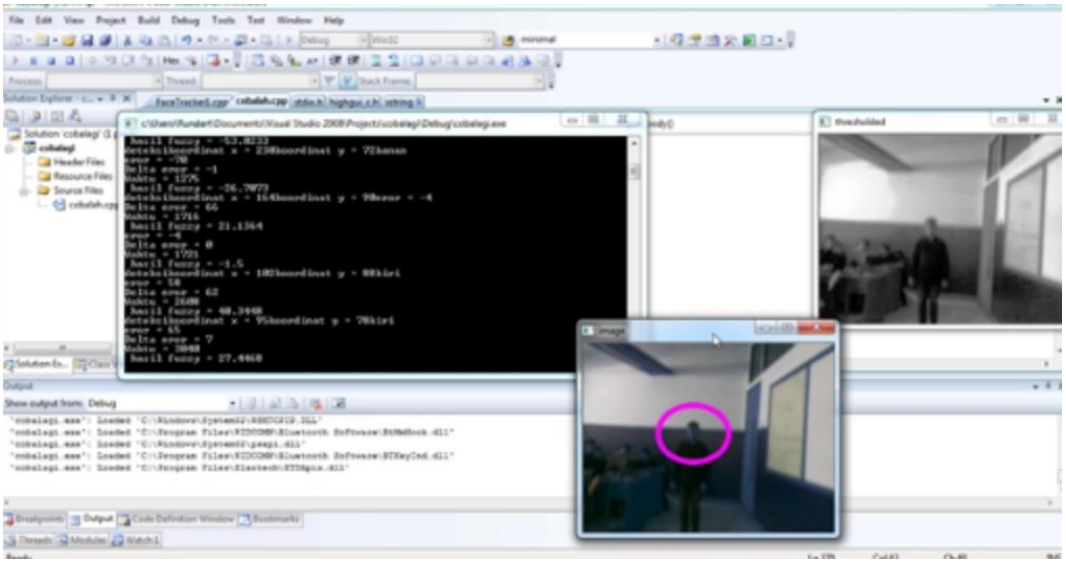
Setelah melalui seluruh prosedur pengujian diatas didapatkan hasil tingkat *error* dari proses penjejakan badan manusia bagian atas adalah kurang dari 2% . Dari semua percobaan diatas robot berhasil mendekati objek badan manusia baik yang bergerak menjauhi dan mendekati maupun objek badan manusia yang diam. Dari Gambar 4.14 sampai dengan Gambar 4.17 merupakan hasil pengujian penjejakan badan manusia bagian atas dengan berbagai kondisi



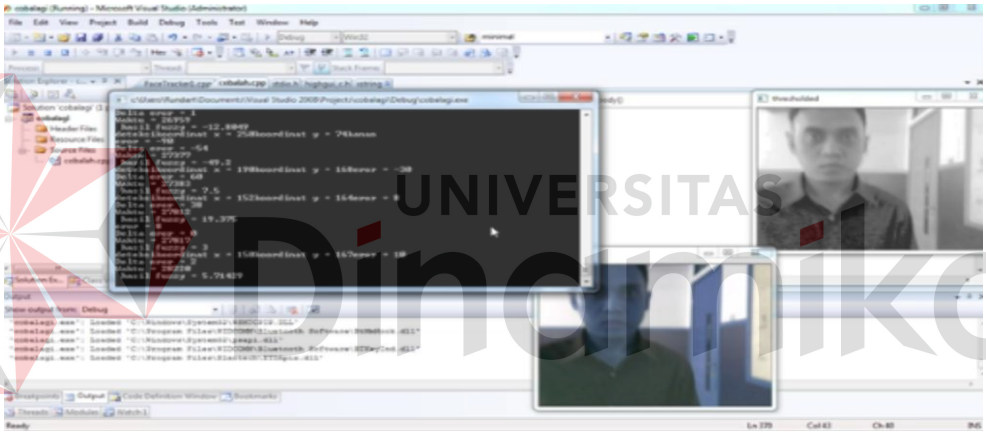
Gambar 4.14 Deteksi badan manusia bagian atas dengan kondisi badan membelakangi robot



.Gambar 4.15 Hasil proses penjejakan badan manusia dengan kondisi badan manusia menjauhi robot



Gambar 4.16 Deteksi badan manusia bagian atas dengan kondisi badan menghadap robot



.Gambar 4.17 Hasil proses penjejakan badan manusia dengan kondisi badan manusia mendekati robot

Dari hasil-hasil pengujian seperti pada Gambar 4.14 sampai dengan Gambar 4.17, Robotino akan mendeteksi badan manusia bagian atas lalu akan mendekati badan manusia. Robotino yang telah berhasil melakukan penjejakan pada badan manusia bagian atas akan menghasilkan *output* seperti pada Gambar 4.14 sampai Gambar 4.17. Dari seluruh pengujian didapatkan tingkat *error* penjejakan badan manusia bagian atas seperti yang ditampilkan dibawah ini :

Tabel 4.15 Tabel hasil seluruh pengujian *error* penjejakan badan manusia bagian atas

Pengujian	Keterangan	Percobaan ke-	Hasil Prosentase
1	Badan membelakangi robot dan tidak ada pergerakan	1	4.335
		2	7.992
		3	6.159
2	Badan menghadap robot dan tidak ada pergerakan	1	8.42
		2	8.8101
		3	6.4967
3	Pergerakan manusia menjauhi robot	1	16
4	Pergerakan manusia menjauhi robot	1	19
		Total	77.2127

Dari tabel diatas dapat dihitung akurasi rata-rata dari 8 hasil *error* yang ada menggunakan rumus :

$$\text{Rata-rata Error} = \sum \frac{\text{Nilai } error}{N} \quad \text{.....(1)}$$

Dimana : akurasi sampel adalah nilai *error* dari tiap sampel.

n adalah jumlah sampel yang digunakan.

Rata-rata *Error* = (43.35 + 79.92 + 61.59 + 84.2 + 88.101 + 64.967 + 16 + 19)/ 8 = 9.6515875 %

BAB V

PENUTUP

5.1 Kesimpulan

Berdasarkan hasil evaluasi yang telah dilakukan dalam pembuatan aplikasi ini dapat disimpulkan bahwa Tugas Akhir ini telah sesuai dengan tujuan awal. Berikut adalah beberapa poin kesimpulan dari pengerjaan tugas akhir ini:

1. Dengan memanfaatkan *webcam* yang terintegrasi pada *mobile robot omnidirectional* aplikasi penjejakan badan manusia dengan menggunakan metode *Haar like-feature* ini telah berhasil berjalan dengan baik. Jarak minimal untuk mendeteksi badan manusia bagian atas adalah 1 Meter dan jarak Maksimal untuk mendeteksi badan adalah 6 Meter. Jarak yang paling efektif untuk deteksi badan manusia adalah 1 – 4 Meter dimana prosentase keberhasilan untuk mendeteksi badan manusia adalah 100%. Namun ada beberapa kendala yang terjadi pada aplikasi ini yaitu pada saat proses deteksi badan manusia bagian atas masih ada beberapa objek lain yang menyerupai badan manusia bagian atas tetap dikenali sebagai badan manusia bagian atas. Selain itu metode *Haar like-feature* hanya menggunakan perhitungan selisih antara hasil *pixel* daerah hitam dikurangi daerah putih sehingga keaadan lingkungan akan mempengaruhi hasil deteksi
2. Metode *Haar like-feature* dan logika *fuzzy* telah berhasil diimplementasikan untuk aplikasi penjejakan badan manusia bagian atas. Kekurangan dari metode ini, yaitu faktor pendeteksian badan manusia bagian atas sehingga pada saat proses penjejakan masih terdapat objek lain yan menyerupai badan manusia

bagian atas ikut terdeteksi sehingga faktor ini sangat berpengaruh pada saat penentuan arah kendali robot pada saat proses penjejakan sehingga membuat prosentase error dalam penjejakan badan manusia adalah 9.6515875 %.

3. Dengan melakukan proses transformasi citra dari format citra BGR ke RGB terlebih dahulu sebelum ke proses *grayscale* dan proses pendeteksian akan lebih efektif dengan kondisi intensitas cahaya antara 6 ev – 12 ev.

5.2 Saran

Agar pada penelitian selanjutnya sistem ini dapat dikembangkan lebih sempurna, maka penulis memberikan saran sebagai berikut:

1. Untuk mendapatkan proses deteksi yang lebih akurat, pada penelitian selanjutnya *Haar Training* yang digunakan sebaiknya membuat *Haar Training* sendiri. agar proses deteksi yang didapat bisa lebih efektif karena *Haar Training* yang dimiliki oleh OpenCV masih kurang efektif.
2. Aplikasi ini nantinya bisa digabungkan dengan beberapa aplikasi lain, seperti aplikasi pendeteksian sidik jari, pengenalan suara,pengenalan bau yang berbahaya, aplikasi pengenalan wajah dan aplikasi yang mengatur jalan dari robot untuk nantinya dibuat sebuah robot keamanan yang bisa melakukan pekerjaan dari keamanan sebenarnya.
3. Menggabungkan atau menggunakan beberapa metode pendeteksian badan manusia dalam satu aplikasi, agar didapat hasil yang lebih akurat. Seperti metode *HOG (Histogram of Gradient)* , Proses *EIG-Train* dan beberapa metode lain untuk mengurangi kesalahan pada saat penjejakan.

DAFTAR PUSTAKA

Budiharto,Widodo & Purwanto, Djoko. 2012 .Robot Vision: Teknik Membangun Robot Cerdas Masa Depan. Yogyakarta: Andi.

Tharom, Tabratas & Purbo, Onno W. 2000. Pengolahan Citra pada Mobil Robot. Bandung*: ITB.*

Hu, Chuanhu. 2009. *Reliable People Tracking Approach for Mobile Robot in Indoor Environments*.

Kobilarov, Marin & Sukhatme Gauray. 2006. *People Tracking and Following with Mobile Robot Using an Omnidirectional Camera and a Laser*.

Zheng, yuhua & Meng, Yan. 2009. *Real-time People Tracking and Following using a Vision-controlled Mobile Robot.*

Andriessen ,Daniel richard. 2012. Pengendalian *Mobile Robot* Berbasis *Webcam* menggunakan Perintah Isyarat Tangan

Color Spaces. http://compression.ru/download/articles/color_space/ch03.pdf Diakses 16 oktober 2013.

Logika fuzzy,Bab-II-kcb. http://file.upi.edu/Direktori/FPTK/JUR._PEND._TEKNIK_ELEKTRO/197211131999031-ADE_GAFAR_ABDULLAH/file_mk_Pengantar_Kecerdasan_Buatan_(9files)/Bab_II_KCB.pdf. Diakses 4 November 2013

Nahla, Gentang Syabba. 2010. *Tracking* Bola menggunakan Robotino

Feng, Hsuan-ming. 2009. *Intelligent Omni-directional Vision-based Mobile Robot Fuzzy Systems Design and Implementation*

Lai, Dong-Xian. 2009. *Active Contour Tracking of Moving Objects Using Edge Flows and Ant Colony Optimization in Video*.

Gao, Lufang. 2011. *Human Detection by Omni-directional Camera*.

Kusumanto, RD. 2012. Aplikasi *Sensor Vision* untuk Deteksi *Multiface* dan Menghitung Jumlah Orang.

Jain, Himansu Prakash. 2010. *Real-time Upper-body Human Pose Estimation using a Depth Camera.*