ITS
Institut
Teknologi
Sepuluh Nopember

TUGAS AKHIR - TE141599

RANCANG BANGUN SISTEM BANTU BERJALAN DI DALAM RUANGAN UNTUK TUNA NETRA BERBASIS KAMERA

Muhammad Reza Imaduddin
NRP 2210100132

Dosen Pembimbing
Ronny Mardiyanto, ST., MT., Ph.D.
Dr. Ir. Djoko Purwanto, M.Eng.

JURUSAN TEKNIK ELEKTRO
Fakultas Teknologi Industri
Institut Teknologi Sepuluh Nopember
Surabaya 2015

ITS
Institut
Teknologi
Sepuluh Nopember

FINAL PROJECT - TE141599

INDOOR WALKING GUIDANCE SYSTEM BASED ON CAMERA FOR THE VISUALLY IMPAIRED

Muhammad Reza Imaduddin
NRP 2210100132

Advisor
Ronny Mardiyanto, ST., MT., Ph.D.
Dr. Ir. Djoko Purwanto, M.Eng.

ELECTRICAL ENGINEERING DEPARTEMENT
Faculty of Industrial Technology
Institut Teknologi Sepuluh Nopember
Surabaya 2015

RANCANG BANGUN SISTEM BANTU BERJALAN DI DALAM RUANGAN UNTUK TUNA NETRA BERBASIS KAMERA

TUGAS AKHIR

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

Menyetujui :

Dosen Pembimbing I,	Dosen Pembimbing II,
Ronny Mardiyanto, ST., MT., Ph.D.	Dr. Ir. Djoko Purwanto, M.Eng.
NIP. 198101182003121003	NIP.195004021986012001

SURABAYA
JANUARI, 2015

RANCANG BANGUN SISTEM BANTU BERJALAN DI DALAM RUANGAN UNTUK TUNA NETRA BERBASIS KAMERA

M. Reza Imaduddin
2210100132

Dosen Pembimbing 1 : Ronny Mardiyanto, ST., MT.,Ph.D
Dosen Pembimbing 2 : Dr. Ir. Djoko Purwanto, M.Eng

ABSTRAK

Dewasa ini para penyandang tuna netra sangat kesulitan dalam menentukan arah mereka berjalan karena mereka hanya berpedoman pada intuisi dan pendengaran mereka. Berbagai macam cara dilakukan untuk menolong para tuna netra ini, seperti penggunan *Global Positioning System* (GPS), sensor ultrasonik, pemasangan vibrator, dan lain-lain. Cara-cara tadi sangat membebani tuna netra karena banyaknya alat yang harus dipasangkan ke tubuh mereka ataupun yang mereka bawa ketika sedang berjalan. Kamera menjadi salah satu cara untuk mengurangi beban-beban tadi karena mampu menangkap seluruh gambar yang ada di depan tuna netra, sehingga mereka tahu halangan apa saja yang ada di depan ketika akan berjalan. Dalam tugas akhir ini kamera yang digunakan adalah Kinect. Dimana Kinect memiliki lensa depth yang mampu mengukur jarak digunakan untuk memandu *user* untuk menghindari halangan yang ada di depannya. Metode yang digunakan adalah memanfaatkan sinar inframerah yang sinarnya akan terpantul dan ditangkap oleh lensa depth lalu informasi tersebut dikonversi menjadi kriteria-kriteria aman untuk berjalan yang akan disampaikan melalui suara. Dari hasil percobaan, tugas akhir ini memiliki eror rata-rata yang sangat kecil yaitu ±2,2% dalam pengukuran jarak, ±20% dalam pendeteksian manusia dan berhasil memandu *user* untuk menghindari halangan yang ada di depannya sekaligus mengetahui ada atau tidaknya manusia yang sedang berada di depannya dengan kondisi apapun.

Kata Kunci : sistem bantu berjalan orang buta, kamera, Kinect, *image processing, earphone*

INDOOR WALKING GUIDANCE SYSTEM BASED ON CAMERA FOR THE VISUALLY IMPAIRED

M. Reza Imaduddin
2210100132

1st Advisor : Ronny Mardiyanto, ST.,MT.,Ph.D
2nd Advisor : Dr. Ir. Djoko Purwanto, M.Eng

ABSTRACT

These days, the visually impaired need to know where they are going and what are there in front of them since they only consider to their hearing and intuition. Many researches have been developed to help them, such as applying GPS (Global Positioning System), ultrasonic sensors, and vibrator. Those researches still cannot solve the problem mentioned below.

Camera as one of the way to reduce the number of devices applied as mentioned below. In fact that camera could see what they are and there are in front of them so it would be so helpful and they could probably interact with others beside their environment. The camera used in this final project is Kinect for windows. Kinect has 3 lenses, RGB, depth, and infrared. Since Kinect has the infrared and depth lenses we can know the exact distance from user to their environment, so they would not be hit or crashed. The depth's lenses data converted from depth data into millimeter and from those data, created some criteria to guide user using speech guiding and from the RGB lenses using human detection method they could notice people at the front.

From the experimental results, this final project resulted average ±2,2% for the distance measurement, ±20% for human detection and this system could make the user guided to avoid obstacles and notice whether there are people or not at the front in every single condition.

Keywords: visual impaired guiding, Kinect, earphone

KATA PENGANTAR

Puji syukur kepada Tuhan Yang Maha Esa atas kasih dan rahmat-Nya penulis dapat menyelesaikan Tugas Akhir ini dengan judul :

RANCANG BANGUN SISTEM BANTU BERJALAN DI DALAM RUANGAN UNTUK TUNA NETRA BERBASIS KAMERA

Tugas Akhir ini merupakan persyaratan dalam menyelesaikan pendidikan program Strata-Satu di Jurusan Teknik Elektro, Fakultas Teknologi Industri, Institut Teknologi Sepuluh Nopember Surabaya.

Tugas Akhir ini dibuat berdasarkan teori-teori yang didapat selama mengikuti perkuliahan, berbagai literatur penunjang dan pengarahan dosen pembimbing dari awal hingga akhir pengerjaan Tugas Akhir ini.

Pada kesempatan ini, penulis ingin berterima kasih kepada pihak-pihak yang membantu pembuatan tugas akhir ini, khususnya kepada:

1. Kedua orang tua tercinta, Bapak Bambang Istiono dan Ibu Endang Susilowati, yang tidak pernah putus untuk seluruh do'a, nasihat, motivasi, dan dukungannya.
2. Bapak Ronny Mardiyanto, ST., MT., Ph. D., selaku dosen pembimbing pertama, atas bimbingan, inspirasi, pengarahan, dan motivasi yang diberikan selama pengerjan penelitian Tugas Akhir ini.
3. Bapak Dr. Ir. Djoko Purwanto, M. Eng., selaku dosen pembimbing kedua, atas bimbingan, inspirasi, pengarahan, dan motivasi yang diberikan selama pengerjan penelitian tugas akhir ini.
4. Resti Ramadhianti yang selalu memberi semangat dan memotivasi selama pengerjaan penelitian tugas akhir ini.
5. Rizki Anhar Ramadlan, ST yang turut membantu dalam penyelesaian tugas akhir ini.
6. Tonny Feriandi yang turut membantu dalam penyelesaian tugas akhir ini.
7. Mas Riza dan Mas Janu yang mengajari dari awal mengerjakan tugas akhir ini.
8. Teman-teman *Aviation Lover*, Aufar Gusviandi Pratama dan Arie Arifin, *keep flying guys.*
9. Teman-teman Gudang, Ari, Ananta, Angga, Aufar, Bagas, Imam, Rholly, Reza Yogatama, Sidqi, Wahyu, dan Wawan, *keep being crazy guys*

10. Seluruh rekan-rekan asisten laboratorium elektronika yang turut membantu dalam penyelesaian tugas akhir ini.

Penulis sadar bahwa Tugas Akhir ini masih belum sempurna dan masih banyak hal yang perlu diperbaiki. Saran, kritik dan masukan baik dari semua pihak sangat membantu penulis terutama untuk berbagai kemungkinan pengembangan lebih lanjut.

Surabaya, Desember 2014

Penulis

DAFTAR ISI

Halaman

DAFTAR TABEL

Halaman

DAFTAR GAMBAR

Halaman

BAB I
PENDAHULUAN

1.1 Latar Belakang

Kegiatan berjalan adalah salah satu kegiatan yang sulit dilakukan oleh para penyandang tuna netra. Para penyandang tuna netra membutuhkan waktu yang lama untuk menentukan posisi, jarak, dan arah mereka ketika akan melangkah ke titik berbeda dari titik asal mereka. Untuk menanggulangi permasalahan ini, diperlukan sistem panduan berjalan yang mampu memberitahu para penyandang tuna netra arah jalan mana yang akan mereka tuju.

Belum adanya sistem panduan orang buta di Indonesia menjadi salah satu kendala bagi para penyandang tuna netra untuk melangkah dengan mudahnya kemanapun mereka tuju serta mahalnya biaya operasi medis penggantian lensa mata yang mungkin tidak dapat dijangkau oleh beberapa penyandang tuna netra. Kamera, menjadi salah satu solusi untuk dapat memberikan visualisasi kepada para penyandang tuna netra agar mereka dapat mengetahui arah mana mereka akan berjalan yang nanti akan disampaikan melalui media suara.

Sebuah kamera mengambil video secara langsung yang besarnya 8-bit yang memiliki resolusi 640 X 480. Video tersebut lalu diolah kedalam sebuah *processing unit* dengan beberapa metode. Salah satunya adalah *houghline transform* dan *edge detection*. Kamera mendeteksi fitur – fitur seperti warna lantai yang ada di area berjalan para penyandang tuna netra. Lalu gambar di proses dengan *processing unit* adalah *Desktop Personal Computer (PC)* yang nantinya akan memberitahu arah kemana para penyandang tuna netra akan berjalan dengan bantuan suara yang nanti akan dipasangkan ke telinga mereka.

1.2 Perumusan Masalah

Permasalahan yang dibahas dalam tugas akhir ini adalah:

1. Bagaimana menjaga tuna netra agar tidak tertabrak oleh halangan yang berada di depannya saat berjalan.

2. Bagaimana mengkomunikasikan hasil olahan gambar kepada penyandang tuna netra menggunakan media suara.

1.3 Tujuan Penelitian

Penelitian pada tugas akhir ini bertujuan sebagai berikut :

1. Mampu menuntun dan memandu para penyandang tuna netra untuk berjalan dengan mandiri.
2. Mampu mengkomunikasikan adanya halangan yang berada di depan para penyandang tuna netra melalui media suara.

1.4 Batasan Masalah

Batasan masalah dari tugas akhir ini adalah sebagai berikut:

1. Pemanduan navigasi tuna netra hanya sebatas di dalam lorong sebuah gedung.
2. Penentuan jarak aman untuk tuna netra berjalan tidak lebih dari 3 meter.
3. Lokasi lorong pelaksanaan pemanduan harus seluruhnya solid.
4. Halangan yang terdeteksi tidak berbahan kaca.
5. Area pendeteksi halangan tidak mencakup *blind spot* yang berada pada bagian bawah tubuh *user*

1.5 Metodologi Penelitian

Dalam penyelesaian Tugas Akhir ini digunakan metodologi sebagai berikut :

1. Studi Literatur

Pada tahap ini dilakukan pengumpulan dasar teori yang menunjang dalam penulisan Tugas Akhir. Dasar teori ini dapat diambil dari buku-buku, jurnal, proceeding, dan artikel-artikel di internet.

2. Perancangan Hardware

Perancangan hardware yang akan digunakan dalam implementasi sistem ini meliputi kamera sebagai input visual dan *earphone* sebagai ouput suara yang akan dihubungkan masing-masing ke *processing unit.*

3. **Perancangan Software**

   Langkah pertama yang dilakukan pada tahap ini adalah pembuatan source code untuk menangkap citra kedalaman (*depth*) yang selanjutnya dilakukan konversi data jarak dalam bentuk data kedalaman menjadi bentuk milimeter. Data jarak dalam tugas akhir ini digunakan untuk memandu tuna netra untuk menentukan apakah dia berada di dalam jarak aman untuk berjalan dan mampu berjalan menghindari halangan yang berada di depannya.

4. **Pengujian Sistem**

   Pengujian Alat dilakukan untuk menentukan keandalan dari sistem yang telah dirancang. Pengujian dilakukan dalam dua tahap yaitu pengujian secara *online* dan *offline*. Pengujian *offline* dilakukan pada tahap pertama, yaitu menggunakan video lorong sebuah bangunan di komputer. Sedangkan pengujian *online* dilakukan secara *real time* kepada kondisi lorong bangunan pada waktu tersebut, pengujian ini dilakukan di dua buah jenis lorong yang berbeda yaitu, lorong full solid dan lorong semi solid. Pengujian ini juga dilakukan di dua waktu yang berbeda yaitu pada saat siang hari dan malam hari.

5. **Penulisan Laporan Akhir**

   Tahap penulisan laporan Tugas Akhir dilakukan pada saat tahap pengujian sistem dimulai serta setelahnya.

1.6 Sistematika Penulisan

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

- **Bab 1 : Pendahuluan**

  Bab ini meliputi latar belakang, perumusan masalah, tujuan, sistematika penulisan, metodologi, dan relevansi.

- **Bab 2 : Dasar Teori**

Bab ini menjelaskan tentang dasar-dasar teori yang dibutuhkan dalam pengerjaan tugas akhir ini, yang meliputi teori dasar pengambilan citra RGB, pengambilan citra kedalaman (*depth)*, konversi data kedalaman menjadi data millimeter, .

- **Bab 3: Desain Sistem dan Implementasi**
  Bab ini menjelaskan tentang perencanaan sistem baik perangkat keras (hardware) maupun perangkat lunak (software) untuk pengukuran jarak dan memandu tuna netra dalam berjalan agar tidak tertabrak sesuatu apapun yang ada di depannya
- **Bab 4 : Pengujian Alat**
  Bab ini menjelaskan tentang data yang didapat dari pengujian *takeoff* serta hasil evaluasi sistem tersebut.
- **Bab 5 : Penutup**
  Bab ini menjelaskan tentang kesimpulan meliputi kekurangan-kekurangan pada kerja alat dari hasil analisa serta saran untuk pengembangan ke depan.

1.7 Relevansi

Hasil dari tugas akhir ini diharapkan dapat memberikan solusi untuk kebutuhan takeoff *quadrotor* dengan menggunakan sensor jarak inframerah.

BAB 2
DASAR TEORI DAN TINJAUAN PUSTAKA

Dasar teori dalam bab ini menjelaskan tentang teori penunjang yang berhubungan dengan keseluruhan sistem pada tugas akhir ini. Sedangkan tinjauan pustaka dalam bab ini menjelaskan tentang sistem-sistem yang berhubungan dengan tugas akhir ini dan pernah diimplementasikan oleh penulis-penulis sebelumnya

2.1 Dasar Teori

Pada sub bab ini akan dibahas tentang beberapa hal yaitu, karakteristik kognitif, Kinect beserta pengertiannya, dan human detection, untuk lebih jelasnya bisa dilihat pada pembahasan di bawah ini

2.1.1 Karakteristik Kognitif

Lowenfeld (Friend, 2005 : 417) menggambarkan dampak kebutaan (*totally blind*) atau kurang lihat (*low vision*) terhadap perkembangan kognitif, dengan mengidentifikasi keterbatasan yang mendasar pada anak terdapat tiga bagian, yaitu :

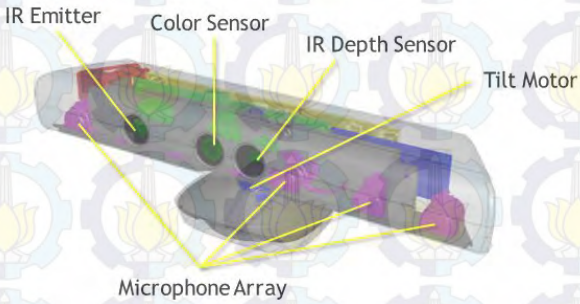
a. Tingkat dan Keragaman Pengalaman
b. Kemampuan untuk berpindah tempat (mobilitas)
c. Interaksi dengan lingkungan

dalam hal ini akan ditekankan pada kemampuan untuk berpindah (mobilitas), Penglihatan memungkinkan kita untuk bergerak dengan leluasa dalam suatu lingkungan,namun kebutaan atau hambatan penglihatan yang parah akan menghambat gerakantersebut. Keterbatasan tersebut membatasi seseorang dalam memperoleh pengalaman dan mempengaruhi hubungan sosial. Tidak seperti anak-anak lainnya, anak tunanetra harus belajar cara berjalan dengan aman dan efisien dalam suatu lingkungan dengan menggunakan berbagai keterampilan dan teknik orientasi dan mobilitas.

2.1.2 Kinect

2.1.2.1 Pengertian Kinect

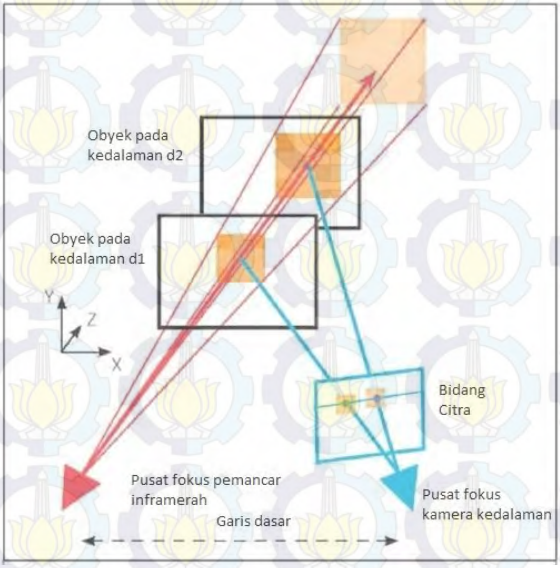
Kinect adalah perangkat yang digunakan sebagai pendeteksi gerakan penggunanya dimana bisa sebagai kontroler dan secara *realtime*. Kinect dibangun dari teknologi perangkat lunak yang didirikan oleh Rare, anak perusahaan Microsoft Game Studios[1]. *Sensor* kamera pada kinect dikembangkan oleh perusahaan dari Israel, yaitu Primesense dimana dapat menafsirkan gerakan pada tubuh tertentu ke dalam beberapa rangkaian kontrol. Kinect memiliki dua buah kamera utama, yaitu kamera *depth* dan kamera *RGB*, dan sebuah pemancar inframerah. Jarak optimal yang harus dipenuhi ketika melakukan proses penangkapan citra adalah 1.2 meter sampai dengan 3.5 meter[4]. Apabila jarak yang digunakan melebihi atau kurang dari jarak yang ditentukan, maka obyek tidak akan tertangkap oleh kamera. Pada bagian bawah kinect terdapat *multi-array mic* yang digunakan untuk merekam atau menginputkan suara. Kinect juga dilengkapi dengan *motorized tilt* dimana dapat digunakan untuk mengatur sudut pandang kamera, sehingga area yang bisa ditangkap oleh kamera menjadi lebih luas. Untuk mengatur sudut pandang kamera bisa menggunakan program tertentu, dengan jangkauan sudut kurang lebih 27 derajat.



Gambar 2.1 Kinect

Kamera kinect dapat digunakan pada berbagai perangkat lunak untuk komputer. Kinect merupakan kamera *stereovision,* dimana untuk proses pengambilan citra menggunakan dua arah sudut pandang yang berbeda.

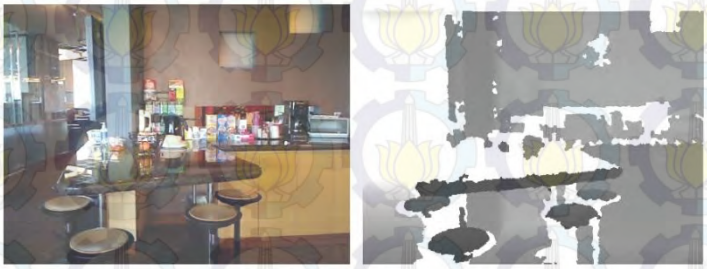
Kinect menggunakan dua buah kamera diantaranya adalah kamera *RGB* dan kamera *depth*, dan satu buah pemancar inframerah. Kamera *RGB* memiliki resolusi yang lebih luas apabila dibandingkan dengan kamera *depth*, dimana hasil dari kamera tersebut memiliki resolusi maksimal 640x480 piksel dengan rentang 30 fps berupa informasi citra *RGB*. Sedangkan kamera *depth* memiliki resolusi maksimal sebesar 640x480 piksel dengan rentang 30 fps. Pemancar inframerah berfungsi untuk memancarkan titik-titik inframerah dengan pola yang tidak teratur dan memiliki intensitas yang acak. Sinar inframerah yang dipancarkan tersebut akan mengenai obyek yang berada didepan kamera dan dipantulkan kembali sehingga dapat dikenali secara cepat dan tepat oleh kamera *depth*.



Gambar 2.2 Pancaran inframerah kinect[1]

Ditinjau dari hasil data yang berhasil ditangkap berdasarkan dari pancaran inframerah, kamera kinect dapat mendeteksi data *depth* dari suatu gambar. Data *depth* yang ditangkap oleh kamera nantinya digunakan untuk mendeteksi kedalaman obyek yang ada di depan kamera dan berada di

dalam *frame*. Data *depth* bisa ditampilkan pada *depth image* dalam beberapa macam warna sesuai yang diinginkan, karena tidak terlalu signifikan untuk hasilnya. Pada *depth image* menampilkan beberapa intensitas warna dari obyek yang berhasil ditangkap untuk mendeteksi jarak dari obyek tersebut. Untuk jarak obyek yang dekat dengan kamera, warna yang ditampilkan semakin gelap dan apabila semakin jauh jarak obyek warna yang ditampilkan semakin terang hingga batas jarak pandang kamera paling jauh. Kamera kinect memiliki jarak pandang minimum dan jarak pandang maksimum, sehingga obyek yang berada diluar batas jarak tersebut tidak memiliki data *depth*. Demikian juga untuk obyek yang tidak bisa memantulkan pancaran dari inframerah yang mengakibatkan kamera *depth* tidak menerima data dari obyek tersebut sehingga tidak ada data yang ditangkap, pada gambar ditampilkan dengan warna putih. Hasil dari data *depth* yang ditangkap oleh kamera dapat dilihat pada gambar 2.4.

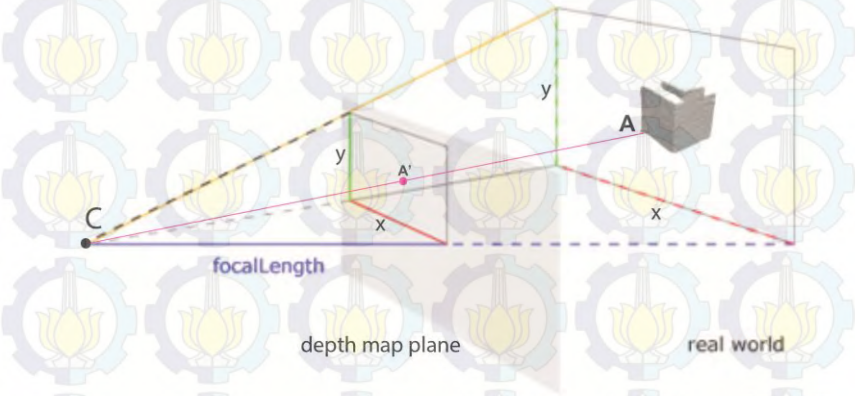


Gambar 2.3 *Image RGB* dan *Image Depth* pada Kinect

2.1.2.2 Konversi Data *Depth*

Dalam dunia grafika komputer dimensi tiga, peta kedalaman (depth map) adalah sebuah gambar dua dimensi yang berisi data kedalaman yang berisi informasi tentang jarak permukaan sebuah obyek terhadap sudut pandang (view point) tertentu. Jarak ini biasanya berhubungan dengan data kedalaman yang berupa koordinat Z dari sebuah koordinat kartesian. Dalam kasus ini, sumbu Z yang dimaksud adalah sumbu Z yang relatif terhadap view dari kamera, dan bukan merupakan sumbu Z dari koordinat dunia.

Untuk melakukan proses lebih lanjut, peta kedalaman harus dikonversi dari data dua dimensi menjadi data tiga dimensi yang memiliki sumbu X, Y, dan Z.



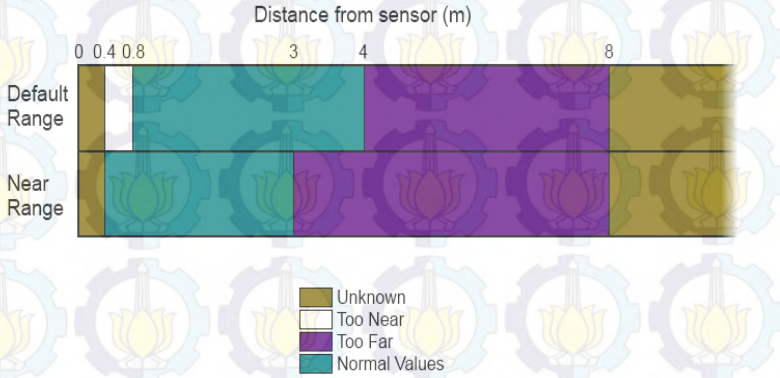
Gambar 2.4 Konversi *Depth Map*

Ilustrasi konversi peta kedalaman dapat dilihat pada gambar 2.5. Titik A diproyeksikan sebagai A' pada depth map plane. Dengan memperhitungkan focal length kamera, yaitu jarak antara pusat kamera (C) dan depth map plane, titik dalam dimensi dua pada depth map plane dapat dikonversikan menjadi titik dalam dimensi tiga. Konversi depth map ini melibatkan parameter intrinsik kamera yang disebut sebagai K dan juga melibatkan data depth map yang disebut sebagai D. Konversi ini dilakukan secara terpisah pada masing-masing titik di depth map yang dilambangkan sebagai u. Dengan kedua data tersebut, maka persamaan untuk melakukan konversi dapat dilihat pada persamaan 2-1. [2]

$$V_i(u) = D_i(u)K^{-1}[u, 1] \quad (2\text{-}1)$$

2.1.2.3 Depth Space Range

Depth Space Range adalah rentang jarak dari lensa *Depth* Kinect untuk menangkap jarak dari benda yang tertangkap oleh frame lensa tersebut. *Depth Space Range* memiliki dua buah mode, yaitu default mode dan near mode. Rentang jarak kedua mode dapat dilihat pada gambar 2.5



Gambar 2.5 *Depth space range*

dapat dilihat bahwa untuk mode default, jarak yang mampu diukur dan dibaca secara optimal mulai dari 80 cm hingga 4 meter dan untuk mode near, yaitu mulai dari 0 cm hingga 3 meter.

2.1.2.4 Human Detection

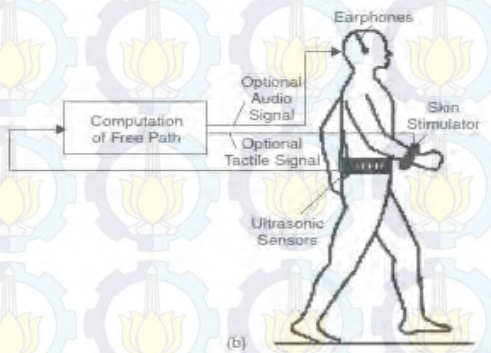
Dalam proses berjalan, mengetahui apa saja yang ada di depan *user* akan sangat membantu *user* dalam berinteraksi. *Human Detection* dipilih sebagai salah satu fitur yang diberikan untuk membantu *user*. *Human Detection* adalah sebuah metode untuk mendeteksi ada atau tidaknya orang yang berada di depan *user*. Metode ini memanfaatkan library tambahan yang ada pada opencv yaitu "haarcascade_upperbody.xml". Metode ini mendeteksi bagian atas dari manusia yang nantinya akan diproses dan dikomunikasikan kepada *user* apabila terdeteksi manusia yang berada di depannya.

2.2 Penelitian Terkait

Pada sub-bab ini akan dibahas mengenai beberapa penelitian yang sudah pernah dilakukan untuk membantu tuna netra diantaranya, *navbelt*, *guidecane* dan *three-dimensional sound*, juga membahas tentang *libray* program yang digunakan dalam tugas akhir ini

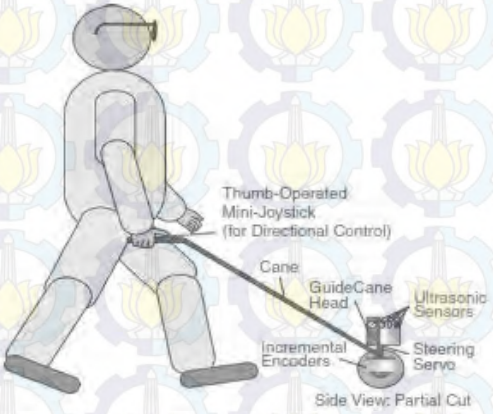
2.2.1 *NavBelt and GuideCane*

Navbelt dan *GuideCane* adalah salah satu metode yang digunakan untuk memandu tuna netra berjalan [3]. *Navbelt* adalah sebuah sabuk yang telah dimodifikasi dan berisi array ultrasonik sebanyak 8 buah yang diletakkan di bagian depan dari sabuk tersebut dimana setiap ultrasonik mampu menjangkau area 15° di sekitarnya maka total *navbelt* akan menjangkau sebesar 120 di sekitar bagian depan *user*. Ilustrasi dari *navbelt* bisa dilihat pada gambar 2.6. *Navbelt* akan mengirimkan sinyal ke komputer yang nantinya akan diproses dan mengirimkan sinyal keluaran ke *skin stimulator* yang akan memberi tanda bahaya apabila ada sesuatu benda yang berada di dekat *user*. *Navbelt* memiliki kelemahan ketika *user* ingin menghindari sebuah halangan di depannya, oleh karena itu *GuideCane* ditemukan, *GuideCane* adalah sebuah robot pemandu yang dikombinasikan dengan tongkat berjalan *user*.



Gambar 2.6 *Navbelt*

GuideCane juga memiliki array ultrasonik yang menjangkau area seluas 120° di depan sistem ini dan sepasang servo sehingga *user* dapat merasakan arah pergerakannya dan mampu melewati halangan yang berada di depan *GuideCane*. *GuideCane* dikendalikan dari 4 buah tombol yang diletakkan di handle yang berada di ujung tongkat. Tombol - tombol itu adalah arah - arah kemana *user* akan berjalan seperti, maju, mundur, kanan dan kiri. Tapi sekali lagi sistem ini belum mampu untuk mengkomunikasikan deteksi apa yang ada di depan dari si *user* sehingga *user* masih buta tentang apa – apa saja yang ada di depan mereka ketika mereka berjalan. Ilustrasi dari *GuideCane* dapat dilihat pada gambar 2.7.

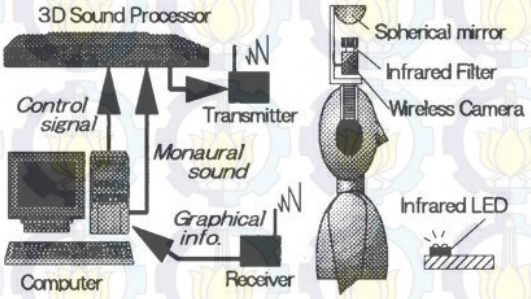


Gambar 2.7 User menggunakan *GuideCane*

2.2.2 *Three-Dimensional Sound (Suara Tiga Dimensi)*

Three-Dimensional Sound system atau bisa disebut sistem suara tiga dimensi adalah sebuah metode baru yang sedang dikembangkan juga untuk memandu para tuna netra. Sistem ini terdiri dari omni-vision image sensor dan prosesor suara tiga dimensi. Untuk bagian prosesor tiga dimensi ini sendiri terdiri dari 3 buah speaker, 2 buah yang dipasangkan ke user dan 1 buah lagi menjadi referensi sumber suara panduan yang akan memandu tuna netra dari hasil pengolahan citra gambar yang dikonversikan menjadi suara [4]. Tetapi sekali lagi sistem ini belum

mampu mengkomunikasikan apa saja yang berada di sekitar user, sehingga sistem ini masih perlu pengembangan lebih lanjut. Ilustrasi dari sistem ini dapat dilihat dari gambar 2.8.



Gambar 2.8 *Three-Dimensional sound using omni-vision sensor*

2.2.3 OpenCV

OpenCV (Open source Computer Vision) merupakan sebuah library dari fungsi programming untuk real-time computer vision. OpenCV dapat dimanfaatkan oleh program-program lainnya (seperti C++, C dan Phyton) untuk melakukan pengambilan, pengolahan serta penampilan data gambar, baik dalam bentuk image dan video maupun real-time video. OpenCV bersifat Open Source (dapat digunakan secara

Gambar 2.9 Logo OpenCV[4]

bebas) baik untuk akademik maupun untuk komersil, yang dapat berjalan dengan operating system Windows, Linus, Android dan Mac.

2.2.4 Microsoft Kinect SDK

Gambar 2.10 Logo Kinect for Windows[1]

Microsoft Kinect SDK (Software Development Kit) merupakan sebuah library dari fungsi programming untuk penggunaan divais khusus yaitu Kinect. Library ini dimanfaatkan oleh program-program lainnya (seperti C++, C# dan lain-lain) untuk melakukan pengambilan, pengolahan serta penampilan data gambar, baik dalam bentuk image dan video maupun real time video. Library ini juga sering digunakan dalam hal interaksi manusia dengan mesin sehingga user dapat langsung berinteraksi dengan program yang telah dibuat oleh programmer. Library ini juga biasa digunakan dalam pembuatan sebuah game yang tentunya basis dari game tersebut adalah Kinect itu sendiri untuk melakukan kontrol dalam game tersebut.

Gambar 2.10 Interaksi manusia dengan komputer melalui kinect

BAB 3
DESAIN SISTEM DAN IMPLEMENTASI

Sistem yang dirancang dalam tugas akhir ini hanya memiliki satu bagian yang saling berhubungan tiap komponennya, sistem ini terdiri dari Kinect yang berfungsi untuk menangkap data lorong jalan, komputer yang berfungsi untuk memproses data hasil tangkapan Kinect, dan *earphone* yang berfunsgi untuk menyampaikan informasi yang telah diolah oleh komputer menuju user yang dalam hal ini adalah tuna netra, cara kerja sistem secara keseluruhan dapat diilustrasikan dalam gambar berikut ini:

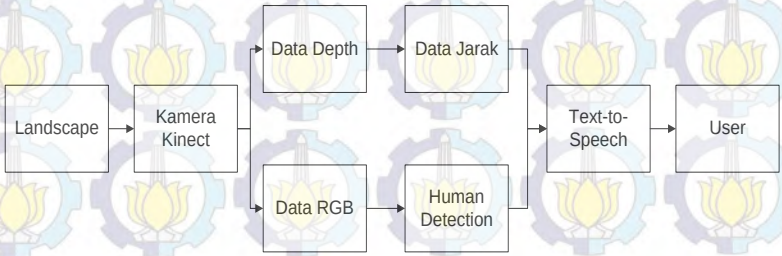
Gambar 3.1 Ilustrasi cara kerja sistem

3.1 Perancangan Sistem

Sistem yang saling berhubungan ini bertugas untuk menuntun tuna netra ketika sedang berjalan agar tidak tertabrak oleh benda-benda yang ada di sekitarnya. Cara kerja sistem ini secara keseluruhan sebagai berikut:

- Kinect menangkap data gambar dan kedalaman dari lorong
- Untuk data RGB yang ditangkap oleh Kinect dilakukan proses smoothing untuk menghilangkan noise dan detil-detil yang sangat kecil
- Kemudian diubah dari RGB direduksi menjadi citra grayscale untuk mempermudah proses pengolahan selanjutnya
- Kemudian dilakukan deteksi manusia yang memanfaatkan file tambahan dari opencv
- Untuk data *Depth* yang ditangkap oleh Kinect dilakukan proses konversi data *Depth* menjadi data millimeter
- Hasil dari pengolahan citra tersebut tadi disampaikan ke user menggunakan media suara

Blok diagram dari sistem dapat dilihat dari gambar 3.2 sebagai berikut



Gambar 3.2 Diagram blok sistem keseluruhan

3.2 Perangkat Keras

Perangkat keras yang digunakan pada Tugas Akhir ini merupakan fungsi standart dari sebuah laptop atau komputer sebagai pengolah citra, kemudian Kinect sebagai sensor visual untuk pengambil data gambar RGB dan *Depth* serta earphone untuk mengkomunikasikan hasil pengolahan citra kepada *user*. Berikut adalah perangkat-perangkat keras yang digunakan beserta spesifikasinya :

3.2.1 Kinect sebagai sensor visual

Kinect yang digunakan adalah Kinect generasi pertama atau yang sering disebut dengan Kinect Xbox 360 dengan spesifikasi :

Kinect memiliki 1 buah lensa RGB dengan spesifikasi sebagai berikut :

- *Resolution* : 1600 x 1200 *pixels*
- *Frame Rate* : 30 *frame per second*

Kinect juga memiliki 1 buah lensa *Depth* dengan spesifikasi sebagai berikut :

- *Resolution* : 640 x 480 *pixels*
- *Frame rate* : 60 *frame per second*
- *Operation Range* : 0 - 3 m

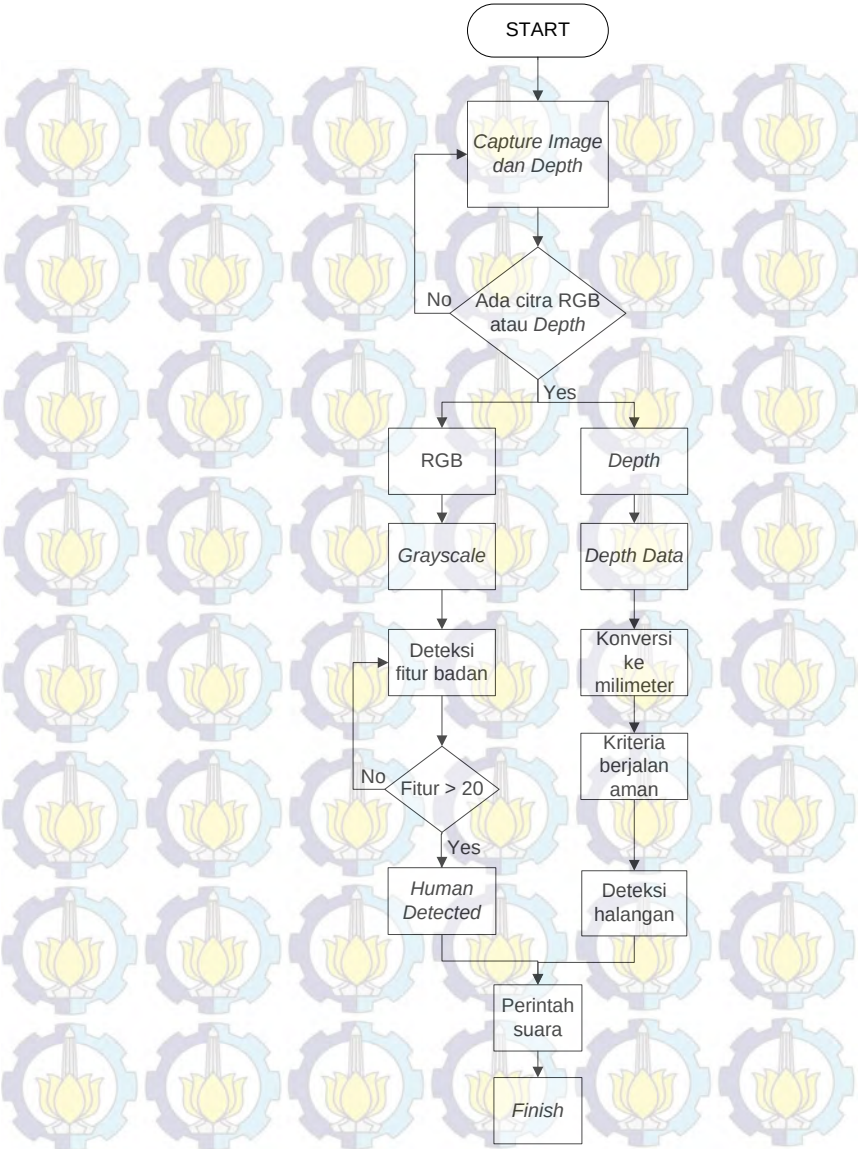
3.2.2 Komputer

pada tugas akhir ini digunakan sebuah komputer dengan spesifikasi :

- CPU : Intel Core i5-3210M (2.50 GHz)
- RAM : 4 GB

3.3 Perangkat-Perangkat Lunak

Pada bagian perancangan lunak ini meliputi algoritma pemrograman yang digunakan antara lain pengambilan data, pengolahan data, dan interpretasi data.

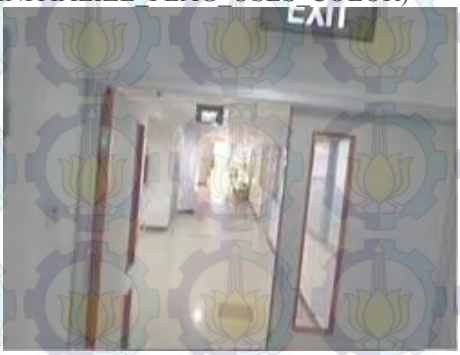


Gambar 3.3 *Flow chart image processing* sistem

3.3.1 Pengambilan Data RGB

Untuk melakukan pengambilan data, digunakan Kinect sebagai sensor visual, Microsoft Visual C++ sebagai program utama dan opencv dan Microsoft Kinect SDK sebagai library untuk mengambil, mengolah, dan menampilkan data gambar.

```
hr = m_pNuiSensor->
NuiInitialize(NUI_INITIALIZE_FLAG_USES_COLOR)
```

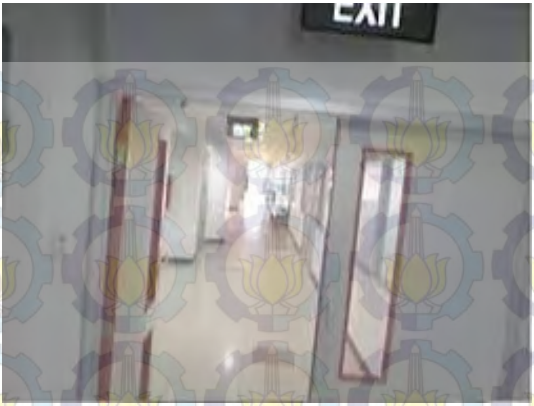


Gambar 3.4 Video capture data RGB

3.3.2 Smoothing Gambar

Setelah data berupa gambar diambil, akan terdapat noise (detil-detil kecil yang sangat kecil) serta warna yang sangat mencolok. Oleh karena itu, perlu dilakukan penghalusan (penghamburan) dengan menggunakan Gaussian *Blur*, sehingga persebaran warna akan menjadi merata dan noise dapat dikurangi seminimal mungkin

```
cvSmooth(color,frame_copy,CV_GAUSSIAN,3,3,0,0);
```

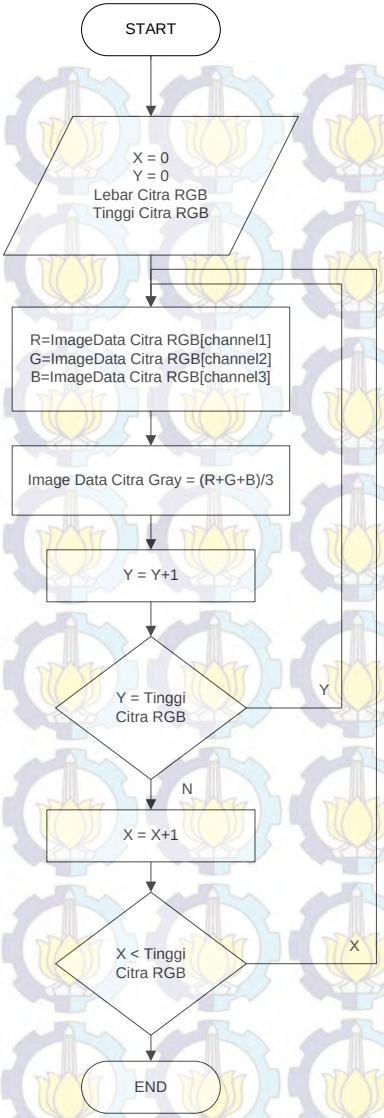


Gambar 3.5 Video capture hasil smoothing

3.3.3 Grayscaling

Untuk memudahkan proses pengolahan dan pemrosesan maka perlu untuk mereduksi matriks penyusun citra dari 3 matrik menjadi 1 matrik. Proses ini dinamakan grayscaling yaitu mengubah citra berwarna mnjadi bentuk citra grayscale.

Gambar 3.6 Video capture hasil *grayscale*



Gambar 3.7 *Flow chart* proses *grayscaling* video

3.3.4 Human Detection

Untuk menginformasikan *user* apakah ada seseorang di depannya, maka human detection dipilih di dalam tugas akhir ini untuk membantu *user* dalam berinteraksi dengan lingkungan di sekitarnya. Human detection memanfaatkan file tambahan dari opencv yaitu "haarcascade_upperbody.xml" untuk mengetahui apakah ada orang di sekitar area yang tertangkap oleh kamera. Fungsi untuk mendeteksi ada atau tidaknya orang yaitu,

CvSeq* body = cvHaarDetectObjects(small_img, cascade, storage, 1.1, 2, 0|CV_HAAR_FIND_BIGGEST_OBJECT,cvSize(100, 100));

Dalam tugas akhir ini digunakan metode haar cascade, yang merupakan bagian dari metode *viola jones.* Metode *viola jones* adalah metode pendeteksi objek yang terdapat EmguCV, metode ini merupakan metode yang paling banyak digunakan untuk mendeteksi objek, hal ini dikarenakan metode *viola jones* memiliki algoritma yang efisien, sehingga tidak memerlukan waktu lama dalam melakukan proses pendeteksian objek. Teori ini menjumlahkan nilai-nilai dari beberapa citra (I) dalam sebuah kotak *frame.* Penjumlahan tersebut dapat dilihat dalam formula di bawah ini,

$$\sum_{k} \delta_k B_k(I)$$

dimana, $\delta \in \{1, -1\}$ dan

$$B_k(I) = \sum_{i=u1(k)}^{u2(k)} \sum_{j=v1(k)}^{v2(k)} I_y$$

Beberapa fitur dapat dengan cepat di evaluasi oleh integral *image*, dimana formula integral *image* dapat di bawah ini,

$$I_y = \sum_{u=1}^{i} \sum_{v=1}^{j} I_{uv}$$

dimana u dan v adalah nilai dari pixel integral *image*. Maka dari itu penjumlahan nilai dari sebuah *frame* dapat dievaluasi dengan 4 tahap integral *image*. Maka dengan mudah kita dapat memeriksa bahwa

$$\sum_{i=u1}^{u2} \sum_{j=v1}^{v2} I_y = I_{u2v2} - I_{u1v2} - I_{u2v1} + I_{u1v1}$$

setiap fitur dari citra yang tertangkap dapat dievaluasi dengan metode integral image[6]. Algoritma deteksi *upper body* dapat dilihat pada tabel di bawah ini

1. Deteksi *upper body*
    - Membuat sebuah detektor objek *cascade.*
    - Baca *frame video* dan jalankan detektor.
    - Gambar sebuah kotak yang mengelilingi *upper body* yang terdeteksi.
2. Identifikasi fitur *upper body* untuk dilakukan pelacakan
    - Dapatkan informasi *upper body* dengan cara mengekstrak *Hue* dari *frame video.*
    - Ubah ke dalam bentuk warna Hue, Saturation and Value (HSV).
    - Tampilkan *channel data Hue* dan gambar kotak di sekitar *upper body.*
    - Deteksi *upper body* di sekitar area *frame video.*
3. Pelacakan *upper body*
    - Buat sebuah pelacak objek.
    - Inisialisasi pelacak histogram menggunakan *channel pixel Hue.*
    - Buat sebuah *video* untuk menampilkan *frame video.*
    - Lacak *upper body* di video hingga video selesai
    - Ekstrak *frame video* selanjutnya
    - Beri sebuah kotak mengelilingi *upper body* pada objek yang terdeteksi
    - Tampilkan dalam *video* yang telah diolah

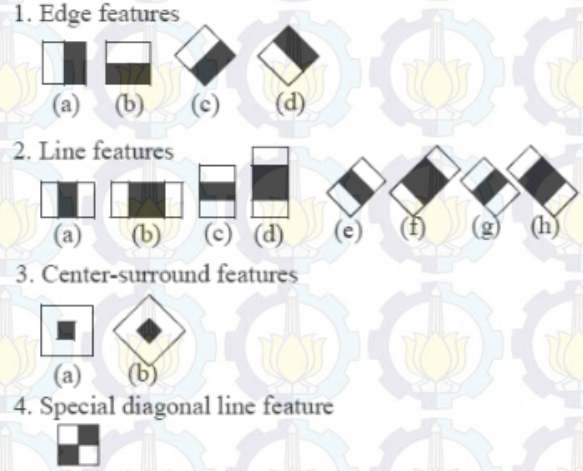
Proses pendeteksian objek dilakukan dengan mengklasifikasikan sebuah citra. Terdapat 4 hal tahap penting dalam teori *viola jones,* yaitu :

- Fitur
- *Integral image*
- *Adaptive boosting* atau *AdaBoost*
- *Kombinasi Classifier of Cascade*

3.3.4.1 Fitur

Fitur merupakan tahap paling awal yang diperlukan dalam pendeteksian objek. Penggunaan fitur dilakukan karena pemrosesan fitur

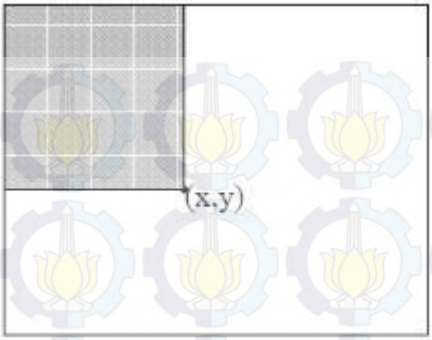
berlangsung lebih cepat dibandingkan pemrosesan *image* per *pixel*. Fitur yang digunakan oleh *viola-jones* berdasarkan pada *Wavelat Haar*. *Wavelat Haar* merupakan gelombang tunggal bujur sangkar yang mempunyai satu interval tinggi dan satu interval rendah yang kemudian dikembangkan untuk pendeteksian objek visual yang lebih dikenal dengan fitur *Haar* atau fitur *Haarlike.*



Gambar 3.8 Fitur-fitur Haar (Lienhar, Kuranove, dan Pisarevsky)

3.3.4.2 Integral Image

Integral image merupakan tahap kedua yang dilakukan dalam metode *viola jones*. *Integral image* adalah sebuah *image* yang nilai tiap *pixel*-nya merupakan akumulasi dari nilai pixel atas dan kirinya. Sebagai contoh, pixel (a,b) memiliki nilai akumulatif untuk semua *pixel* (x,y) dimana $x \leq a$ dan $y \leq b$.



Gambar 3.9 Nilai dari integral image pada titik (x,y) adalah jumlah dari semua pixel dari atas sampai kiri

Menurut viola dan jones *integral image* pada lokasi x,y berisikan jumlah pixel dari atas sampai kiri dari x,y, perhitungannya dapat dicari dengan menggunakan rumus di bawah ini :

$$ii(x,y) = \sum_{x' \le x, y' \le y} i(x', y')$$

dimana

ii(x,y) = *integral image*

i(x,y) = *original image*

lalu dengan menggunakan pasangan rumus berikut ini :

$$s(x,y) = s(x,y-1) + i(x,y)$$

$$ii(x,y) = ii(x-1,y) + s(x,y)$$

dimana s(x,y) adalah penjumlahan kumulatif baris, s(x-1) = 0, dan ii(-1,y) = 0 sehingga *integral image* dapat dihitung dengan mengabaikan *original image*[7]. Contoh perhitungan *integral image* dapat dilihat pada gambar 3.9 di bawah ini

Gambar 3.10 Contoh perhitungan integral image

3.3.4.3 Adaptive Boosting atau AdaBoost

AdaBoost merupakan tahap ketiga dalam metode *viola-jones*. Algoritma AdaBoost berfungsi untuk melakukan pemilihan fitur-fitur dalam jumlah banyak, dengan hanya memilih fitur-fitur tertentu.

Boosting merupakan meta-algortima dalam *machine learning* untuk melakukan *supervised learning*. Kebanyakan teori *boosting* mengikuti sebuah rancangan. Secara umum *boosting* terjadi dalam iterasi, secara *incremental* menambahkan *weak learner* ke dalam satu *strong learner*. Pada setiap iterasi satu *weak learner* belajar dari suatu data latihan. Kemudian *weak learner* itu ditambahkan ke dalam *strong learner*. Setelah *weak learner* ditambahkan, data-data kemudian diubah masing-masing bobotnya. Data-data yang mengalami kesalahan klasifikasi akan mengalami penambahan bobot, dan data-data yang terklasifikasi dengan benar akan mengalami pengurangan bobot. Oleh karena itu, *weak learner* pada iterasi selanjutnya akan lebih terfokus pada data-data yang mengalami kesalahan klasifikasi oleh *weak learner* yang sebelumnya[6].

Menurut *viola-jones* salah satu metode praktis untuk beradaptasi dengan *weak learner* yaitu dengan membatasi *weak learner* ke set klasifikasi fungsi, yang masing-masing bergantung pada fitur tunggal. Untuk mendukung tujuan ini, algoritma pembelajaran yang lemah dirancang untuk memilih fitur persegi panjang tunggal, dimana persegi panjang tunggal merupakan yang terbaik untuk memisahkan contoh positif dan negatif. Untuk masing-masing fitur *weak learner* menentukan ambang batas klasifikasi fungsi yang optimal, sehingga jumlah minimum kesalahan sebuah pengklasifikasian yang lemah ($h_j(x)$) terdiri dari fitur (f_j), sebuah threshold (θ_j) dan kesamaan (p_j) menunjukkan arah dari ketidaksetaraan tanda :

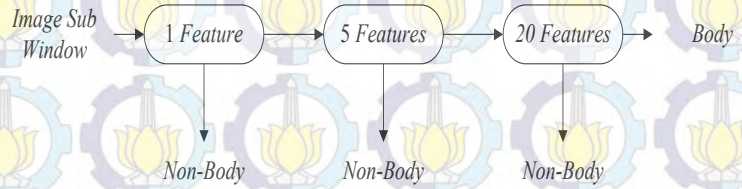
$$h_j(x) = \begin{cases} 1 \; jika \; p_j f_j(x) < p_j \theta_j \\ 0 \qquad\qquad\qquad lainnya \end{cases}$$

dimana x adalah sebuah 24 x 24 *pixel sub-window* dari sebuah gambar.

3.3.4.4 Kombinasi Classifier of Cascade

Kombinasi *Cascade of Classifier* merupakan tahap terakhir dalam metode *viola-jones*. Dengan mengkombinasikan pengklasifikasian dalam sebuah struktur *cascade* atau *cascade of classifier*, kecepatan dari proses pendeteksian dapat meningkat, yaitu dengan cara memusatkan perhatian pada daerah-daerah dalam *image* yang berpeluang saja. Hal ini dilakukan untuk menentukan dimana letak objek yang dicari pada suatu *image*.

Karakteristik dari algoritma *viola-jones* adalah adanya klasifikasi bertingkat. Klasifikasi pada algoritma ini terdiri dari 3 tingkatan dimana tiap tingkatan mengeluarkan *subimage* yang diyakini bukan objek. Hal ini dilakukan karena lebih mudah untuk menilai *subimage* tersebut bukan objek yang ingin dideteksi daripada menilai apakah *subimage* tersebut merupakan objek yang ingin dideteksi. Alur kerja klasifikasi bertingkat dapat dilihat di gambar 3.10 di bawah ini



Gambar 3.11 Diagram blok kombinasi classifier of cascade

Struktur *cascade* mencerminkan fakta bahwa dalam setiap gambar tunggal mayoritas berasal dari *sub-window* negative. Dengan demikian, *cascade* berguna untuk menolak nilai negatif sebanyak mungkin dan secepat mungkin kembali ke level yang memungkinkan. Oleh karena itu, dibutuhkan suatu formula untuk menghitung tingkat positif yang salah dari suatu *cascade classifier*. Menurut penelitian *viola-jones*, formula untuk menghitung hal tersebut adalah:

$$F = \prod_{i=1}^{k} f_i,$$

dimana,
F = tingkat positif yang salah dari *cascaded classifier*
K = jumlah *classifier*
f = kesalahan tingkat positif dari *classifier* ke *i.*

sedangkan rumus untuk menghitung tingkat deteksi adalah:

$$D = \prod_{i=1}^{K} d_i,$$

dimana
D = tingkat deteksi dari *cascaded classifier*
K = jumlah *classifier*
d = tingkat deteksi dari *classifier* ke *i*

pemberian *sub-window* akan berlangsung turun melalui *cascade* dari satu *classifier* pada satu waktu sampai diputuskan bahwa *window* tersebut negatif, sedangkan *window* yang berhasil dalam masing-masing tes akan diberi label positif. Menurut *viola-jones* rumus untuk menghitung jumlah yang diharapkan dari fitur yang dievaluasi yaitu :

$$N = n_0 + \sum_{i=1}^{K} \left(n_i \prod_{j<i} p_j \right)$$

dimana
N = jumlah yang diharapkan dari fitur yang dievaluasi
K = jumlah *classifier*
p = tingkat positif dari *classifier* ke *i*
n = jumlah fitur dalam *classifier* ke *i*

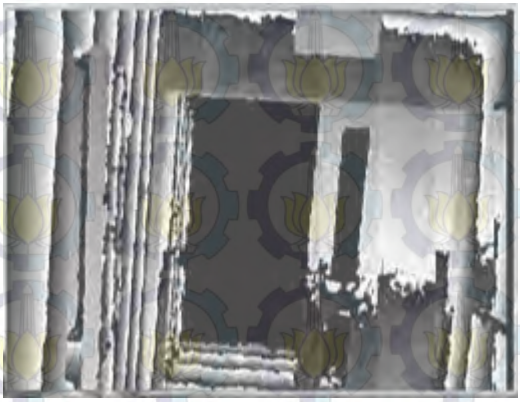
Menurut *viola-jones* dari analisa di atas dapat diambil kesimpulan bahwa terdapat tiga factor yang mempengaruhi kecepatan dan tingkat akurasi dalam pendeteksian, factor-faktor tersebut antara lain :

1. Jumlah dari tahapan *classifier*
2. Jumlah dari *feature* n_i untuk setiap tahapan
3. *Threshold* dari setiap tahapan.

3.3.5 Pengambilan Data Depth

Untuk melakukan pengukuran jarak, maka fungsi lensa depth dari Kinect diaktifkan agar hasil pantulan dari sinar inframerah dapat ditangkap dan dapat diolah menjadi data milimeter. Fungsi untuk mengaktifkan lensa depth yaitu,

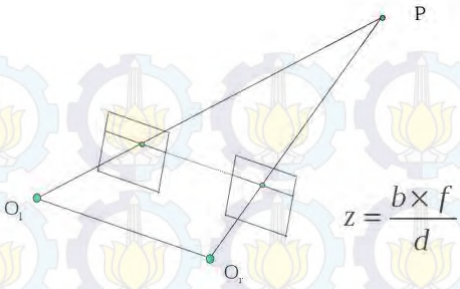
```
hr=m_pNuiSensor->
NuiInitialize(NUI_INITIALIZE_FLAG_USES_DEPTH);
```



Gambar 3.12 Pengambilan video dengan menggunakan lensa depth

3.3.6 Konversi Data Depth menjadi Milimeter

Data kedalaman yang dikonversi menjadi data jarak dalam tugas akhir ini digunakan untuk menjaga jarak aman tuna netra agar tidak tertabrak benda-benda yang ada di sekitar area dia berjalan. Pengukuran jarak dilakukan dengan memanfaatkan sinar laser inframerah dan lensa *depth* yang dimiliki oleh lensa Kinect digunakan sebagai projector yang akan menangkap kembali sinar laser inframerah. Proses pengukuran jarak ini dapat dilihat pada gambar 3. sebagai berikut



Gambar 3.13 Ilustrasi perhitungan dan pengukuran jarak menggunakan lensa depth

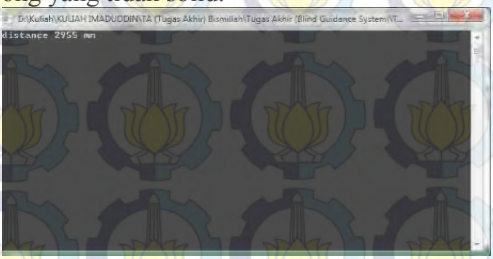
dari gambar di atas, jarak (z) dapat dihitung dengan formula

$$z = \frac{b\ x\ f}{\frac{1}{8}(1091.5 - d_{kinect})}$$

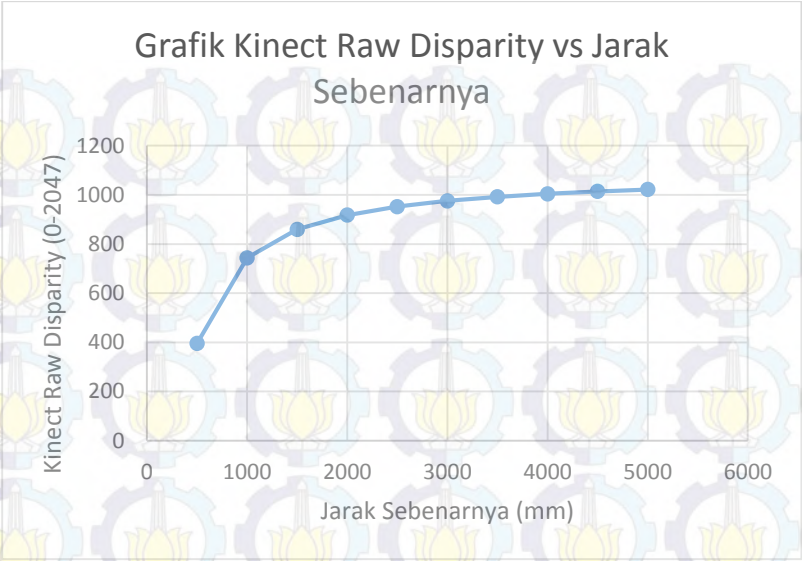
dimana,

Z = jarak benda dengan Kinect dalam satuan cm
b = jarak antara lensa IR dan Depth (projector)
f = fokus lensa IR Kinect (580 pixel units)
d = nilai kedalaman benda yang ditangkap projector (raw disparity)

hasil perhitungan dan konversi dari *depth map* di atas menghasilkan data jarak dalam satuan millimeter. Dari hasil perhitungan dan pengukuran jarak tadi maka diklasifikasikan menjadi beberapa variabel yaitu, jarak aman berjalan, jarak deteksi halangan, dan jarak aman ketika ada lorong yang tidak solid.



Gambar 3.14 Hasil konversi data depth menjadi data milimeter



Gambar 3.15 Hasil konversi data depth menjadi data milimeter

3.3.7 Text to Speech

Untuk menyampaikan informasi yang dibutuhkan oleh *user* yang dalam tugas akhir ini notabene adalah tuna netra, maka penyampaian melalui media suara sangat dibutuhkan agar *user* tahu apakah dia berada dalam zona aman ketika akan berjalan atau apakah ada sesuatu benda yang berada di depannya, dalam tugas akhir ini digunakan Microsoft Sound-API, berikut adalah fungsi untuk mengaktifkan Sound-API tersebut

```
HRESULT hr = CoCreateInstance(CLSID_SpVoice, NULL,
CLSCTX_ALL, IID_ISpVoice, (void **)&pVoice);
```

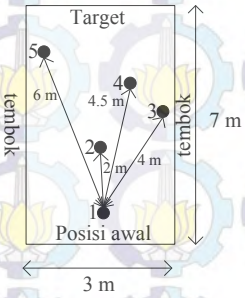
3.3.8 Proses Guiding

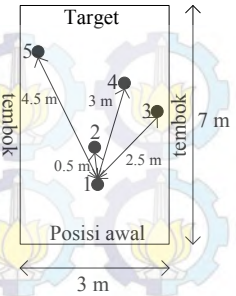
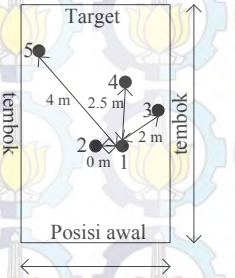
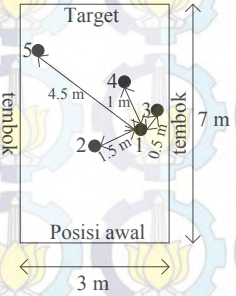
Proses *guiding* atau menuntun *user* pada tugas akhir ini ditekankan kepada pendeksian adanya halangan atau tidak yang berada di area yang tertangkap oleh lensa Kinect. Berikut adalah kriteria-kriteria dalam pemanduan *user*:

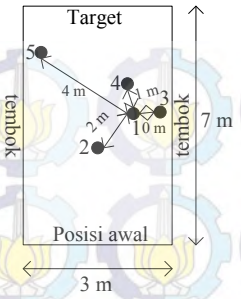
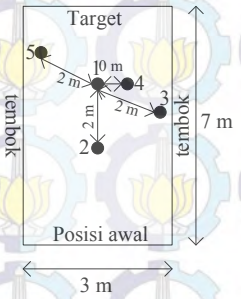
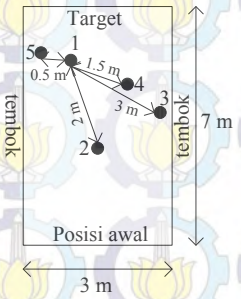
Tabel 3.1 Tabel kriteria-kriteria pemanduan user

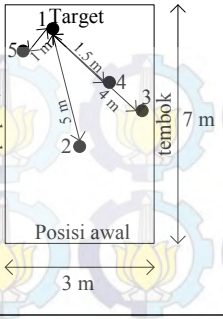
No.	Kondisi	Jarak yang Tertangkap Lensa	Informasi yang disampaikan
1.	Jalan Kosong	1-2.9 m	Boleh jalan
2.	Jalan Buntu	2.95-3 m	Berhenti, tidak ada jalan
3.	Jalan tidak semuanya solid	> 7 m	Berhenti, jalan tidak semua solid
4.	Jalan Terdapat Halangan	0.5-1 m	Berhenti, cari jalan lain

Berikut adalah tabel skenario langkah-langkah penuntunan *user* dari posisi awal berjalan menuju tujuan atau target yang diinginkan secara acak tergantung dari keinginan *user*

No	Ilustrasi Posisi *user* dan halangan	Keterangan
1.		• Pada posisi awal ini, *user* mencari dahulu jarak aman untuk dia berjalan lurus secara aman dengan menoleh kanan atau kirinya terlebih dahulu.

2.		• Pada posisi ini *user* berhenti sejenak karena terdeteksi ada halangan di jarak 0.5 m dan *user* akan mencari lagi jalan yang aman untuk melewati halangan yang ada di depannya, dalam kasus ini *user* dapat memilih jalan yang ada di depan kanan dan depan kirinya.
3.		• Pada posisi ini *user* memilih jalan sebelah kanan dari halangan yang ada di depannya dan melewatinya, meskipun arah jalannya belum tentu sama dengan arah jalan sebelumnya dan tetap mengikuti arahan dari program.
4.		• Pada posisi ini *user* tetap terus berjalan lurus sesuai yang diarahkan dengan program hingga terdeteksi lagi ada halangan di titik 3 dan *user* kembali berhenti dan mencari lagi jalan yang aman untuk dia lewati.

5.		• Pada posisi ini *user* menghindari halangan yang ada di depannya ke arah kirinya karena tidak ada jarak aman lagi yang berada di bagian kanannya, tetapi karena ada halangan lagi di titik 4 maka *user* berhenti lagi dan mencari jalan aman lagi.
6.		• Pada posisi ini *user* memilih jalan yang ada di kirinya karena jarak antara titik 3 dan 4 masih belum termasuk jarak aman bagi *user* untuk berjalan. *user* terus berjalan sesuai dengan arahan dari program.
7.		• Pada posisi ini kembali terdeteksi halangan di titik 5, sehingga *user* kembali berhenti dan mencari jalan yang aman.

8.		• Pada posisi ini *user* memilih jalan yang berada di sebelah kanannya dan berjalan lurus hingga berhasil mencapai target

 # Halaman ini sengaja dikosongkan #

BAB 4
PENGUJIAN DAN ANALISA DATA

Pada tugas akhir ini, pengujian dilakukan untuk mengetahui kinerja dari sistem. Pengujian sistem ini dilakukan pada dua jenis lorong yang berbeda, yaitu lorong yang seluruhnya solid dan lorong yang pada salah satu sisinya berpagar (tidak solid) dan pada waktu yang berbeda, yaitu siang hari dan malam hari dengan pencahayaan dan tanpa pencahayaan sehingga tingkat kesalahan (*error*) dapat diketahui dan performa sistem yang optimal dapat diketahui.

Dengan menggunakan *Depth Space Range*[1] yang sudah ditentukan dari Kinect, maka kinerja optimal dari Kinect dengan menggunakan mode dekat (*near mode*) adalah sebesar dari 0 – 3 m sehingga seharusnya ketika pengukuran yang melebihi batas tersebut akan mengalami sedikit *error* karena sudah diluar dari kemampuan optimal dari kinect tersebut.

4.1 Pengujian Pengukuran Jarak di Dalam Lorong yang Seluruhnya Solid

Dengan menggunakan metode-metode yang sudah dijelaskan pada bagian disain dan implementasi, maka dilakukan pengujian terhadap lorong yang seluruhnya solid apakah sistem mampu menjaga *user* tetap berada dalam jarak aman untuk berjalan atau tidak. Dalam hal ini pengujian sudah dilakukan dengan dua jenis pencahayaan yang berbeda, yaitu menggunakan pencahayaan lampu dan tidak menggunakan cahaya lampu, kamera diletakkan di atas helm dan diarahkan ke bawah sebesar -20°, berikut data-data yang didapatkan dari hasil pengujian

Dari data di atas dapat diketahui bahwa error rata-rata dari pengukuran di atas mendapatkan nilai 2.15% dan disimpulkan bahwa jarak yang masih termasuk jangkauan dari Kinect dan jarak yang berada di luar jangkauan Kinect memiliki error yang sangat kecil sehingga tidak dapat membahayakan *user* dalam mendeteksi halangan pada siang hari tanpa pencahayaan pada lorong yang seluruhnya solid.

- Pencahayaan : Lampu Menyala

Dari data di atas dapat diketahui bahwa error rata-rata dari pengukuran di atas mendapatkan nilai 1.85% dan disimpulkan bahwa

jarak yang masih termasuk jangkauan dari Kinect dan jarak yang berada di luar jangkauan Kinect memiliki error yang sangat kecil sehingga tidak dapat membahayakan *user* dalam mendeteksi halangan pada siang hari dengan pencahayaan pada lorong yang seluruhnya solid.

- Waktu : Pukul 19:00 WIB
  Tempat : Laboratorium B202 Teknik Elektro ITS
  Pencahayaan : Tidak ada Pencahayaan

Dari data di atas dapat diketahui bahwa error rata-rata dari pengukuran di atas mendapatkan nilai 0.927% dan disimpulkan bahwa jarak yang masih termasuk jangkauan dari Kinect dan jarak yang berada di luar jangkauan Kinect memiliki error yang sangat kecil sehingga tidak dapat membahayakan *user* dalam mendeteksi halangan pada malam hari tanpa pencahayaan pada lorong yang seluruhnya solid.

- Pencahayaan : Lampu Menyala

Dari data di atas dapat diketahui bahwa error rata-rata dari pengukuran di atas mendapatkan nilai 0.753% dan disimpulkan bahwa jarak yang masih termasuk jangkauan dari Kinect dan jarak yang berada di luar jangkauan Kinect memiliki error yang sangat kecil sehingga tidak dapat membahayakan *user* dalam mendeteksi halangan pada malam hari dengan pencahayaan pada lorong yang seluruhnya solid.

4.2 Pengujian Pengukuran Jarak di Dalam Lorong yang Tidak Seluruhnya Solid

Dengan menggunakan metode-metode yang sudah dijelaskan pada bagian disain dan implementasi, maka dilakukan pengujian terhadap lorong yang tidak seluruhnya solid pada siang hari, apakah sistem mampu menjaga *user* tetap berada dalam jarak aman untuk berjalan atau tidak. Dalam hal ini pengujian sudah dilakukan dengan dua jenis pencahayaan yang berbeda, yaitu menggunakan pencahayaan lampu dan tidak menggunakan cahaya lampu, kamera diletakkan di atas helm dan diarahkan ke bawah sebesar -20°, berikut data-data yang didapatkan dari hasil pengujian

Dari data di atas dapat diketahui bahwa error rata-rata dari pengukuran di atas mendapatkan nilai 1.048% dan disimpulkan bahwa jarak yang masih termasuk jangkauan dari Kinect dan jarak yang berada di luar jangkauan Kinect memiliki error yang sangat kecil sehingga tidak dapat membahayakan *user* dalam mendeteksi halangan pada siang hari tanpa pencahayaan pada lorong yang seluruhnya tidak solid.

- Pencahayaan : Lampu Menyala

Dari data di atas dapat diketahui bahwa error rata-rata dari pengukuran di atas mendapatkan nilai 0.594% dan disimpulkan bahwa jarak yang masih termasuk jangkauan dari Kinect dan jarak yang berada di luar jangkauan Kinect memiliki error yang sangat kecil sehingga tidak dapat membahayakan *user* dalam mendeteksi halangan pada siang hari dengan pencahayaan pada lorong yang seluruhnya tidak solid.

- Waktu : Pukul 19:00 WIB
  Tempat : Lorong Gedung B Lantai 2 Teknik Elektro ITS
  Pencahayaan : Tidak ada Pencahayaan

Dari data di atas dapat diketahui bahwa error rata-rata dari pengukuran di atas mendapatkan nilai 0.548% dan disimpulkan bahwa jarak yang masih termasuk jangkauan dari Kinect dan jarak yang berada di luar jangkauan Kinect memiliki error yang sangat kecil sehingga tidak dapat membahayakan *user* dalam mendeteksi halangan pada malam hari tanpa pencahayaan pada lorong yang seluruhnya tidak solid.

- Pencahayaan : Lampu Menyala

Dari data di atas dapat diketahui bahwa error rata-rata dari pengukuran di atas mendapatkan nilai 0.548% dan disimpulkan bahwa jarak yang masih termasuk jangkauan dari Kinect dan jarak yang berada di luar jangkauan Kinect memiliki error yang sangat kecil sehingga tidak dapat membahayakan *user* dalam mendeteksi halangan pada malam hari tanpa pencahayaan pada lorong yang seluruhnya tidak solid.

Dari data di atas dapat disimpulkan bahwa jarak yang masih termasuk jangkauan dari Kinect dan jarak yang berada di luar jangkauan Kinect memiliki error yang sangat kecil sehingga tidak dapat membahayakan *user* dalam mendeteksi halangan pada malam hari dengan pencahayaan atau tanpa pencahayaan pada lorong yang tidak seluruhnya solid.

4.3 Pengujian Pendeteksi Orang di Dalam Lorong yang Seluruhnya Solid

Dengan metode-metode yang sudah dijelaskan di atas pada bagian disain dan implementasi, maka dilakukan pengujian apakah sistem sudah mampu mendeteksi orang di dalam lorong yang seluruhnya solid pada siang hari untuk membantu *user* berinteraksi dengan lingkungan sekitarnya khususnya manusia. Pengujian dilakukan pada 2 buah pencahayaan yang berbeda yaitu, menggunakan pencahayaan lampu dan tidak menggunakan pencahayaan lampu, kamera diletakkan di atas helm dan diarahkan ke bawah sebesar -20°, berikut adalah data-data yang didapatkan dari hasil pengujian

- Waktu : Pukul 10:00 WIB
  Tempat : Laboratorium B202 Teknik Elektro ITS
  Pencahayaan : Tidak Ada Pencahayaan

Dari data di atas dapat disimpulkan bahwa untuk pendeteksian manusia pada siang hari di dalam lorong yang seluruhnya solid tanpa pencahayaan tidak dapat mendeteksi keberadaan manusia jika fitur objek manusia itu tidak masuk dalam area jangkauan *frame* dari Kinect arau bisa dibilang objek terlalu dekat dengan lensa kamera tersebut.

- Pencahayaan : Lampu Menyala

Dari data di atas dapat disimpulkan bahwa untuk pendeteksian manusia pada siang hari di dalam lorong yang seluruhnya solid dengan pencahayaan tidak dapat mendeteksi keberadaan manusia jika fitur objek manusia itu tidak masuk dalam area jangkauan *frame* dari Kinect bisa dibilang objek terlalu dekat dengan lensa kamera tersebut.

- Waktu : Pukul 20:00 WIB
  Tempat : Laboratorium B202 Teknik Elektro ITS
  Pencahayaan : Tidak Ada Pencahayaan

Dari data di atas dapat disimpulkan bahwa untuk pendeteksian manusia pada malam hari di dalam lorong yang seluruhnya solid tanpa pencahayaan tidak dapat mendeteksi keberadaan manusia jika jarak objek manusia itu tidak masuk dalam area jangkauan *frame* dari Kinect dan juga tidak mampu mendeteksi keberadaan manusia pada jarak tertentu apabila terdapat beberapa manusia yang berada lebih dekat dengan area jangkauan tersebut.

Dari data di atas dapat disimpulkan bahwa untuk pendeteksian manusia pada malam hari di dalam lorong yang seluruhnya solid dengan pencahayaan tidak dapat mendeteksi keberadaan manusia jika jarak objek manusia itu tidak masuk dalam area jangkauan *frame* dari Kinect dan juga tidak mampu mendeteksi keberadaan manusia pada jarak tertentu apabila terdapat beberapa manusia yang berada lebih dekat dengan area jangkauan tersebut.

4.4 Pengujian Pendeteksi Orang di Dalam Lorong yang Tidak Seluruhnya Solid pada Siang Hari

Dengan metode-metode yang sudah dijelaskan di atas pada bagian disain dan implementasi, maka dilakukan pengujian apakah sistem sudah mampu mendeteksi orang di dalam lorong yang tidak seluruhnya solid pada siang hari untuk membantu *user* berinteraksi dengan lingkungan sekitarnya khususnya manusia. Pengujian dilakukan pada 2 buah pencahayaan yang berbeda yaitu, menggunakan pencahayaan lampu dan tidak menggunakan pencahayaan lampu, kamera diletakkan di atas helm dan diarahkan ke bawah sebesar -20°, berikut adalah data-data yang didapatkan dari hasil pengujian

- Waktu : Pukul 10:00 WIB
  Tempat : Lorong Gedung B Lantai 2 Teknik Elektro ITS
  Pencahayaan : Tidak Ada Pencahayaan

Dari data di atas dapat disimpulkan bahwa untuk pendeteksian manusia pada siang hari di dalam lorong yang tidak seluruhnya tanpa pencahayaan solid dengan tidak dapat mendeteksi keberadaan manusia jika jarak objek manusia itu tidak masuk dalam area jangkauan *frame* dari Kinect tersebut dan juga apabila cahaya yang tertangkap oleh lensa Kinect terlalu terang sehingga mengganggu fokus dari Kinect itu sendiri.

Dari data di atas dapat disimpulkan bahwa untuk pendeteksian manusia pada siang hari di dalam lorong yang tidak seluruhnya solid dengan pencahayaan tidak dapat mendeteksi keberadaan manusia jika jarak objek manusia itu tidak masuk dalam area jangkauan *frame* dari Kinect tersebut dan juga apabila cahaya yang tertangkap oleh lensa dari Kinect terlalu terang sehingga mengganggu fokus dari Kinect itu sendiri.

Dari data di atas dapat disimpulkan bahwa untuk pendeteksian manusia pada malam hari di dalam lorong yang tidak seluruhnya solid tidak dapat mendeteksi keberadaan manusia jika jarak objek manusia itu tidak masuk dalam area jangkauan *frame* dari Kinect tersebut dan juga apabila manusia tersebut berada pada area yang gelap.

Dari data di atas dapat disimpulkan bahwa untuk pendeteksian manusia pada malam hari di dalam lorong yang tidak seluruhnya solid tidak dapat mendeteksi keberadaan manusia jika jarak objek manusia itu tidak masuk dalam area jangkauan *frame* dari Kinect tersebut dan juga apabila manusia tersebut berada pada area yang gelap.

4.5 Pengujian Pengaruh Gangguan Manusia Lain, Intensitas Cahaya, dan Jarak Tangkap Kamera terhadap Efektifitas Human Detection terhadap Target

Dengan metode-metode yang sudah dijelaskan di atas pada bagian disain dan implementasi, maka dilakukan pengujian apakah sistem sudah mampu mendeteksi manusia di dalam lorong yang seluruhnya solid dari posisi yang berbeda untuk membantu *user* berinteraksi dengan lingkungan sekitarnya khususnya manusia. Pengujian dilakukan pada 6 buah tingkat pencahayaan lampu, 10 jarak yang berbeda, dan adanya gangguan manusia lain yang menutupi tubuh target, penutupan tubuh target dibagi

menjadi 3 bagian yaitu, 1/3 bagian kiri tubuh target, 1/3 bagian tengah tubuh target, dan 1/3 bagian kanan tubuh target, berikut adalah data-data yang didapatkan dari hasil pengujian

Dari data diatas dapat diketahui bahwa eror pendeteksian target mendapatkan nilai 75%, hal ini dikarenakan gangguan manusia lain menghalangi kamera untuk mendeteksi fitur-fitur dari target sehingga target tidak dapat terdeteksi sama sekali apabila fitur dari badan target tertutupi meskipun hanya 1/3 bagian tubuhnya.

Dari data di atas dapat disimpulkan bahwa sistem dapat menangkap ada atau tidaknya manusia secara optimal di semua jarak pada nilai lumens antara 100-185 lux dan sistem tidak mampu menangkap ada tidaknya manusia pada nilai lumens

4.6 Pengujian Sistem Keseluruhan Pada Subjek yang Berbeda

Setelah didapatkan data-data mengenai pengukuran jarak dan deteksi manusia, maka dilakukan pengujian sistem secara keseluruhan apakah sistem ini bisa dipakai secara *universal* sehingga kedepannya dapat berguna bagi para penyandang tuna netra. Berikut adalah hasil dari pengujian sistem secara keseluruhan :

- Subjek 1 :

Dari data peta di atas dapat disimpulkan bahwa, untuk pendeteksian halangan dan manusia sudah berhasil dilakukan sistem serta perjalanan *user* terhindar dari bahaya tabrakan.

- Subjek 2 :

Dari data peta di atas dapat disimpulkan bahwa, untuk pendeteksian halangan dan manusia sudah berhasil dilakukan sistem serta perjalanan *user* terhindar dari bahaya tabrakan.

- Subjek 3 :

Dari data peta di atas dapat disimpulkan bahwa, untuk pendeteksian halangan dan manusia sudah berhasil dilakukan sistem serta perjalanan *user* terhindar dari bahaya tabrakan akan tetapi *user* tidak mampu mencapai tujuan hal ini dikarenakan sistem ini belum dilengkapi sistem navigasi.

- Subjek 4 :

Dari data peta di atas dapat disimpulkan bahwa, untuk pendeteksian halangan dan manusia sudah berhasil dilakukan sistem serta perjalanan *user* terhindar dari bahaya tabrakan meskipun *user* sempat berputar-putar untuk mencari jalan yang aman untuk dirinya.

- Subjek 5 :

Dari data peta di atas dapat disimpulkan bahwa, untuk pendeteksian halangan dan manusia sudah berhasil dilakukan sistem serta perjalanan *user* terhindar dari bahaya tabrakan.

4.7 Pengujian Kecepatan Berjalan Terhadap Ketepatan Sistem

Setelah dilakukan pengujian terhadap berbagai *user* kemudian dilakukan pengujian terhadap kecepatan berjalan masing-masing *user* terhadap ketepatan sistem dalam memandu *user* untuk menghindari halangan yang ada di depannya. Pada pengujian ini dihadirkan kembali 3 orang yang sudah pernah menggunakan sistem ini dan 2 orang yang belum pernah menggunakan sistem ini. Berikut adalah data-data yang didapat dari pengujian ini.

- Subjek 1 : subjek yang sudah pernah menggunakan sistem

  Dari data peta di atas, dapat disimpulkan bahwa subjek 1 lebih berhati-hati lagi dalam berjalan dengan cara mencari lagi jalan-jalan yang menurut program dapat dilewati.
- Subjek 2 : subjek yang belum pernah menggunakan sistem

Dari data peta di atas dapat disimpulkan bahwa subjek yang belum pernah menggunakan sistem ini berjalan sangat cepat dan kurang hati-hati, tetapi program masih dapat menginformasikan subjek bahwa ada halangan yang berada di depannya.

- Subjek 3 : subjek yang sudah pernah menggunakan sistem

Dari data peta di atas dapat disimpulkan bahwa subjek 3 sebagai subjek yang sudah pernah menggunakan sistem ini Nampak berjalan lebih hati-hati dan pelan-pelan sehingga dapat berjalan dengan selamat sampai tujuan atau target.

- Subjek 4 : subjek yang belum pernah menggunakan sistem

Dari data peta di atas dapat disimpulkan bahwa subjek 4 sebagai subjek yang belum pernah menggunakan sistem ini berjalan secara tidak beraturan dan langsung bergerak menuju tujuan atau target dengan selamat tanpa menabrak sesuatu apapun.

- Subjek 5 : subjek yang sudah pernah menggunakan sistem

Dari data peta di atas dapat disimpulkan bahwa subjek 5 sangat berhati-hati dalam mengambil keputusan kemana dia akan berjalan sehingga jumlah langkah dalam berjalan menjadi lebih banyak daripada yang lainnya dan subjek berhasil dengan selamat berjalan dari posisi awal hingga target atau tujuan.

BAB 5
KESIMPULAN DAN SARAN

Setelah tahap pengujian sistem, dilakukan pengujian sistem untuk beberapa user

5.1 Kesimpulan

Kesimpulan yang diperoleh dalam Tugas Akhir ini adalah :

1. Kesalahan rata-rata dari pengukuran jarak pada waktu kapanpun menghasilkan nilai 2.2%, dari data tersebut dapat disimpulkan bahwa sistem ini tidak membahayakan *user* dalam melakukan aktifitas berjalan.
2. Performa metode ini sebenarnya tidak memiliki delay, satu-satunya delay yang membuat metode ini melambat adalah penyampaian melalui suara.
3. Hasil pengujian diperoleh pada pengujian sistem secara keseluruhan untuk beberapa *user* didapatkan error sebesar 20%, dikarenakan perbedaan interpretasi dari masing-masing *user* untuk memilih jalan mana yang akan dipilih.
4. Hasil pengujian diperoleh pada pengujian deteksi manusia didapatkan error sebesar 20%, hal ini dikarenakan keterbatasan area jangkau dari *frame* Kinect dan besar *Frame Per Second* (FPS) yang dimiliki oleh Kinect.
5. Hasil pengujian diperoleh pada kecepatan berjalan terhadap ketepatan sistem didapatkan pengguna yang sudah pernah menggunakan sistem ini berjalan 1 langkah/detik sedangkan pengguna yang belum pernah menggunakan sistem ini berjalan 3 langkah/detik.

5.2 Saran

Beberapa saran yang penulis dapat berikan untuk pengembangan tugas akhir adalah sebagai berikut :

1. Sensor visual yang digunakan diharapkan lebih memiliki kualitas *Frame Per Second* (FPS) yang lebih baik.

2. Diharapkan metode penuntunan dilengkapi dengan rekonstruksi 3 Dimensi dan penggunaan kompas untuk memudahkan *user* dalam penentuan arah dan tujuan kemana akan berjalan.
3. Demi fleksibilitas pengguna *processing unit* dapat diganti dengan yang lebih ringkas contohnya seperti *Raspberry Pi* atau mini PC.

DAFTAR PUSTAKA

[1] Microsoft Developer Network Library. “Kinect for Windows Sensor Components and Specifications”. <URL: http://msdn.microsoft.com /en-us/library/jj131033.aspx> Desember, 2010.

[2] Izadi, Zahram. “KinectFusion: Real-time 3D Reconstruction and Interaction Using a Moving Depth Camera”. Microsoft Research. Cambridge University, 2011.

[3] S. Shao, “Mobility aids for the blind.” in *Electronic Devices for Rehabilitation*. New York: Wiley, 1985, pp. 79-100.

[4] Peter B. L. Meijer, “An Experimental System for Auditory Image Representations”, IEEE Trans. On Biomedical Eng., V01.39, N0.2 1992,pp.112-121

[5] Bradski, Gary dan Kaebler, Adrian. “*Learning OpenCV Computer Vision with the OpenCV Library”,* O’ REILLY. 2008.

[6] P. Viola and M. Jones, "Rapid Object Detection using a Boosted Cascade of Simple Features," in Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on, 2001, pp. I-511-I-518 vol.1.

[7] S. Ioffe and D. A. Forsyth, "Probabilistic Methods for Finding People," International Journal of Computer Vision, vol. 43, pp. 45-68, 2001.

LAMPIRAN

Berikut adalah listing program dari tugas akhir saya,

```
#include "stdafx.h"
#include <stdio.h>
#include <iostream>
#include <cv.h>                    // Main OpenCV library.
#include <cvaux.h>                 // Auxiliary (experimental) OpenCV
functions.
#include <highgui.h>          // OpenCV functions for files and graphical
windows.
#include <cxcore.h>                // Data structures and linear algebra
support.
#include "NuiApi.h"                // library sensor kinect
#include <sapi.h>
using namespace std;
static CvMemStorage* storage = 0;
static CvHaarClassifierCascade* cascade = 0;

int suara=0;int oke=0;
//IplImage *image = 0;
const int ab = (640/3),ac = 480;
const int ad = (640/3),ae = 480;
const int af = (640/3),ag = 480;
//const int am = 160,an = 480;
int add_remove_pt = 0;
int DMAA[ab][ac];int ax=0;int ay=0;
int DMAB[ad][ae];int bx=0;int by=0;
int DMAC[af][ag];int cx=0;int cy=0;
//int DMAD[am][an];int dx=0;
int deg=0,deg2=0,deg3=0;
ISpVoice * pVoice = NULL;

//-------------- deklarasi variabel kinect
INuiSensor*                     m_pNuiSensor;
HRESULT                                 hr;
INuiSensor*                     pNuiSensor;
NUI_IMAGE_FRAME                 imageFrame;
int                                             iSensorCount;
//------------------------------------ deklarasi variabel depth
```

```
IplImage*                                       depthh;
IplImage*                                       color;
IplImage  *frame_copy, *gray, *small_img = 0;
HANDLE              m_pColorStreamHandle;
HANDLE              m_hNextColorFrameEvent;
HANDLE              m_pDepthStreamHandle;
HANDLE                                          m_hNextDepthFrameEvent;
BYTE*                           m_depthRGBX;
static const int        cColorWidth  = 640;
static const int        cColorHeight = 480;
static const int        cDepthWidth  = 640;
static const int        cDepthHeight = 480;
static const int        cBytesPerPixel = 4;


HRESULT inisialisasi_kinect()
{
// Initialize the Kinect and specify that we'll be using depth
   hr = m_pNuiSensor-
>NuiInitialize(NUI_INITIALIZE_FLAG_USES_COLOR|NUI_INITIALIZ
E_FLAG_USES_DEPTH);
   if (SUCCEEDED(hr))
    {
// Create an event that will be signaled when depth data is available
      m_hNextColorFrameEvent = CreateEvent(NULL, TRUE, FALSE,
NULL);
                m_hNextDepthFrameEvent = CreateEvent(NULL,
TRUE, FALSE, NULL);
// Open a depth image stream to receive depth frames

hr = m_pNuiSensor-
>NuiImageStreamOpen(NUI_IMAGE_TYPE_COLOR,NUI_IMAGE_RES
OLUTION_640x480, 0, 2, m_hNextColorFrameEvent,
&m_pColorStreamHandle);
                if( FAILED( hr ) )
                {
                        cout<<"Could not open image stream
video"<<endl;
                        return hr;
                }
```

```
                hr = m_pNuiSensor->NuiImageStreamOpen(
NUI_IMAGE_TYPE_DEPTH, NUI_IMAGE_RESOLUTION_640x480, 0,
2, m_hNextDepthFrameEvent, &m_pDepthStreamHandle);
                if( FAILED( hr ) )
                {
                        cout<<"Could not open depth stream
video"<<endl;
                        return hr;
                }

        }
        return hr;
}

HRESULT initial()
{
        iSensorCount = 0;
        hr = NuiGetSensorCount(&iSensorCount);
        if (FAILED(hr))    return hr;
        // Look at each Kinect sensor
        for (int i = 0; i < iSensorCount; ++i)
  {
                // Create the sensor so we can check status, if we can't
create it, move on to the next
    hr = NuiCreateSensorByIndex(i, &pNuiSensor);
    if (FAILED(hr))         continue;
    // Get the status of the sensor, and if connected, then we can initialize it
    hr = pNuiSensor->NuiStatus();
    if (S_OK == hr)
    {
                        m_pNuiSensor = pNuiSensor;
       break;
    }
    // This sensor wasn't OK, so release it since we're not using it
    pNuiSensor->Release();
  }
        if (NULL != m_pNuiSensor)
  {
                hr = inisialisasi_kinect();
        }
        if (NULL == m_pNuiSensor || FAILED(hr))
```

```
    {
                cout << "No ready Kinect found!" <<endl;
                return E_FAIL;
    }
        else
        {
                cout << "Kinect Ready" <<endl;
                //-----------------------------------------creat variabel image
depth
                color = cvCreateImageHeader(cvSize(cColorWidth,
cColorHeight), IPL_DEPTH_8U, 4);
                depthh = cvCreateImageHeader(cvSize(cDepthWidth,
cDepthHeight),IPL_DEPTH_8U, cBytesPerPixel);
                // create heap storage for depth pixel data in RGBX
format
                m_depthRGBX = new
BYTE[cDepthWidth*cDepthHeight*cBytesPerPixel];
                cvNamedWindow("SHOW IMG",
CV_WINDOW_AUTOSIZE);
                cvNamedWindow("SHOW IMG_",
CV_WINDOW_AUTOSIZE);
                //cvSetMouseCallback("SHOW IMG", on_mouse, 0 );
        }
}

void get_collor()
{
        if (WAIT_OBJECT_0 ==
WaitForSingleObject(m_hNextColorFrameEvent, 0))
        {
                cascade =
(CvHaarClassifierCascade*)cvLoad("haarcascade_frontalface_alt.xml",0,0,0
);
                storage = cvCreateMemStorage(0);
                double scale = 1;
                hr = m_pNuiSensor-
>NuiImageStreamGetNextFrame(m_pColorStreamHandle, 0,
&imageFrame);
                if (FAILED(hr))    return;
                INuiFrameTexture * pTexture =
imageFrame.pFrameTexture;
```

```
                NUI_LOCKED_RECT LockedRect;
                // Lock the frame data so the Kinect knows not to modify
it while we're reading it
                pTexture->LockRect(0, &LockedRect, NULL, 0);
                // Make sure we've received valid data
                if (LockedRect.Pitch != 0)
                {
                        BYTE * pBuffer = (BYTE*) LockedRect.pBits;
                        cvSetData(color, pBuffer, LockedRect.Pitch);
                }
                cvFlip(color,color,1);
                //------------human detection---------------
                frame_copy=cvCloneImage(color);
                cvSmooth(color,frame_copy,CV_GAUSSIAN,3,3,0,0);
                gray = cvCreateImage( cvSize( color -> width, color ->
height), 8, 1 );
                small_img = cvCreateImage( cvSize( cvRound (color-
>width/scale), cvRound (color->height/scale)), 8, 1 );
                cvCvtColor( color, gray, CV_BGR2GRAY );
                cvResize( gray, small_img, CV_INTER_LINEAR );
                cvEqualizeHist( small_img, small_img );
                cvClearMemStorage( storage );
                deg = (ax-(depthh->width/2))/7;
                deg2 = (bx-(depthh->width/2))/7;
                deg3 = (cx-(depthh->width/2))/7;
                if( cascade )
                {
                        CvSeq* body = cvHaarDetectObjects(
small_img, cascade, storage,
                                1.1, 2, 0
                                //|CV_HAAR_FIND_BIGGEST_OBJECT
                                |CV_HAAR_DO_ROUGH_SEARCH
                                //|CV_HAAR_DO_CANNY_PRUNING
                                //|CV_HAAR_SCALE_IMAGE
                                ,
                                cvSize(100, 100) );

                                        for(int i = 0; i < (body ? body->total :
0); i++ )
                                        {
```

```
                                        CvRect* r =
(CvRect*)cvGetSeqElem( body, i );
                                        CvPoint center;
                                        int radius=80;
                                        center.x = cvRound((r->x +
r->width*0.5)*scale);
                                        center.y = cvRound((r->y +
r->height*0.5)*scale);
                                        //cvCircle(frame_copy,
center, radius, CV_RGB(255,0,0), 3, 8, 0);
                                        if(i+1>=1 && deg3<20 &&
deg>-20){
                                        if(suara==1)hr = pVoice-
>Speak(L"be careful, people detected at your front ", 0, NULL);
                                        cvRectangle( color,cvPoint(
center.x - 75, center.y - 50 ),cvPoint( center.x + 75 , center.y + 200
),cvScalar( 255, 0, 0, 0 ), 2, 0, 0 );

            //cvSaveImage("D:\\Kuliah\\KULIAH IMADUDDIN\\TA (Tugas
Akhir) Bismillah\\Tugas Akhir (Blind Guidance System)\\TA\\TA
KINECT\\TA_Kinect\\TA_Kinect\\Capture.jpg",color);
                                        printf("ada orang di depan");
                                        printf("derajat = %d\n",deg);
                                        printf("jumlah orang =
%d\n",i+1);

                                        }
                                        if(i+1>=1 && deg<-20 &&
deg>-45){
                                        if(suara==1)hr = pVoice-
>Speak(L"be careful, people detected on your front right ", 0, NULL);
                                        cvRectangle( color,cvPoint(
center.x - 75, center.y - 50 ),cvPoint( center.x + 75 , center.y + 200
),cvScalar( 255, 0, 0, 0 ), 2, 0, 0 );

            //cvSaveImage("D:\\Kuliah\\KULIAH IMADUDDIN\\TA (Tugas
Akhir) Bismillah\\Tugas Akhir (Blind Guidance System)\\TA\\TA
KINECT\\TA_Kinect\\TA_Kinect\\Capture.jpg",color);
                                        printf("ada orang di depan
kiri");
                                        printf("derajat = %d\n",deg);
```

```
                                        printf("jumlah orang =
%d\n",i+1);
                                        }
                                        if(i+1>=1 && deg3>20 &&
deg3<45){
                                        if(suara==1)hr = pVoice-
>Speak(L"be careful, people detected on your front left ", 0, NULL);
                                        cvRectangle( color,cvPoint(
center.x - 75, center.y - 50 ),cvPoint( center.x + 75 , center.y + 200
),cvScalar( 255, 0, 0, 0 ), 2, 0, 0 );

            //cvSaveImage("D:\\Kuliah\\KULIAH IMADUDDIN\\TA (Tugas
Akhir) Bismillah\\Tugas Akhir (Blind Guidance System)\\TA\\TA
KINECT\\TA_Kinect\\TA_Kinect\\Capture.jpg",color);
                                        printf("ada orang di depan
kanan");
                                        printf("derajat = %d\n",deg);
                                        printf("jumlah orang =
%d\n",i+1);
                                        }
                                    }

                    }
                    //cvShowImage("result",frame_copy);
                    cvShowImage("SHOW IMG_", color);
                    // We're done with the texture so unlock it
                    pTexture->UnlockRect(0);
                    ReleaseFrame:
                    // Release the frame
                    m_pNuiSensor-
>NuiImageStreamReleaseFrame(m_pColorStreamHandle, &imageFrame);
                    cvWaitKey(1);
            }
}

void get_depth()
{

            for(int x=0; x<(640/3); x++)    //This loops on the rows.
                    {
```

```
                    for(int y=0; y<=ac; y++) //This loops on the
columns
                    {
                                if (WAIT_OBJECT_0 ==
WaitForSingleObject(m_hNextDepthFrameEvent, 0))
                                {
                                        // Attempt to get the depth
frame
                                        hr = m_pNuiSensor-
>NuiImageStreamGetNextFrame(m_pDepthStreamHandle, 0,
&imageFrame);
                                        if (FAILED(hr))    return;
                                        BOOL nearMode = true;
                                        INuiFrameTexture*
pTexture;
                                        // Get the depth image pixel
texture
                                        hr = m_pNuiSensor-
>NuiImageFrameGetDepthImagePixelFrameTexture(m_pDepthStreamHand
le, &imageFrame, &nearMode, &pTexture);
                                        if(FAILED(hr))    goto
ReleaseFrame;
                                        NUI_LOCKED_RECT
LockedRect;
                                        // Lock the frame data so the
Kinect knows not to modify it while we're reading it
                                        pTexture->LockRect(0,
&LockedRect, NULL, 0);
                                        if (LockedRect.Pitch != 0)
                                        {
                                                // Get the min and
max reliable depth for the current frame
                                                int minDepth =
(nearMode ? NUI_IMAGE_DEPTH_MINIMUM_NEAR_MODE :
NUI_IMAGE_DEPTH_MINIMUM) >>
NUI_IMAGE_PLAYER_INDEX_SHIFT;
                                                int maxDepth =
(nearMode ? NUI_IMAGE_DEPTH_MAXIMUM_NEAR_MODE :
NUI_IMAGE_DEPTH_MAXIMUM) >>
NUI_IMAGE_PLAYER_INDEX_SHIFT;
```

```
                                                    BYTE * rgbrun =
m_depthRGBX;
                                                    const
NUI_DEPTH_IMAGE_PIXEL * pBufferRun = reinterpret_cast <const
NUI_DEPTH_IMAGE_PIXEL*>(LockedRect.pBits);
                                                    // end pixel is start
+ width*height - 1
                                                    const
NUI_DEPTH_IMAGE_PIXEL * pBufferEnd = pBufferRun +
(cDepthWidth * cDepthHeight)-1;
                                                    //----------hitung
distance depth
                                                    int selisih=0;
                                                    selisih =
pBufferEnd-pBufferRun;
                                                    //----------

                                                    while ( pBufferRun
< pBufferEnd )
                                                    {
                                                            USHORT
depth = pBufferRun->depth;
                                                            BYTE
intensity = static_cast<BYTE>(depth >= minDepth && depth <= maxDepth
? depth % 255 : 0);

        *(rgbrun++) = intensity;

        *(rgbrun++) = intensity;

        *(rgbrun++) = intensity;
                                                            ++rgbrun;

        ++pBufferRun;
                                                    }

                                                    pBufferRun =
pBufferRun - selisih;
                                                    //int posisi_pixel;
```

```
                                                    int posisi_pixel =
640*y + x; // posisi dalam koordinat x,y (10,10) => dalam index pixel =
640*y + x ; 640 = lebar frame
                                                    pBufferRun =
pBufferRun + posisi_pixel;
                                                    USHORT depth =
pBufferRun->depth;
                                                    DMAA[ab][ac] =
depth;
                                                    //system("cls");
                                                    USHORT * pBuff
= (USHORT*)m_depthRGBX;

            cvSetData(depthh,pBuff ,LockedRect.Pitch);
                                                    ax=x;
                                                    ay=y;

                                                }

                        cvFlip(depthh,depthh,1);
                        cvShowImage("SHOW IMG", depthh);

                        pTexture->UnlockRect(0);
                        pTexture->Release();

                        // Release the frame
                        m_pNuiSensor-
>NuiImageStreamReleaseFrame(m_pDepthStreamHandle, &imageFrame);
                        cvWaitKey(1);
                                            }
                                    }
                        }

            for(int x=(640/3); x<(640/3)*2; x++)    //This loops on the rows.
                        {
                                    for(int y=0; y<=ae; y++) //This loops on the
columns
                                    {
                                                if (WAIT_OBJECT_0 ==
WaitForSingleObject(m_hNextDepthFrameEvent, 0))
                                                {
```

```
                                        // Attempt to get the depth
frame
                                        hr = m_pNuiSensor-
>NuiImageStreamGetNextFrame(m_pDepthStreamHandle, 0,
&imageFrame);
                                        if (FAILED(hr))    return;
                                        BOOL nearMode = true;
                                        INuiFrameTexture*
pTexture;
                                        // Get the depth image pixel
texture
                                        hr = m_pNuiSensor-
>NuiImageFrameGetDepthImagePixelFrameTexture(m_pDepthStreamHand
le, &imageFrame, &nearMode, &pTexture);
                                        if(FAILED(hr))    goto
ReleaseFrame;
                                        NUI_LOCKED_RECT
LockedRect;
                                        // Lock the frame data so the
Kinect knows not to modify it while we're reading it
                                        pTexture->LockRect(0,
&LockedRect, NULL, 0);
                                        if (LockedRect.Pitch != 0)
                                        {
                                                // Get the min and
max reliable depth for the current frame
                                                int minDepth =
(nearMode ? NUI_IMAGE_DEPTH_MINIMUM_NEAR_MODE :
NUI_IMAGE_DEPTH_MINIMUM) >>
NUI_IMAGE_PLAYER_INDEX_SHIFT;
                                                int maxDepth =
(nearMode ? NUI_IMAGE_DEPTH_MAXIMUM_NEAR_MODE :
NUI_IMAGE_DEPTH_MAXIMUM) >>
NUI_IMAGE_PLAYER_INDEX_SHIFT;
                                                BYTE * rgbrun =
m_depthRGBX;
                                                const
NUI_DEPTH_IMAGE_PIXEL * pBufferRun = reinterpret_cast <const
NUI_DEPTH_IMAGE_PIXEL*>(LockedRect.pBits);
                                                // end pixel is start
+ width*height - 1
```

```
                                                    const
NUI_DEPTH_IMAGE_PIXEL * pBufferEnd = pBufferRun +
(cDepthWidth * cDepthHeight)-1;
                                                    //----------hitung
distance depth
                                                    int selisih=0;
                                                    selisih =
pBufferEnd-pBufferRun;
                                                    //----------

                                                    while ( pBufferRun
< pBufferEnd )
                                                    {
                                                            USHORT
depth = pBufferRun->depth;
                                                            BYTE
intensity = static_cast<BYTE>(depth >= minDepth && depth <= maxDepth
? depth % 255 : 0);

        *(rgbrun++) = intensity;

        *(rgbrun++) = intensity;

        *(rgbrun++) = intensity;
                                                            ++rgbrun;

        ++pBufferRun;
                                                    }

                                                    pBufferRun =
pBufferRun - selisih;
                                                    //int posisi_pixel;
                                                    int posisi_pixel =
640*y + x; // posisi dalam koordinat x,y (10,10) => dalam index pixel =
640*y + x ; 640 = lebar frame
                                                    pBufferRun =
pBufferRun + posisi_pixel;
                                                    USHORT depth =
pBufferRun->depth;
                                                    DMAB[ad][ae] =
depth;
```

```
                                                        //system("cls");
                                                        /*printf("posisi
ad,ae = %d , %d\n",x,y);
                                                        printf("jarak = %d
mm\n", depth);
                                                        printf("nilai jarak di
ad,ae = %d\n",DMAB[ad][ae]);*/

                                                        USHORT * pBuff
= (USHORT*)m_depthRGBX;

            cvSetData(depthh,pBuff ,LockedRect.Pitch);
                                                        bx=x;
                                                        by=y;
                                                    }
                        cvFlip(depthh,depthh,1);
                        cvShowImage("SHOW IMG", depthh);

                        pTexture->UnlockRect(0);
                        pTexture->Release();

                        // Release the frame
                        m_pNuiSensor-
>NuiImageStreamReleaseFrame(m_pDepthStreamHandle, &imageFrame);
                        cvWaitKey(1);
                                    }
                            }
                        }

for(int x=(640/3)*2; x<640; x++)    //This loops on the rows.
            {
            for(int y=0; y<=ag; y++) //This loops on the columns
                        {
                        if (WAIT_OBJECT_0 ==
WaitForSingleObject(m_hNextDepthFrameEvent, 0))
                                    {
// Attempt to get the depth frame
                                                hr = m_pNuiSensor-
>NuiImageStreamGetNextFrame(m_pDepthStreamHandle, 0,
&imageFrame);
                                                if (FAILED(hr))    return;
```

```
                                        BOOL nearMode = true;
                                        INuiFrameTexture*
pTexture;
// Get the depth image pixel texture
                                        hr = m_pNuiSensor-
>NuiImageFrameGetDepthImagePixelFrameTexture(m_pDepthStreamHand
le, &imageFrame, &nearMode, &pTexture);
if(FAILED(hr))     goto ReleaseFrame;
NUI_LOCKED_RECT LockedRect;
// Lock the frame data so the Kinect knows not to modify it while we're
reading it
                                        pTexture->LockRect(0,
&LockedRect, NULL, 0);
                                        if (LockedRect.Pitch != 0)
                                        {
// Get the min and max reliable depth for the current frame
                                            int minDepth =
(nearMode ? NUI_IMAGE_DEPTH_MINIMUM_NEAR_MODE :
NUI_IMAGE_DEPTH_MINIMUM) >>
NUI_IMAGE_PLAYER_INDEX_SHIFT;
                                            int maxDepth =
(nearMode ? NUI_IMAGE_DEPTH_MAXIMUM_NEAR_MODE :
NUI_IMAGE_DEPTH_MAXIMUM) >>
NUI_IMAGE_PLAYER_INDEX_SHIFT;
                                            BYTE * rgbrun =
m_depthRGBX;
                                            const
NUI_DEPTH_IMAGE_PIXEL * pBufferRun = reinterpret_cast <const
NUI_DEPTH_IMAGE_PIXEL*>(LockedRect.pBits);
// end pixel is start + width*height - 1
                                            const
NUI_DEPTH_IMAGE_PIXEL * pBufferEnd = pBufferRun +
(cDepthWidth * cDepthHeight)-1;
                                            //----------hitung
distance depth
                                            int selisih=0;
                                            selisih =
pBufferEnd-pBufferRun;
//----------
                                            while ( pBufferRun
< pBufferEnd )
```

```
                                                    {
                                                        USHORT
depth = pBufferRun->depth;
                                                        BYTE
intensity = static_cast<BYTE>(depth >= minDepth && depth <= maxDepth
? depth % 255 : 0);

            *(rgbrun++) = intensity;

            *(rgbrun++) = intensity;

            *(rgbrun++) = intensity;
                                                        ++rgbrun;

            ++pBufferRun;
                                                    }

                                                    pBufferRun =
pBufferRun - selisih;
                                                    //int posisi_pixel;
                                                    int posisi_pixel =
640*y + x; // posisi dalam koordinat x,y (10,10) => dalam index pixel =
640*y + x ; 640 = lebar frame
                                                    pBufferRun =
pBufferRun + posisi_pixel;
                                                    USHORT depth =
pBufferRun->depth;
                                                    DMAC[af][ag] =
depth;
                                                    //system("cls");
                                                    /*printf("posisi
af,ag = %d , %d\n",x,y);
                                                    printf("jarak = %d
mm\n", depth);
                                                    printf("nilai jarak di
af,ag = %d\n",DMAC[af][ag]);*/

                                                    USHORT * pBuff
= (USHORT*)m_depthRGBX;

            cvSetData(depthh,pBuff ,LockedRect.Pitch);
```

```
                                                    cx=x;
                                                    cy=y;
                                                }
                    cvFlip(depthh,depthh,1);
                    cvShowImage("SHOW IMG", depthh);

                    pTexture->UnlockRect(0);
                    pTexture->Release();
                    ReleaseFrame:
                    m_pNuiSensor-
>NuiImageStreamReleaseFrame(m_pDepthStreamHandle, &imageFrame);
                    cvWaitKey(1);
                                }
                            }
                    }



        //=============================================
==========
}


int _tmain(int argc, _TCHAR* argv[])
{

   if (FAILED(::CoInitialize(NULL)))
      return FALSE;

   HRESULT hr = CoCreateInstance(CLSID_SpVoice, NULL,
CLSCTX_ALL, IID_ISpVoice, (void **)&pVoice);
   if( SUCCEEDED( hr ) )
   {
      suara=1;
                    if(suara==1)hr = pVoice->Speak(L"welcome to VoGVI",
0, NULL);

   }
```

```
            // inisialisasi sensor kinect

            if(suara==1)hr = pVoice->Speak(L"finding a clear way", 0,
NULL);

            hr = initial();
            for(;;)
            {

                        get_collor();
                        get_depth();
//----------------proses guiding----------------------

//----------------mencari jalan lurus-----------------

                        if (DMAB[ad][ae]>1000 && DMAB[ad][ae]<6500 &&
oke==0){
                                                                    oke=0;

            printf("ada jalan di ad,ae \n");
                                                                    printf("x =
%d, y = %d \n",bx,by);

            printf("jarak = %d\n",DMAB[ad][ae]);

            if(suara==1)hr = pVoice->Speak(L"you may walk straight", 0,
NULL);
                                                                    oke=1;
                                                                    }

//----------------deteksi halangan di depan------------

                        if (DMAB[ad][ae]>350 && DMAB[ad][ae]<1000 &&
oke==0 ){
            oke=0;
            printf("ada halangan di ad,ae \n");                     printf("x =
%d, y = %d \n",bx,by);

            printf("jarak = %d\n",DMAB[ad][ae]);
```

```
            if(suara==1)hr = pVoice->Speak(L"hold on, find another way", 0,
NULL);
                                                                    oke=1;
                                                                    }
//-------------------ketika ada lorong yang tidak semuanya solid------------
                        if (DMAB[ad][ae]>7000 && oke==0){
                                                                    oke=0;

            printf("tidak ada jalan di ad,ae \n");
                                                                    printf("x =
%d, y = %d \n",bx,by);

            printf("jarak = %d\n",DMAB[ad][ae]);

            if(suara==1)hr = pVoice->Speak(L"hold on, no more way", 0,
NULL);
                                                                    oke=1;
                                                                    }
                        //-------------------mencari jalan ketika ada halangan di
depan-------------

                        if (DMAB[ad][ae]>50 && DMAB[ad][ae]<1000 &&
DMAA[ab][ac]>1000 && DMAA[ab][ac]<2000 && ay==240 && ay<480
&& oke==0){
                                                                    oke=0;
                                                                    deg = (ax-
(depthh->width/2))/7;
                                                                    if (deg < -
35){

            if(suara==1)hr = pVoice->Speak(L"there is a way on your left", 0,
NULL);

            system("cls");

            printf("deg = %d\n",deg);

            printf("ada jalan di ab,ac \n");
                                                                    printf("x =
%d, y = %d \n",ax,ay);
```

BIODATA PENULIS

Muhammad Reza Imaduddin dilahirkan di Surabaya 06 September 1992. Anak tunggal dari pasangan Bambang Istiono dan Endang Susilowati. Penulis menyelesaikan pendidikan dasar di SDI Al-Azhar 11 Surabaya, Surabaya dilanjutkan dengan pendidikan menengah di SMPN 2 Surabaya dan SMAN 16 Surabaya. Pada tahun 2010, penulis memulai pendidikan di jurusan Teknik Elektro, Fakultas Teknologi Industri, Institut Teknologi Sepuluh Nopember (ITS) Surabaya. Selama kuliah penulis aktif membantu penyelenggaran kegiatan dan aktif sebagai asisten laboratorium Elektronika Dasar dan praktikum Elektronika pada semester ganjil 2013-2014.

Email :

reza.imaduddin@yahoo.com
muhammadrezaimaduddin@gmail.com