Institut
Teknologi
Sepuluh Nopember

TUGAS AKHIR – EE184801

RANCANG BANGUN ALAT BANTU NAVIGASI UNTUK TUNANETRA MENGGUNAKAN SENSOR RGB-D

Julius Sintara
NRP 07111640000100

Dosen Pembimbing
Dr. Ir. Hendra Kusuma, M.Eng.Sc.
Muhammad Attamimi, B.Eng., M.Eng., Ph.D.

DEPARTEMEN TEKNIK ELEKTRO
Fakultas Teknologi Elektro dan Informatika Cerdas
Institut Teknologi Sepuluh Nopember
Surabaya 2020

ITS
Institut
Teknologi
Sepuluh Nopember

TUGAS AKHIR – EE184801

RANCANG BANGUN ALAT BANTU NAVIGASI UNTUK TUNANETRA MENGGUNAKAN SENSOR RGB-D

Julius Sintara
NRP 07111640000100

Dosen Pembimbing
Dr. Ir. Hendra Kusuma, M.Eng.Sc.
Muhammad Attamimi, B.Eng., M.Eng., Ph.D.

DEPARTEMEN TEKNIK ELEKTRO
Fakultas Teknologi Elektro dan Informatika Cerdas
Institut Teknologi Sepuluh Nopember
Surabaya 2020

ITS
Institut
Teknologi
Sepuluh Nopember

FINAL PROJECT – EE184801

DESIGN OF NAVIGATION AID DEVICE FOR THE VISUALLY IMPAIRED USING RGB-D SENSOR

Julius Sintara
NRP 07111640000100

Supervisors
Dr. Ir. Hendra Kusuma, M.Eng.Sc.
Muhammad Attamimi, B.Eng., M.Eng., Ph.D.

ELECTRICAL ENGINEERING DEPARTMENT
Faculty of Intelligent Electrical and Informatics Technology
Institut Teknologi Sepuluh Nopember
Surabaya 2020

PERNYATAAN KEASLIAN
TUGAS AKHIR

Dengan ini saya menyatakan bahwa isi sebagian maupun keseluruhan Tugas Akhir saya dengan judul "Rancang Bangun Alat Bantu Navigasi untuk Tunanetra Menggunakan Sensor RGB-D" adalah benar-benar hasil karya intelektual mandiri, diselesaikan tanpa menggunakan bahan-bahan yang tidak diijinkan dan bukan merupakan karya pihak lain yang saya akui sebagai karya sendiri.

Semua referensi yang dikutip maupun dirujuk telah ditulis secara lengkap pada daftar pustaka. Apabila ternyata pernyataan ini tidak benar, saya bersedia menerima sanksi sesuai peraturan yang berlaku.

Surabaya, Juni 2020

Julius Sintara
NRP 07111640000100

RANCANG BANGUN ALAT BANTU NAVIGASI UNTUK TUNANETRA MENGGUNAKAN SENSOR RGB-D

TUGAS AKHIR

Diajukan Guna Memenuhi Sebagai Persyaratan
Untuk Memperoleh Gelar Sarjana Teknik

Pada

Bidang Studi Elektronika
Departemen Teknik Elektro
Fakultas Teknologi Elektro dan Informatika Cerdas
Institut Teknologi Sepuluh Nopember

Menyetujui:

Dosen Pembimbing I

Dr. Ir. Hendra Kusuma, M.Eng.Sc.
NIP 196409021989031003

SURABAYA
JULI, 2020

RANCANG BANGUN ALAT BANTU NAVIGASI UNTUK TUNANETRA MENGGUNAKAN SENSOR RGB-D

TUGAS AKHIR

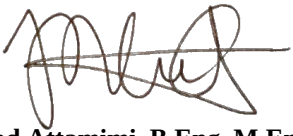
Diajukan Guna Memenuhi Sebagai Persyaratan
Untuk Memperoleh Gelar Sarjana Teknik

Pada

Bidang Studi Elektronika
Departemen Teknik Elektro
Fakultas Teknologi Elektro dan Informatika Cerdas
Institut Teknologi Sepuluh Nopember

Menyetujui:

Dosen Pembimbing II



Muhammad Attamimi, B.Eng, M.Eng, Ph.D.
NIP 198503272019031006

SURABAYA
JULI, 2020

Rancang Bangun Alat Bantu Navigasi untuk Tunanetra Menggunakan Sensor RGB-D

Nama : Julius Sintara
Pembimbing : 1. Dr. Ir. Hendra Kusuma, M.Eng.Sc.
2. Muhammad Attamimi, B.Eng., M.Eng., Ph.D.

ABSTRAK

Penyandang tunanetra dalam kesehariannya memiliki kesulitan untuk berjalan di tempat yang asing bagi mereka. Kesulitan ini terutama dalam hal spasial, yaitu memperkirakan medan jalan dan menghindari rintangan yang menghalangi langkah mereka. Agar penyandang tunanetra dapat berjalan mandiri, maka mereka membutuhkan alat bantu navigasi.

Alat bantu navigasi ini dirancang menggunakan sensor RGB-D (*Red Green Blue + Depth*) untuk menangkap informasi visual di sekitar penggunanya. Informasi visual ini kemudian diolah dan diubah dalam bentuk informasi audio stereo sehingga dapat diinformasikan kepada tunanetra melalui indera pendengarannya. Alat bantu ini juga dirancang agar mudah dibawa dan digunakan (*portable*), nyaman digunakan sehari-hari tanpa menganggu keseharian (*comfortable*), dan mudah digunakan sehingga penyandang tunanetra dapat menyesuaikan diri dengan mudah dan cepat (*easy to use*). Selain itu informasi yang didapatkan dari sensor RGB-D memungkinkan untuk pengembangan dan penambahan fitur lain sesuai dengan kebutuhan tunanetra.

Dari rancangan *wearable device*, alat dipasangkan pada subjek tunanetra dan mendapatkan penilaian yang tinggi untuk tingkat *portability*, *comfortability*, dan *ease of use* dari alat bantu navigasi ini. Dari segi fungsionalitas alat, didapatkan hasil pengujian subjek tunentra dapat mendeteksi posisi dari rintangan di depan mereka dengan akurasi 92.47% dan memperkirakan jarak terhadap objek tersebut dengan MAE sebesar 0.8. Untuk pengujian dalam bernavigasi, dengan menggunakan alat ini subjek tunanetra dapat berhenti sebelum terjadi tabrakan dengan rintangan di depannya, dan melewati celah di antara dua buah rintangan yang cukup untuk dilewatinya.

Kata kunci: alat bantu navigasi, audio stereo, kamera kedalaman, teknologi asistif, tunanetra

Halaman ini sengaja dikosongkan

Design of Navigation Aid Device for the Visually Impaired using RGB-D Sensor

Author : Julius Sintara
Supervisors : 1. Dr. Ir. Hendra Kusuma, M.Eng.Sc.
2. Muhammad Attamimi, B.Eng., M.Eng., Ph.D.

ABSTRACT

People with visual impairment in their daily lives face difficulties to walk in unfamiliar places to them. This problem mainly caused by their lack of spatial awareness, i.e. the ability to estimate the distance between themselves and their surroundings. In order for visually impaired people to navigate independently, an effective navigation aid is required.

The proposed navigation aid device utilizes depth camera to collect visual information of surrounding objects. Then, it represents the obtained visual data into stereophonic sound to notify the user directly through an audio device. The aid device is designed to be portable, comfortable, and easy to use. It can further be developed and upgraded to suit the needs of visually impaired users.

Designed to be wearable, this proposed device was tested and received good score in portability, comfortability, and ease of use. The subjects were able to detect the position of obstacles in front of them with 92.47% accuracy, and could also estimate the distance of the object with Mean Absolute Error of 0.8. Examination on their navigation ability indicated that the subjects could stop before collision with an object and manoeuvre through the gap between two parallel obstacles.

Keywords: assistive technology, depth camera, navigation aid, stereo audio, visual impairment

Halaman ini sengaja dikosongkan

KATA PENGANTAR

Hidup adalah sebuah proses pembelajaran dan pendewasaan yang tidak akan pernah berhenti. Di setiap tahap dalam kehidupan kita, akan ada banyak perubahan dan tantangan tersendiri agar kita sebagai individu semakin berkembang dan bermanfaat. Begitu juga selama penulisan laporan tugas akhir ini, banyak sekali proses pembelajaran dan pendewasaan yang dialami oleh penulis. Kesulitan dan hambatan semakin membawa penulis untuk bisa lebih baik lagi di tahap-tahap kehidupan yang selanjutnya.

Sebagai manusia yang lemah dan rentan, penyelesaian laporan tugas akhir ini adalah hal yang mustahil tanpa kehadirat Tuhan di dalam setiap proses pembuatan tugas akhir ini. Penulis ingin mengucapkan terima kasih sebesar-besarnya terhadap setiap pihak yang membantu penyelesaian tugas akhir ini, mulai dari dosen pembimbing Dr. Hendra Kusuma dan Muhammad Attamimi, Ph.D. yang memberikan bimbingan dan arahan dalam setiap proses pengerjaan tugas akhir ini, keluarga terdekat penulis yang selalu memberikan dukungan moral dan spiritual kepada penulis terutama pada masa-masa penulis ingin menyerah, teman-teman dekat penulis yang senantiasa menemani, memberikan banyak saran, dan mendukung secara moral di saat-saat tersulit, subjek pengujian tugas akhir ini yang telah rela meluangkan waktu dan tenaga untuk mencoba dan memberi banyak saran mulai, serta semua orang yang memberikan dukungan dan bantuan bagi penulis baik secara langsung dan tidak langsung, yang tidak dapat disebutkan satu per satu.

Tiada gading akar pun jadi, begitu juga dalam seluruh proses dan hasil penyelesaian tugas akhir ini. Penulis meminta maaf atas segala kekurangan dan kesalahan dalam laporan tugas akhir ini. Semoga tugas akhir ini dapat bermanfaat bagi orang banyak dan membantu untuk penelitian yang akan datang.

Surabaya, Juni 2020

Penulis

Halaman ini sengaja dikosongkan

DAFTAR ISI

Halaman ini sengaja dikosongkan

DAFTAR GAMBAR

DAFTAR TABEL

Halaman ini sengaja dikosongkan

BAB I
PENDAHULUAN

1.1 Latar Belakang

Di dunia, diperkirakan terdapat 285 juta orang yang memiliki gangguan penglihatan dengan 13.68% dari mereka menderita kebutaan total (*blind*), dan sisanya menderita *low vision* [1]. Tunanetra berdasarkan tingkat gangguan penglihatannya dapat digolongkan menjadi 2 jenis, yaitu orang dengan kebutaan total (*blind*) dan orang dengan *low vision*. [2]. Dalam kesehariannya penyandang tunanetra menghadapi berbagai kesulitan, terutama dalam bernavigasi karena ketidakmampuannya dalam mengamati lingkungan di sekitarnya. Indera penglihatan merupakan indera yang sangat penting untuk melakukan navigasi, mengetahui lingkungan dan mengidentifikasi objek di sekitar kita, serta memperkirakan lokasi dari objek di jarak pandang kita [3]. Hal ini tentunya tidak dapat dilakukan oleh penyandang tunanetra sehingga mereka harus menggunakan indera-indera yang lain untuk menghasilkan persepsi spasial.

Di Indonesia sendiri, infrastruktur dan fasilitas-fasilitas bagi penyandang tunanetra sendiri masih belum mendukung. Dibandingkan dengan negara lain, fasilitas untuk membantu penyandang tunanetra masih sangat minim sekali. Banyak tempat-tempat umum di Indonesia yang tidak dilengkapi dengan ubin tekstur pemandu (*tactile paving*) [4]. Begitu juga dengan fasilitas-fasilitas umum seperti bis dan kereta api, bantuan bagi penyandang tunanetra juga kurang diimplementasikan sehingga penyandang tunanetra sangat kesulitan untuk melakukan perjalanan dengan mandiri tanpa bantuan orang lain.

Dengan perkembangan tekonologi yang semakin pesat, sudah seharusnya kita dapat membantu penyandang tunanetra dalam mengatasi keterbatasannya tersebut. Informasi visual yang tidak mereka dapatkan, dengan bantuan teknologi, seharusnya dapat disampaikan menggunakan indera mereka yang lain.

Dari kebutuhan-kebutuhan tersebut, diperlukan sebuah alat bantu navigasi bagi penyandang tunanetra yang mudah dibawa dan digunakan oleh mereka (*portable*), nyaman digunakan sehari-hari tanpa menganggu keseharian mereka (*comfortable*), dan yang terakhir mudah digunakan sehingga penyandang tunanetra dapat menyesuaikan diri dengan alat tersebut (*easy to use*). Alat navigasi juga diharapkan dapat dikembangkan

dan ditambah fiturnya sesuai kebutuhan, sehingga penyandang tunanetra tidak perlu berganti-ganti atau menggunakan lebih dari satu alat bantu navigasi. Pada topik tugas akhir yang diajukan, diimplementasikan sensor berupa kamera RGB-D (*Red Green Blue + Depth*) yang dilengkapi dengan IMU (*Inertial Measurement Unit*) untuk menangkap citra warna serta kedalaman untuk mengetahui rintangan di sekitar penyandang tunanetra. Informasi visual akan diolah dan diubah menjadi bentuk informasi audio ke penyandang tunanetra. Representasi visual dengan audio dilakukan dalam frekuensi tertentu dan stereofonik untuk menghasilkan representasi spasial.

1.2 Permasalahan

Perumusan masalah pada tugas akhir ini adalah:

1. Bagaimana cara mengambil informasi visual di sekitar penyandang tunanetra dari kamera RGB-D yang ada.
2. Bagaimana mengolah informasi dari kamera RGB-D untuk mengetahui rintangan dan posisinya
3. Bagaimana cara merepresentasikan posisi rintangan yang ada dalam bentuk informasi audio
4. Bagaimana rancangan alat bantu navigasi *wearable* yang *portable*, *comfortable*, dan *easy to use*.

1.3 Tujuan

Tujuan dari tugas akhir ini adalah:

1. Pengembangan alat bantu tunanetra untuk bernavigasi dan berjalan mandiri tanpa tertabrak rintangan.
2. Pengembangan alat bantu tunanetra *wearable* yang *portable* (mudah dibawa dan tidak membebani), *comfortable* (nyaman dan tidak mengganggu ketika digunakan), dan *easy to use* (mudah digunakan sehari-hari)
3. Pengembangan alat bantu tunanetra yang memungkinkan untuk pengembangan berbagai fitur sesuai kebutuhan tunanetra.

1.4 Batasan Masalah

Batasan masalah dalam tugas akhir ini adalah:

1. Sensor yang digunakan adalah kamera Intel® RealSense™ D435i.
2. Objek rintangan yang dideteksi berupa halangan dengan ketinggian lebih tinggi atau hampir sejajar dengan ketinggian sensor.

3. Karena sensor menggunakan *Active Stereo IR*, objek yang dapat memantulkan cahaya dengan sempurna (cermin) menyebabkan kesalahan pembacaan dan tidak dapat dideteksi.

1.5 Metodologi

Langkah-langkah dalam pengerjaan tugas akhir ini adalah:

1. **Studi Literatur**

   Pada tahap studi literatur, dikumpulkan dan dikaji teori-teori yang mendukung tugas akhir ini. Literatur yang digunakan bersumber dari buku, jurnal, paper, artiket, datasheet, serta spesifikasi yang diterbitkan dan dipublikasi oleh institusi pendidikan, perusahaan perangkat, badan pemerintahan, serta institusi lainnya yang terpercaya. Literatur mengenai penelitian yang dilakukan pada tunanetra mengenai kesulitan yang dialami, kemampuan dalam bernavigasi, persepsi spasial, dan psikologinya. Selain itu juga mengenai spesifikasi dan cara kerja dari perangkat yang digunakan (Intel® RealSense™, NVIDIA® Jetson™), proses pengambilan dan pengolahan data dengan bahasa pemrograman Python beserta berbagai API yang mendukung, serta berbagai penelitian alat bantu navigasi tunanetra yang sudah dikembangkan.

2. **Perancangan Alat**

   Pada tahap perancangan alat, dirancang dan dibuat alat bantu tunanetra yang terdiri dari sensor RGB-D berupa Intel® RealSense™, Processing Unit (CPU) berupa NVIDIA® Jetson™, konektor untuk audio, dan *wearable support*. Rancangan dari alat berupa sensor RGB-D yang menghadap ke arah depan pengguna, kemudian processing unit (CPU) yang berada dikemas sehingga dapat digunakan sehari-hari. *Wearable support* diharapkan dapat digunakan dengan nyaman dan tanpa menganggu keseharian penggunanya. Konektor audio diletakkan di tempat yang mudah untuk diakses dan diberi transduser suara (*speaker*) seperti *headphone* atau *earphone*.

3. **Proses Pengambilan dan Pengolahan Data**

   Pada tahap ini, diambil data berupa informasi visual dengan kedalaman (RGB-D) dari sensor RGB-D dan data IMU dari sensor IMU yang digunakan. Pengambilan dan pengolahan data dilakukan oleh CPU yang terpasangi kamera warna dan kedalaman. Informasi visual dan IMU kemudian diolah agar dapat diproses

lebih lanjut untuk mengetahui rintangan yang ada di sekitar penggunanya. Pengolahan gambar kedalaman memegang peranan penting dalam proses selanjutnya.

4. **Deteksi Rintangan dan Posisinya**

   Pada tahap ini, data informasi visual dan IMU yang telah diolah akan dideteksi objek-objek yang ada. Akan diidentifikasi pula objek mana yang berpotensi menjadi rintangan bagi penggunanya dan objek mana yang bukan merupakan rintangan. Setelah diketahui objek yang merupakan rintangan, selanjutnya dicari letak dari objek tersebut. Informasi letak dari objek tersebut kemudian diinformasikan kepada penggunanya

5. **Representasi Informasi dalam Bentuk Audio**

   Pada tahap ini, informasi letak dari rintangan yang telah diolah akan dikonversi dalam bentuk informasi audio. Suara yang dihasilkan sederhana dan dinamis dengan frekuensi tertentu dan juga stereofonik (kanan dan kiri). Konsep yang digunakan adalah seperti pemindaian berputar. Informasi letak rintangan akan diinformasikan seperti berputar dengan sumbu vertikal, dengan frekuensi yang menggambarkan jarak antara pengguna dengan objek di depannya. Untuk ketinggian vertikal, digunakan data dari IMU yang diarahkan oleh penggunanya untuk mengetahui objek rintangan seperti di bawah dan di atas. Metode ini akan dikembangkan dan disesuaikan dengan kenyamanan dan kemudahan penggunanya sehingga memungkinkan untuk pergantian konsep sesuai dengan kebutuhan.

6. **Pengujian Alat pada Penyandang Tunanetra**

   Pada tahap ini, alat yang telah dirancang dan dibuat diujikan kepada penyandang tunanetra. Penyandang tunanetra yang menggunakan alat ini akan dihadapkan pada sebuah tempat yang asing bagi mereka yang terdapat rintangan dan kemudian mereka akan berjalan dengan rute tertentu untuk mencapai suatu lokasi. Alat ini akan diujikan pada lingkungan indoor dan juga outdoor untuk mengetahui kehandalannya. Pengguna dari alat ini yang diujikan adalah penyandang tunanetra, untuk mendapatkan banyak masukan terhadap alat bantu navigasi ini. Pertama, pengguna akan disurvey mengenai kenyamanannya dalam menggunakan alat tersebut, serta kemudahan dalam penggunaan alat tersebut setelah beberapa waktu menggunakan alat tersebut. Kemudian dalam melakukan navigasi, akan dilihat apakah

pengguna dapat mengetahui dan menghindari rintangan yang ada sehingga pengguna tidak tertabrak objek atau rintangan di depannya. Setelah itu pengguna akan diminta memberikan komentar, kritik, dan masukan mengenai pengalaman menggunakan alat bantu navigasi ini.

7. **Analisa dan Evaluasi**

   Hasil pengambilan data dari uji coba pada penyandang tunanetra kemudian akan dianalisa keakuratan dan tingkat kebermanfaatannya. Hasil survey dari kenyamanan dan kemudahan dalam penggunaan alat juga digunakan sebagai evaluasi dari rancangan alat navigasi. Akan dievaluasi pula apakah alat navigasi dapat membantu tunanetra dalam mengidentifikasi rintangan yang ada di sekitarnya dan dapat menghindari rintangan tersebut.

8. **Penyusunan Laporan**

   Penyusunan laporan dilaksanakan bersamaan dengan tahap-tahap yang lain. Isi dari laporan sesuai dengan format tugas akhir, meliputi pendahuluan, studi literatur, perancangan dan pembuatan, pengujian dan analisa, dan penutup.

1.6 Sistematika Penulisan

Sistematika dari penulisan laporan tugas akhir ini adalah:

- **BAB I PENDAHULUAN**

  Pada Bab 1 Pendahuluan berisi hal-hal meliputi latar belakang, rumusan masalah, tujuan, metodologi, sistematika penulisan, serta relevansi dan menfaat

- **BAB II STUDI LITERATUR**

  Bab ini berisi teori-teori penunjang serta literatur dari berbagai sumber antara lain buku, jurnal, paper, artiket, datasheet, serta spesifikasi yang diterbitkan dan dipublikasi oleh institusi pendidikan, perusahaan perangkat, badan pemerintahan, serta institusi lainnya

- **BAB III PERANCANGAN SISTEM**

  Pada bab ini dijelaskan perancangan sistem dari tugas akhir ini yaitu dari sisi perangkat keras maupun perangkat lunak dari alat bantu navigasi untuk tunanetra

- **BAB IV PENGUJIAN DAN ANALISIS**

  Bab ini berisi hasil jadi alat, pengujian alat, serta komparasi-komparasi antar parameter yang diatur, beserta dengan analisa dari hasil tersebut

- **BAB V PENUTUP**

  Pada bab terakhir ini, berisi kesimpulan yang diperoleh dari penelitian tugas akhir serta saran untuk pengembangan yang lebih lanjut.

1.7 Relevansi dan Manfaat

Hasil dari tugas akhir ini diharapkan dapat bermanfaat bagi tunanetra untuk dapat mengakses tempat-tempat umum yang jarang dikunjungi dan mengurangi keraguan tunanetra untuk dapat mengeksplorasi berbagai tempat.dengan independen tanpa bergantung pada bantuan orang lain.

BAB II
STUDI LITERATUR

2.1 Gangguan Penglihatan

Gangguan penglihatan (*visual impairment*) merupakaan keadaan hilangnya kemampuan mata (indera penglihatan) manusia untuk melihat pada tingkat dimana kemampuan visual penderitanya terbatasi secara signifikan atau dapat dikatakan tidak dapat melihat. Hal ini dapat disebabkan oleh penyakit, trauma, bawaan (*congenital*), atau kondisi degeneratif yang tidak dapat diperbaiki secara konvensional pada umumnya seperti dengan kacamata atau pengobatan.

Berdasarkan *International Classification of Diseases* (ICD) oleh organisasi kesehatan dunia (WHO) [5], tingkat gangguan dalam penglihatan dikategorikan seperti pada **Tabel 2.1**.

Tabel 2.1 Kategori tingkat gangguan penglihatan

Kategori	**Tingkat Ketajaman Penglihatan**	
	Lebih buruk dari:	**Sama atau lebih baik dari:**
Tanpa Gangguan Penglihatan		6/12 5/10 (0.5) 20/40
Gangguan Penglihatan Ringan	6/12 5/10 (0.5) 20/40	6/18 3/10 (0.3) 20/70
Gangguan Penglihatan Sedang	6/18 3/10 (0.3) 20/70	6/60 1/10 (0.1) 20/200
Gangguan Penglihatan Berat	6/60 1/10 (0.1) 20/200	3/60 1/20 (0.05) 20/400
Kebutaan	3/60 1/20 (0.05) 20/400	Tidak Ada Persepsi Cahaya Sama Sekali

Tingkat ketajaman penglihatan yang ditampilkan pada **Tabel 2.1** adalah berdasarkan bagan Snellen [6]. Angka pertama pada bagan Snellen merupakan jarak (feet) antara mata telanjang dengan bagan Snellen dan

angka kedua merupakan jarak (feet) dimana mata telanjang masih mampu membaca huruf pada barisan tersebut dengan jelas

Di Indonesia sendiri, kata tunanetra memiliki makna yang menimbulkan berbagai penafsiran yang berbeda. Menurut Kamus Besar Bahasa Indonesia (KBBI), kata tunanetra sendiri memiliki artu tidak dapat melihat atau buta. Akan tetapi, jika kita melihat dari arti gabungan kata pada KBBI, kata tuna memiliki arti luka; rusak dan kurang; tidak memiliki, sedangkan kata netra berarti mata. Jika digabungkan, maka tunanetra sendiri memiliki makna kerusakan mata dan kurang atau tidak dapat melihat. Orang dengan kelainan atau cacat indera penglihatan dapat digolongkan menjadi dua, yaitu buta total (*blind*) dan *low vision*. [7]

Pada penelitian ini, untuk menyamakan persepsi terminologi, kata tunanetra yang dimaksud adalah orang dengan gangguan penglihatan atau cacat indera penglihatan, sehingga tidak terbatas pada orang dengan kebutaan saja. Target dari penelitian ini merupakan tunanetra sesuai dengan deskripsi tersebut, sehingga tidak dispesifikkan mengenai tingkat ketajaman penglihatan minimum dari pengguna alat ini.

2.2 Kesulitan yang Dihadapi Tunanetra

Keterbatasan yang dimiliki oleh tunanetra menyebabkan mereka mengalami berbagai kesulitan dalam kehidupan sehari-hari. Tantangan terbesar bagi tunanetra, terutama bagi orang dengan kebutaan adalah dalam bernavigasi di berbagai tempat. Bagi mereka, berkeliling di tempat kediaman mereka adalah hal yang mudah karena mereka telah mengetahui posisi dari barang-barang serta lokasi dari ruangan-ruangan yang ada, dan mereka terbiasa bernagivasi dan berpindah tempat antar ruangan sehari-harinya. Hal ini berbeda dengan tempat-tempat lain yang asing bagi mereka. Pada tempat umum, biasanya terdapat bantuan bagi tunanetra yaitu berupa ubin tekstur (*tactile tiles / paving*). Akan tetapi, tidak semua tempat umum memiliki fasilitas ini, terutama di Indonesia dimana fasilitas untuk penyandang disabilitas termasuk tunanetra masih sangat minim [4]. Hal ini menyebabkan kesulitan yang cukup besar ketika penyandang tunanetra harus mengunjungi tempat-tempat umum dan asing bagi mereka.

Permasalahan lain yang dihadapi adalah orang yang terlalu membantu. Tentunya saling membantu adalah hal yang baik di masyarakat, tetapi orang yang terlalu membantu juga menjadi permasalahan bagi tunanetra secara psikologis [3]. Banyak orang yang mengabaikan dan kurang membantu para disabilitas, tetapi beberapa

malah merasa senang dalam membantu penyandang disabilitas. Membantu orang dengan disabilitas memang hal yang baik, akan tetapi terkadang orang-orang yang ingin membantu tersebut lupa untuk menanyakan apakah penyandang disabilitas tersebut benar membutuhkan pertolongan atau tidak. Penyandang tunanetra mungkin terlihat kesulitan melakukan sesuatu karena mereka melakukan hal tersebut dengan sangat lambat (dari perspektif orang normal), akan tetapi hal tersebut wajar bagi mereka tanpa perlu membutuhkan pertolongan dari orang lain.

Kemudian yang paling penting dari kesulitan-kesulitan tersebut adalah untuk mendapatkan independensi (kemandirian). Seseorang dengan kebutaan dapat menjalani kehidupannya secara mandiri dengan hal-hal adaptif yang dirancang khusus bagi mereka. Banyak alat-alat bantu yang dikhususkan bagi penyandang tunanetra untuk dapat hidup secara mandiri, sehingga mereka dapat memperoleh independensinya. [8]

2.3 Teknologi Asistif

Teknologi asistif merupakan perangkat atau alat yang dapat membantu, menyesuaikan, dan atau merehabilitasi orang dengan disabilitas atau orang yang lanjut usia. Orang dengan disabilitas seringkali menemui berbagai kesuiltan dalam aktivitasnya sehari-hari dan untuk dapat menjalani kehidupannya secara mandiri. Aktivitas-aktivitas ini meliputi pembuangan kotoran atau penggunaan toilet, mobilitas untuk berpindah tempat dan bernavigasi, makan, minum, mandi, berpakaian, bersolek, dan lain sebagainya. Teknologi asistif diharapkan dapat membantu para penyandang disabilitas untuk dapat melakukan aktivitas-aktivitas tersebut secara mandiri dan tanpa bantuan orang lain. [9]

2.3.1 Teknologi Adaptif

Seringkali terjadi kebingungan antara teknologi asistif dan teknologi adaptif. Kedua hal ini merupakan istilah yang berbeda, akan tetapi teknolog adaptif sendiri merupakan bagian dari teknologi asistif. Teknologi adaptif dapat didefinisikan sebagai objek atau sistem yang dirancang secara khusus untuk meningkatkan atau mempertahankan kemampuan orang dengan disabilitas. Teknologi adaptif ini merupakan tambahan dari teknologi yang sudah ada, yang dapat membantu orang dengan disabilitas, sehingga teknologi adaptif jarang digunakan oleh orang normal atau tanpa disabilitas. Contoh dari teknologi adaptif adalah aplikasi pembaca layer, dimana teknologi ini diterapkan pada teknologi komputer yang sudah ada untuk memberikan kemudahan bagi penyandang tunanetra. [10]

2.3.2 Teknologi untuk orang dengan gangguan mobilitas

Target dari teknologi ini adalah untuk penyandang disabilitas dengan cacat fisik yang menyebabkan mereka kesulitan untuk bermobilitas dalam arti untuk memindahkan tubuhnya dari satu tempat ke tempat lain. Contoh dari teknologi ini adalah kursi roda, yang digunakan oleh orang dengan kelumpuhan kaki untuk dapat berpindah tempat dengan mandiri. Contoh lainnya adalah bagian tubuh prostetik, seperti kaki buatan atau tangan buatan.

2.3.3 Teknologi untuk orang dengan gangguan penglihatan

Target dari teknologi ini adalah untuk penyandang tunanetra. Orang dengan gangguan penglihatan memiliki kesulitan seperti tidak dapat membaca, maka teknologi asistif yang dapat digunakan yaitu *screen reader* pada computer untuk dapat mengubah informasi visual di layer menjadi informasi suara dari teks yang ada pada layar tersebut. Teknologi lainnya adalah seperti *refreshable braille display* [11] sebagai *interface* untuk orang dengan tunanetra dengan mesin atau komputer, dan juga *braille keyboard* untuk mempermudah orang dengan gangguan penglihatan untuk mengendalikan komputer dan mengetik.

Alat bantu navigasi pada penelitian ini ditujukan untuk tunanetra, sehingga dapat digolongkan sebagai teknologi untuk orang dengan gangguan penglihatan (*visual impairments*). Tentunya tunanetra dalam target penelitian ini terbatas pada ketidakmampuan untuk melihat dengan jelas, dan penelitian ini ditujukan untuk membantu permasalahan tersebut dikhususkan untuk bernavigasi. Untuk kemampuan dalam bermobilisasi dalam arti untuk memindahkan tubuhnya bukan merupakan target dan tujuan dari penelitian ini.

2.3.4 Penelitian dan teknologi yang sudah ada pada alat bantu navigasi tunanetra

Teknologi asistif untuk tunanetra ada banyak dan bermacam-macam. Banyak riset dan penelitian yang menghasilkan berbagai teknologi untuk membantu tunanetra dalam bernavigasi. Beberapa teknologi terkemuka masih dalam tahap riset dan belum dapat diperoleh secara bebas, tetapi banyak pula teknologi yang sudah diproduksi secara masal dan mudah didapatkan, seperti navigasi berbasis aplikasi dan *Global Positioning* System (GPS) pada ponsel cerdas [12] hingga perangkat tongkat cerdas seperti *white cane* [13]. Selain itu, penelitian ini juga sebagai kelanjutan dari penelitian yang dilakukan oleh Ichsan [14] dimana pada penelitian tersebut digunakan Kamera ZED sebagai sensor

RGB-D dan masih menggunakan komputer jinjing (*laptop*) sebagai CPU. Pada penelian ini digunakan sensor dan CPU yang berbeda, serta metode pengubahan informasi kedalaman ke suara yang berbeda yang diharapkan dapat lebih baik dan meningkatkan akurasi dan kebermanfaatan alat. Selain itu pada penlitian ini rancangan perangkat keras juga dirancang agar lebih *portable*, *comfortable*, dan *easy to use*.

2.4 Persepsi Spasial Tunanetra

Berbagai sistem penginderaan diperlukan untuk menghasilkan representasi spasial dan untuk bernavigasi. Indera penglihatan (visual) memiliki peranan yang penting dalam pengembangan representasi spasial. Indera penglihatan digabungkan dengan indra pendengaran, penciuman, dan sistem sensorik lainnya untuk bernavigasi dengan lancar [15]. Akan tetapi, pada penyandang tunanetra, mereka tidak dapat menggunakan indera penglihatannya dengan baik, sehingga perlu adanya mekanisme kompensasi untuk menghasilkan kemampuan spasial dalam melakukan navigasi [16].

Sebuah studi dilakukan pada tunanetra untuk mengetahui persepsi mereka tentang bentuk dari suara (audio), yaitu dengan memperdengarkan sebuah bentuk lewat suara, dan kemudian mereka harus mengenali serta mencontohkan bentuk tersebut dengan rute berjalan [17]. Penyandang tunanetra dikelompokkan menjadi dua: tunanetra sejak awal (bawaan) dan tunanetra yang kehilangan penglihatannya pada umur tertentu (bukan bawaan). Hasil dari studi tersebut menyebutkan ada tiga kecenderungan. Pertama adalah penyandang tunanetra bawaan cenderung melakukan kompresi bentuk dalam bernavigasi. Hal ini membuktikan bahwa mereka memiliki tendensi untuk mencari rute tercepat ketimbang membayangkan bentuk dari rute yang akan diambil. Kedua yaitu kesulitan mereka dalam mengenali rangsangan suara yang kompleks. Terakhir, yaitu kesulitan dalam mencontohkan bentuk yang diinginkan. Sebagai contoh, salah satu penyandang tunanetra bawaan diminta untuk mempersepsikan bentuk persegi lewat rute jalannya, tetapi mereka mencontohkan rute berbentuk lingkaran dalam navigasinya.

Untuk menghasilkan alat bantu navigasi dengan representasi visual dalam bentuk audio yang mudah digunakan, diperlukan penginformasian audio yang mudah dipahami dan tidak kompleks. Informasi ini diharapkan dapat membantu penyandang tunanetra dalam membayangkan ruangan (spasial) dan melakukan navigasi dengan baik. Kekurangan mereka dalam membayangkan bentuk dan spasial [17]

tentunya menjadi masalah dan tantangan tersendiri, sehingga diperlukan representasi yang sederhana, mudah, dan dapat disesuaikan dengan baik.

2.5 Suara

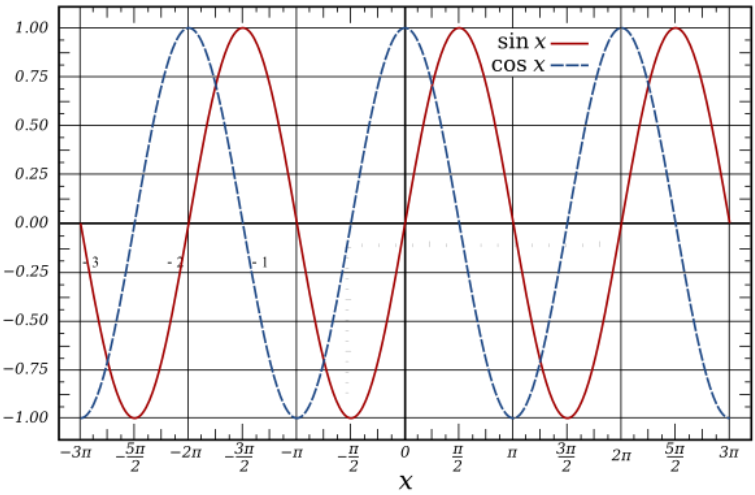
Di dalam fisika, suara adalah getaran yang merambat sebagai gelombang akustik melalui medium transmisi seperti udara, cairan, atau benda padat. Dalam fisiologi dan psikologi, suara merupakan penerimaan gelombang dan persepsi oleh otak. Gelombang akustik yang memiliki frekuensi antara 20 Hz hingga 20 kHz disebut audio frekuensi, yang menimbulakn persepsi pendengaran pada manusia. Suara diatas frekuensi 20 kHz disebut ultrasonik dan tidak dapat didengar manusia. Begitu pula dengan frekuensi di bawah 20 Hz yang disebut infrasonik.

Sebuah suara terjadi akibat adanya getaran yang berosilasi. Osilasi ini terjadi pada medium secara longitudinal maupun transversal. Suara ditransmisikan melalui udara, plasma, dan cairan sebagai gelombang longitudinal, atau disebut gelombang kompresi. Pada benda padat, suara dapat ditransmisikan baik dalam gelombang longitudinal maupun gelombang transversal. Gelombang longitudinal merupakan gelombang dari perubahan tekanan yang berosilasi terhadap tekanan ekuilibrium, sehingga tercipta *compression* dan *rarefaction*. Pada gelombang transversal melalui benda padat gelombang dari osilasi *shear* dan *stress* pada sudut tertentu kearah perambatan gelombang.

Gelombang suara yang ada dapat berupa gelombang yang kompleks. Tapi selombang yang kompleks sekalipun dapat direpresentasikan dalam kombinasi gelombang yang berbentuk sinusoid dengan frekuensi dan amplitude yang berbeda-beda. Parameter dari gelombang tersebut adalah dalam bentuk frekuensi (atau panjang gelombang), amplitudo (atau tekanan suara atau intensitas), cepat rambat suara, dan arah rambat suara.

2.5.1 Parameter Gelombang Sinusoid

Sebelum merepresentasikan gelombang yang kompleks, kita harus mengetahui parameter dari sebuah gelombang sinusoid sederhana. Berikut pada **Gambar 2.1** meripakan grafik sinusoid (sinus dan cosinus). dan persamaan gelombang sinusoid.



Gambar 2.1 Gelombang sinusoid (sinus dan cosinus)

$$y(t) = A\sin(2\pi ft + \varphi) = A\sin(\omega t + \varphi) \qquad \textbf{(2.1)}$$

dimana:

A : amplitudo (puncak deviasi dari fungsi terhadap
f : frekuensi (jumlah osilasi yang terjadi dalam satu detik)
ω : kecepatan angular
(kecepatan perubahan sudut dalam radian per detik)
φ : fase (besar sudut dalam radian saat fungsi berada pada t=0)

Parameter ini penting untuk diketahui untuk membentuk sebuah gelombang. Kumpulan dari gelombang dengan frekuensi dan amplitudo yang berbeda dapat membentuk suatu harmoni yang nyaman didengar oleh manusia. Demikian pula kombinasi yang tidak tepat juga dapat menyebabkan suara tidak nyaman didengar dan terasa aneh saat didengarkan.

2.5.2 Domain Waktu Diskrit dan Sampling Rate

Gelombang kita dengarkan adalah gelombang dalam domain waktu kontinu. Sebuah gelombang dengan domain waktu kontinu berarti gelombang tersebut memiliki nilai pada setiap waktu untuk nilai waktu dalam bilangan real. Namun, dalam dunia digital, tidak mungkin untuk menyimpan atau menghasilkan semua nilai dari gelombang pada semua waktu untuk waktu dalam bilangan real. Diperlukan adanya pengubahan

dari domain waktu kontinu ke domain waktu diskrit, kemudian dilakukan kuantisasi dan koding untuk mengubah nilai ke dalam bentuk biner.

Pertama, untuk mengubah sebuah fungsi kontinu dalam waktu diskrit diperlukan *sampling*. *Sampling* merupakan proses pengambilan nilai dari fungsi dalam dalam rentang waktu tertentu secara berkala. Waktu antar pengambilan nilai / sampel disebut *sampling time* (T_s), sehingga banyaknya pengambilan sampel dalam satu detik dikenal dengan istilah *sampling frequency* (f_s). Fungsi yang telah dilakukan *sampling* akan menjadi:

$$s(nT_s), n \in \mathbb{Z} \quad \textbf{(2.2)}$$

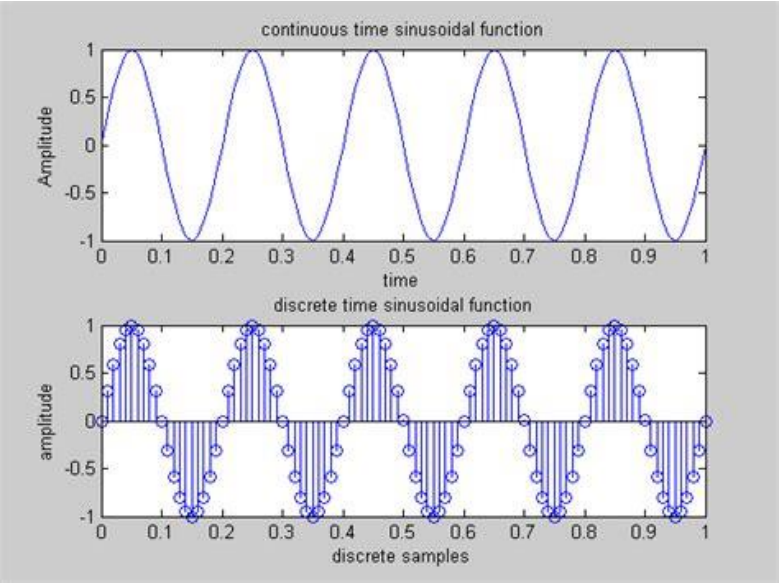
Nilai dari f_s dapat bernilai berapa saja, akan tetapi akan pemilihan nilai f_s sangat berperan penting dalam mencegah distorsi pada suara, atau lebih dikenal dengan *aliasing*. Untuk mencegah hal tersebut, nilai f_s dipilih sesuai dengan teorema *sampling* Nyquist – Shannon, dimana dikatakan "Jika pada sebuah fungsi $x(t)$ tidak terdapat frekuensi yang lebih besar dari B Hz, maka fungsi tersebut dapat ditetapkan dengan memberi ordinatnya sebuah barisan titik dengan jeda $1/2B$ detik antar titik." [18]. Sehingga untuk mencegah *aliasing* pada suara yang dapat kita dengarkan, yaitu 20 Hz hingga 20 kHz, f_s minimum yang harus dipenuhi adalah dua kali dari frekuensi tertinggi yaitu 40 kHz.

Setelah dilakukan sampling, maka gelombang sinusoid semula akan menjadi dalam waktu diskrit dengan persamaan:

$$Y(n) = A \sin\left(2\pi f \frac{n}{f_s} + \varphi\right) \quad \textbf{(2.3)}$$

Berikut pada **Gambar 2.2** perbandingan antara gelombang sinusoid dalam waktu kontiu dan dalam waktu diskrit setelah dilakukan *sampling*.

Dalam dunia audio, ada beberapa *sampling rate* yang biasa digunakan. Pada awal perkembangan transmisi suara, *bandwidth* menjadi hal yang penting dalam pertimbangan pemilihan *sampling rate*. Semakin tinggi *sampling rate*, maka *bandwidth* juga akan semakin lebar dalam transmisi sehingga biaya akan semakin mahal. Sehingga *sampling rate* yang biasa digunakan adalah 8 kHz dimana banyak terjadi *aliasing* pada suara, tetapi cukup untuk percakapan manusia pada umumnya tanpa *sibilance*, biasanya digunakan pada telepon, *walkie-talkie*, dan alat komunikasi untuk bercakap-capap lainnya. Kemudian untuk *sampling rate* yang umum digunakan adalah 44.1 kHz, yang tertama kali digunakan



Gambar 2.2 Perbandingan gelombang sinusoid waktu kontinu dan waktu diskrit

pada CD (*compact disk*) dan menjadi standart umum yang cukup banyak digunakan termasuk dalam MPEG-1 audio (VCD, SVCD, MP3). Untuk nilai f_s yang lebih tinggi biasa digunakan untuk audio HD (*high definition*).

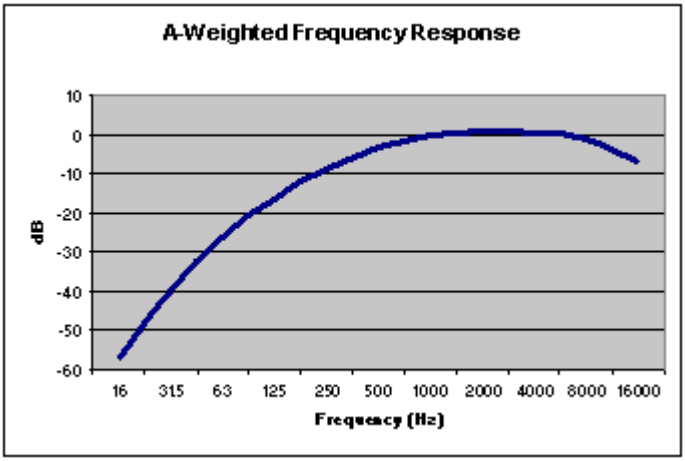
Pada penelitian ini, untuk meproduksi suara digunakan *sampling rate* 44.1 kHz karena cukup untuk semua frekuensi yang dapat didengarkan oleh manusia tanpa terjadi *aliasing*.

2.5.3 Pendengaran Manusia

Seperti dijelaskan sebelumnya, manusia pada umumnya dapat mendengarkan suara dengan frekuensi antara 20 Hz hingga 20 kHz secara maksimum. Setiap individu memiliki kepekaan terhadap frekuensi yang berbeda-beda, tetapi biasanya seiring dengan bertambahnya usia maka kepekaan telinga manusia terhadap frekuensi akan berkurang. Hal ini berkenaan dengan kemampuan pendengaran manusia yang absolut. Namun, bagaimanapun juga telinga kita kurang peka terhadap frekuensi yang terlalu rendah atau terlalu tinggi. Pada umumnya telinga manusia

memliki respon yang lebih besar pada frekuensi antara 500 Hz hingga 8 kHz saja.

Salah satu permodelan untuk mengukur level tekanan suara sesuai standard internasional IEC 61672:2003 [19] adalah A-*weighting*. Penggunaan filter A-*weighting* dilakukan untuk menghitung tingkat kekerasan suara terhadap persepsi telinga manusia, yang kurang sensitive pada frekuensi yang terlalu rendah dan terlalu tinggi. Selain A-*weighting* ada juga B-, C-, D-, dan Z- *weightings*. Untuk grafik respon frekuensi A-*weighting* dalam dB(A) ditunjukkan pada **Gambar 2.3**.



Gambar 2.3 Respon frekuensi A-*weighting*

Fungsi dari A-*weighting* yang mendekati persepsi pendengaran manusia terhadap frekuensi suara dapat dimodelkan [19]:

$$R_A(f) = \frac{12194^2 f^4}{(f^2 + 20.6^2)\sqrt{(f^2 + 107.7^2)(f^2 + 737.9^2)}(f^2 + 12194^2)} \quad \textbf{(2.4)}$$

$$A(f) = 20\log_{10}\big(R_A(f)\big) - 20\log_{10}\big(R_A(1000)\big) \approx 20\log_{10}\big(R_A(f)\big) + 2.00 \quad \textbf{(2.5)}$$

$R_A(f)$ diaplikasikan pada spektrum amplitudo dari level suara yang belum dilakukan *weighting*. Offset berfungsi untuk memastikan normalisasi ke 0 dB pada 1000 Hz.

2.5.4 Audio Digital

Dalam audio digital, perlu diketahui beberapa parameter yang penting dalam menerima dan menyimpan suara maupun membuat dan mengeluarkan suara. Terlepas dari berbagai bentuk kompresi audio dalam dunia digital, ada hal-hal umum yang perlu dipahami dalam audio digital.

Pertama adalah format audio. Ada tiga format audio secara umum, yaitu audio tanpa kompresi, audio dengan kompresi *lossless*, dan audio dengan kompresi *lossy*. Audio tanpa kompresi contohnya WAV, AIFF, dan AU, dimana audio tidak dikompresi sehingga ukuran file audio cenderung lebih besar. Audio dengan kompresi *lossless* seperti FLAC, APE, WV, dan M4A dapat menyimpan audio dalam ukuran yang lebih kecil tanpa adanya informasi yang hilang. Terakhir adalah audio dengan kompresi *lossy* seperti MP3 dan AAC dimana ukuran file jauh lebih kecil namun dengan konsekuensi adanya penurunan kualitas audio.

Selanjutnya hal penting dalam audio digital adalah *channel*. *Channel* merupakan banyaknya suara yang ada dalam suatu file audio, dimana satu *channel* berarti hanya ada satu suara. Pada pemutar audio pada umumnya, digunakan dua *channel* untuk suara kiri dan suara kanan (audio stereofonik). Lebih dari dua *channel* biasa digunakan pada pemutar audio dengan konfigurasi khusus, seperti pada bioskop atau *home theatre*.

Seperti dijelaskan pada domain waktu diskrit, pemilihan *sample rate* sangat penting dalam audio digital. *Sample rate* akan menentukan banyaknya sampel dalam satu detik sehingga dalam pengeluaran suara tidak terjadi perbedaan waktu yang menyebabkan frekuensi yang didengarkan berubah.

Kemudian *frames per buffer* atau biasa disebut *chunk*. Dalam memutar atau menerima audio, perangkat digital biasanya mengirim informasi suara atau sampel dalam bentuk *buffer*. Jumlah dari sampel dalam satu *buffer* ini yang dikenal dengan istilah *chunk*. Sampel atau *frame* merupakan suatu nilai yang jumlahnya tergantung dengan *channel*. Jika ada 2 *channel*, maka dalam 1 *frame* akan ada dua nilai.

2.5.5 Suara Stereofonik

Suara stereofonik atau lebih dikenal dengan stereo adalah sebuah metode reproduksi suara yang menciptakan ilusi berupa persepsi suara multi-arah. Hal ini dilakukan dengan dua atau lebih *channel* audio yang independent dengan konfigurasi dua atau lebih *speaker* (seperti *earphone*, *headphone*, dan sebagainya) sedemikian rupa untuk menciptakan persepsi pendengaran dari berbagai arah, layaknya pendengaran alami manusia.

Sistem suara stereo dapat dibagi menjadi dua. Pertama yaitu suara stereo natural. Suara stereo natural ini ditangkap bersamaan dengan berbagai fenomena suara alami seperti *ambience* dan lain-lain oleh banyak mikrofon (*array of michophones*). Suara yang direkam dari tiap mikrofon kemudian direproduksi ulang pada lebih dari satu *speaker* untuk menciptakan perasaan mendengar sesuai dengan aslinya.

Kedua adalah suara stereo buatan atau *pan-pot*, dimana suara yang merupakan 1 suara (mono) saja yang ditangkap. Suara ini kemudian diproses secara digital dan diatur amplitudonya sedemikian rupa untuk membuat kesan stereo seperti aslinya. Kontrol untuk mengatur amplitudo dari suara ini disebut juga *pan-pot* (*panoramic potentiometer*). Dengan kombinasi beberapa *pan-pot* dari satu suara, akan dihasilkan suara stereo buatan yang seperti asli.

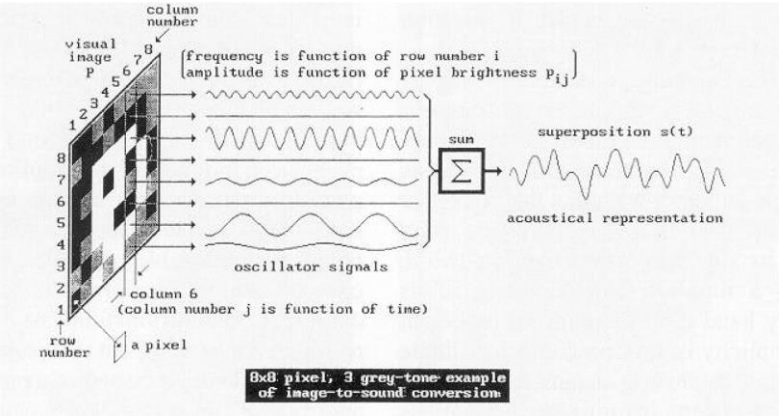
2.6 Representasi Visual dengan Audio

Dalam menyampaikan informasi visual kepada penyandang tunanetra, berbagai metode dapat dilakukan. Metode yang sering digunakan adalah dengan informasi suara (audio) dan haptic. Informasi haptic sendiri dapat diberikan pada tunanetra lewat berbagai bentuk, seperti *refreshable braille* [20], sabuk getar [21], dan lain sebagainya. Kelebihan dari haptic adalah dapat dirasakan oleh bagian tubuh tertentu saja, sehingga indera peraba yang lainnya masih dapat digunakan dengan baik. Kendati demikian, informasi dari haptic terkadang tidak tersampaikan dengan baik karena berbagai faktor, seperti rangsangan dari luar akibat berada pada kendaraan yang bergoyang, dan lain sebagainya.

Salah satu metode yang lainnya adalah dengan informasi audio. Informasi audio ini memiliki kekurangan yaitu mengharuskan penggunanya untuk menggunakan perangkat aktuator suara seperti *speaker*, *headphone*, *earphone*, dan sebagainya. Selain itu saat mendengarkan informasi audio, indera pendengaran dari pengguna hanya terfokus pada informasi audio yang diberikan, sehingga penggunanya akan sulit menggunakan indera pendengarannya untuk hal lain seperti berbincang-bincang dengan orang lain. Meskipun demikian, informasi yang diberikan melalui suara akan lebih jelas dan mudah dipahami, sehingga meminimalisir kesalahan informasi. Selain itu, indera pendengaran manusia dapat menerima informasi suara yang kompleks dan bermacam-macam.

Sebuah eksperimen mengenai representasi visual dengan audio pernah dilakukan pada tahun 1992 [22]. Pada eksperimen tersebut,

dilakukan pengkonversian dari gambar menjadi pola audio untuk merepresentasikan gambar tersebut. Pertama-tama, gambar diubah ke resolusi 64x64 piksel dengan nilai warna 4 bit (16 tingkat warna *grayscale*). Dari informasi visual ini, dilakukan konversi ke bentuk audio dengan frekuensi dan amplitude tertentu tiap pikeselnya, seperti pada Gambar 1 untuk gambar 8x8 piksel dan 3 tingkat warna *grayscale*. Informasi ini kemudian disatukan dan diperdengarkan untuk membayangkan gambar semula. Keterbatasan dari metode ini adalah kemampuan tiap individu tentang persepsi suara yang berbeda-beda. Metode ini juga masih terus dikembangkan dalam berbagai produk [23] dan penelitian [14].



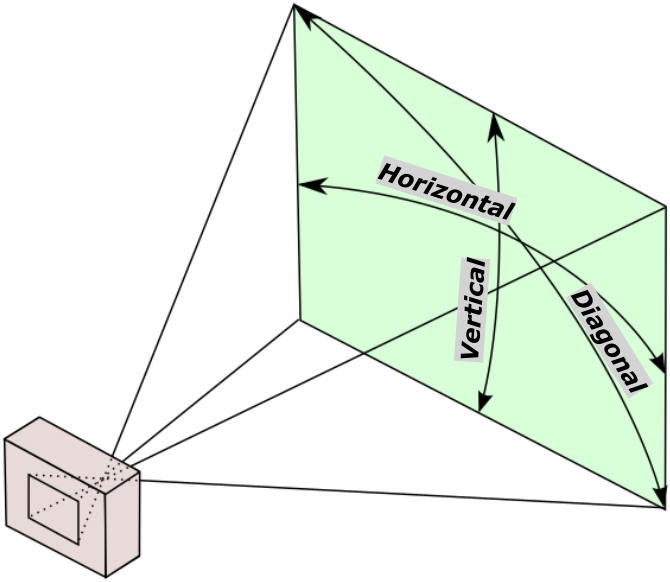
Gambar 2.4 Pemetaan gambar ke suara oleh Meijer [22]

Dalam alat bantu navigasi yang ditawarkan pada proposal ini, digunakan representasi visual dalam bentuk audio yang berbeda dengan metode Meijer [22]. Jika suara yang dihasilkan tiap *frame* pada metode Meijer periodik dan kompleks, maka pada konsep yang baru dicoba sebuah metode pengkonversian sebuah frame ke dalam informasi suara yang dinamis dan sederhana. Suara stereofonik akan menggambarkan sebuah *frame* dalam bentuk audio yang mudah dibayangkan dengan menggunakan konsep pemindaian berputar serta menggunakan IMU untuk menentukan ketinggian posisi vertikal yang direpresentasikan.

2.7 Teknologi Citra Kedalaman

Gambar kedalaman atau jarak merupakan sebuah teknologi pemindaian pada suatu lingkungan atau objek dimana pemindaian

tersebut untuk mengetahui jarak dari sensor kedalaman ke objek tersebut. Berbeda dengan sebuah sensor ultrasonik atau inframerah yang hanya mendeteksi jarak pada sebuah titik atau area kecil, dengan sensor atau kamera kedalaman, didapatkan nilai kedalaman ke semua titik yang ada dalam suatu rentang (horisontal, vertikal, dan diagonal). Untuk rentang pandang horisontal, vertikal, dan diagonal adalah seperti pada **Gambar 2.5**, yang pada umumnya dalam satuan derajat menunjukkan lebar rentang bidang pandang dari kamera. Dengan adanya citra kedalaman, banyak aplikasi yang dapat digunakan termasuk untuk permainan konsol hingga *robot vision*.



Gambar 2.5 Rentang pandang vertikal, horisontal, dan diagonal

Biasanya, sebuah sensor atau kamera kedalaman dilengkapi juga dengan kamera warna, dikenal dengan kamera RGB-D (*Red Green Blue + Depth*). Kamera RGB-D mengombinasi informasi citra warna (RGB) dengan informasi jarak atau kedalaman (*depth*) per pikselnya. Sebenarnya sensor semacam ini sudah lama dikembangkan, seperti pada produk pada PMD Tech [24] dan Switss Ranger SR4000 [25]. Meskipun demikian, sensor-sensor tersebut kurang terjangkau dengan harga yang berkisar

pada 100 juta rupiah, sedangkan sensor RGB-D yang tersedia di pasaran sekarang sangat terjangkau di harga antara 2 – 5 juta rupiah. Teknologi *depth sensing* tiap pikselnya yang sekarang digunakan dalam sensor RGB-D di pasaran dikembangkan oleh PrimeSense [26] dan telah dipatenkan [27]. Aplikasi dari teknologi ini yaitu pada Microsoft Kinect dan Asus Xtion PRO. Keduanya merupakan produk yang digunakan sebagai NUI (*Natural User Interface*). Microsoft Kinect, yang digunakan sebagai aksesori dari Xbox, sangat diminati oleh konsumen dan terjual dengan sangat cepat setelah peluncurannya pada November 2010. Sampai sekarang, kamera RGB-D yang tersedia di pasaran (consumer RGB-D camera) banyak digunakan dalam berbagai proyek dan aplikasi dari amatir hingga profesional.

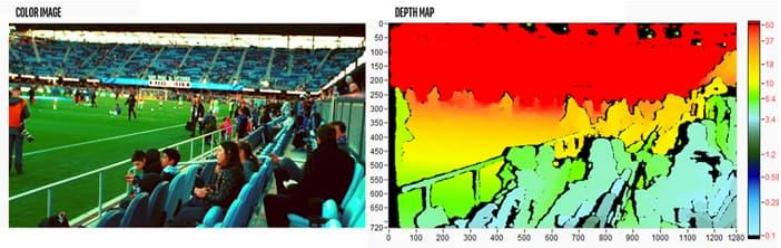
2.7.1 Kamera Kedalaman (Depth Camera)

Kamera digital pada umumnya menghasilkan gambar berupa kumpulan nilai piksel yang membentuk sebuah matriks 2 dimensi. Setiap pikselnya akan memiliki nilai, yang biasanya dikenal sebagai nilai RGB (*red green blue*) atau merah hijau biru. Setiap nilai dari warna merah, hijau, dan biru memiliki rentang dari nol sampai dua ratus dua puluh lima (0-255) jika tiap warna dikodekan dengan resolusi 8-bit. Sebagai contoh, warna hitam akan memiliki nilai RGB secara berturut-turut (0, 0, 0) sedangkan warna merah murni yang paling terang bernilai RGB (255, 0, 0). Ribuan hingga jutaan piksel yang tergabung membentuk sebuah gambar fotografi yang biasa kita kenal sehari-hari.

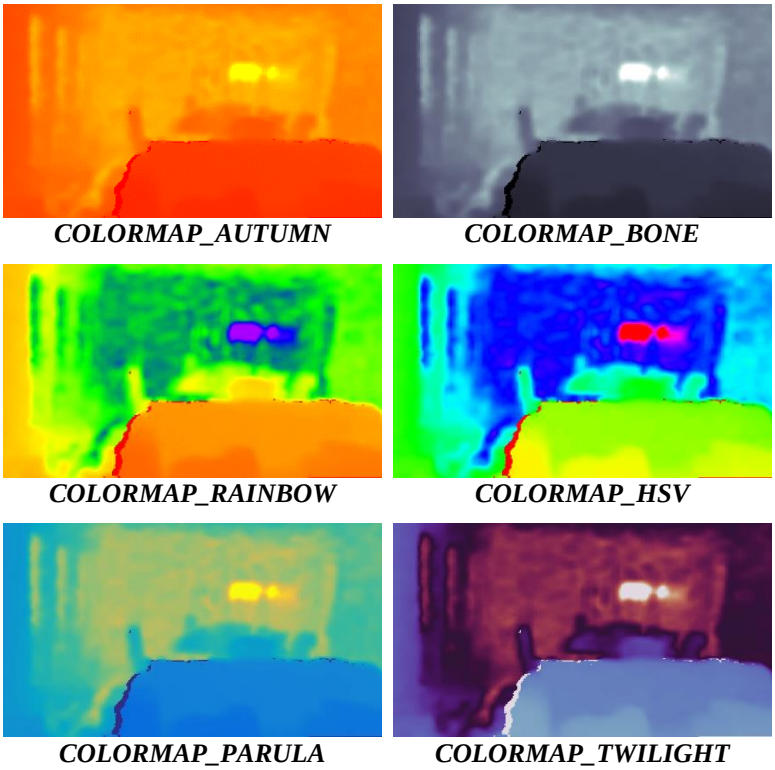
Di sisi lain, sebuah kamera kedalaman (*depth camera*), memiliki piksel-piksel juga dimana nilai tiap pikselnya menunjukkan jarak antara kamera dengan piksel tersebut, atau disebut kedalaman. Banyak dari kamera kedalaman memiliki sensor RGB sekaligus dengan sensor kedalaman, sehingga dapat memberi nilai dari tiap piksel menjadi terdiri dari 4 nilai, yaitu RGB-D (*red green blue + depth*) atau merah hijau biru + kedalaman. Tentunya, dalam menggabungkan sensor warna RGB dan sensor kedalaman diperlukan algoritma perhitungan dan teknik khusus, sehingga piksel yang direpresentasikan oleh sensor RGB harus sama dengan piksel yang direpresentasikan oleh sensor kedalaman [28].

Meskipun gambar kedalaman memiliki sudut pandang yang berbeda dengan warna, tetapi untuk titik yang direpresentasikan pada gambar kedalaman memiliki titik yang sama pada gambar warna dengan lokasi yang berbeda. Jika jarak (*baseline*) antara kamera warna dan kamera kedalaman bernilai tetap dan diketahui, maka proses penggabungan akan lebih mudah dengan teknik yang sama untuk

menggabungkan gambar kedalaman dari sensor kiri dan sensor kanan pada *stereo vision* untuk mendapatkan gambar yang *align*.



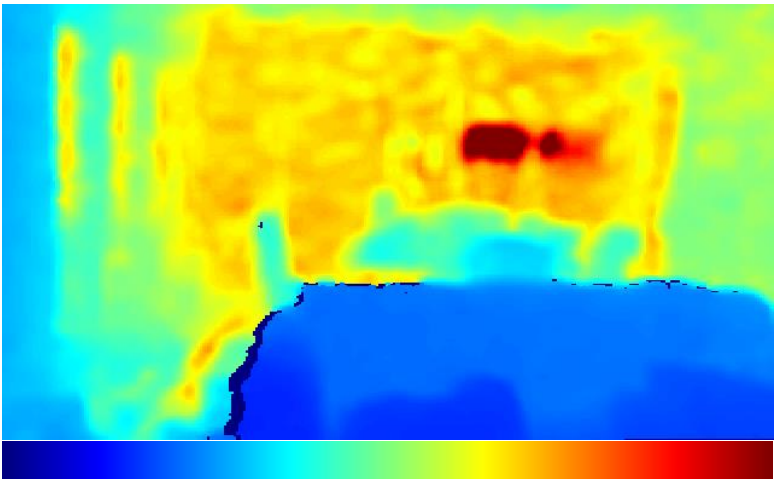
Gambar 2.6 Hasil penampilan gambar kedalaman dengan *colormap*



Gambar 2.7 Contoh jenis-jenis *colormap*

Pemetaan warna kedalaman ini disebut juga *colormap*, dimana urutan sekaligus jenis warna yang digunakan dapat berbeda-beda tergantung *colormap* yang digunakan. Berikut pada **Gambar 2.7** merupakan jenis-jenis *colormap* yang lain dengan obyek gambar yang sama.

Pada laporan tugas akhir ini, salah satu jenis *colormap* digunakan dari awal hingga akhir sehingga tidak menimbulkan kebingungan denga pemetaan warna terhadap kedalaman. *Colormap* yang digunakan adalah jenis *COLORMAP_JET*. Berikut adalah contoh warna dari gambar kedalaman yang akan digunakan sepanjang laporan tugas akhir ini beserta pemetaan warnanya (warna paling kiri ada nilai minimum atau nol, dan warna paling kanan adalah nilai maksimum) pada **Gambar 2.8**



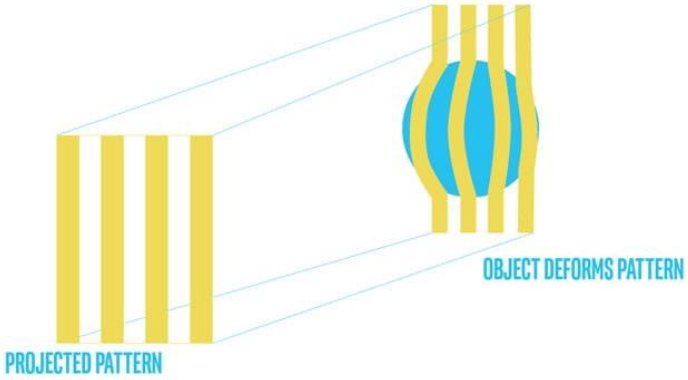
Gambar 2.8 Jenis *colormap* yang digunakan dalam laporan ini beserta pemetaan warnanya (*COLORMAP_JET*)

2.7.2 Jenis-Jenis Kamera Kedalaman

Terdapat beberapa metode dalam kalkulasi jarak atau kedalaman, dengan masing-masing kelebihan dan kekurangannya, serta rentang optimalnya. Pemilihan jenis kamera kedalaman yang cocok tergantung aplikasi serta fungsi dan kegunaan yang diinginkan dari kamera tersebut, seperti berapa rentang jarak yang ingin dibaca, seberapa tinggi akurasi yang diinginkan, apakah akan dioperasikan di luar ruangan, dan sebagainya.

2.7.2.1 Structured Light dan Coded Light

Structured Light berbeda dengan *Coded Light*, tetapi keduanya menggunakan teknologi yang serupa. Teknologi yang digunakan adalah dengan melakukan proyeksi cahaya (seperti *Infrared*) pada arah proyeksinya melalui *emitter*. Cahaya yang diproyeksikan memiliki pola tertentu, secara visual maupun secara waktu. Setelah itu, pola yang didapatkan dari hasil proyeksi ditangkap kembali oleh sensor dan dilihat informasi perubahan pola yang ada untuk mengetahui informasi kedalaman (*depth*). Sebagai contoh, jika pola yang digunakan adalah pola garis-garis, dan kemudian diproyeksikan pada sebuah benda berbentuk bola, maka garis-garis yang diproyeksikan akan ter-deformasi dan terlihat bengkok di sekitar permukaan bola seperti pada **Gambar 2.9**.



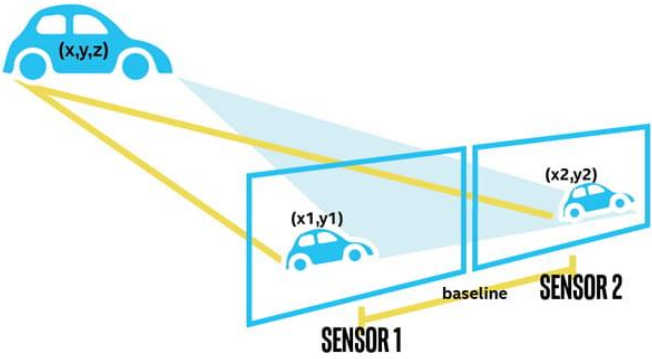
Gambar 2.9 Contoh proyeksi *structured light*

Ketika benda digerakkan mendekati atau menjauhi *emitter*, maka pola yang terbentuk akan berubah pula. Dari *disparity* antara gambar pola yang ditangkap sensor dengan gambar pola yang seharusnya, dapat dihitung jarak antara sensor kamera dengan benda tersebut tiap pikselnya. Sensor *depth* dengan teknologi ini memiliki kelemahan yaitu hanya bekerja dalam rentang jarak yang pendek, meskipun akurasi yang didapatkan dalam rentang yang pendek tersebut cukup baik.

2.7.2.2 Stereo Depth

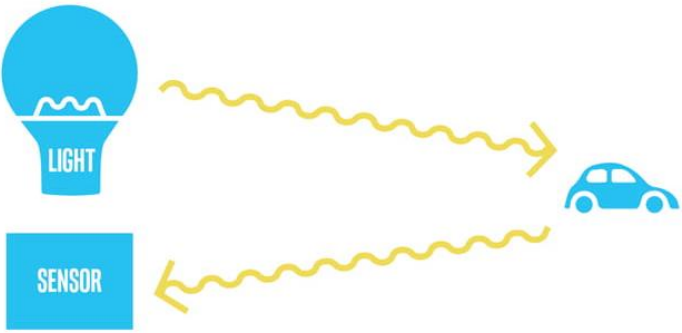
Kamera *stereo depth* menggunakan dua sensor yang terpisah pada jarak tertentu. Kedua sensor ini akan menangkap gambar yang berbeda dari sudut pandangnya masing-masing. Karena jarak dari kedua sensor bersifat tetap dan diketahui (disebut *baseline*), maka dapat

dikomparasikan antara kedua gambar yang didapatkan untuk mengetahui informasi kedalaman (*depth*) dengan perhitungan trigonometri. Walaupun kamera *stereo depth* dilengkapi dengan *emitter infrared* pula untuk meningkatkan akurasi, tetapi cara kerjanya sangat berbeda dengan *structured light* dan *coded light*. Salah satu perbedaannya adalah *stereo depth* dapat bekerja dengan berbagai jenis cahaya termasuk cahaya yang dapat dilihat oleh mata kita. Cara kerja dari metode ini seperti pada **Gambar 2.10** serupa dengan bagaimana mata kita melihat untuk menghasilkan persepsi kedalaman. Otak kita mengkalkulasi perbedaan proyeksi yang ditangkap oleh mata kanan dan mata kiri untuk menghasilkan persepsi dan mengetahui jarak antara kita dengan objek tersebut, meskipun jarak yang didapatkan tidak tepat secara metrik dan akurat.



Gambar 2.10 Cara kerja *stereo depth*

Selanjutnya adalah *time of flight* dan LIDAR. Pada metode-metode sebelumnya, menggunakan parameter yang telah diketahui sebelumnya, seperti pola dan jarak. Pada metode ini waktu perjalanan dari suatu cahaya juga sudah diketahui sebelumnya, dan informasi tersebut digunakan dalam menghitung jarak. LIDAR sensor memanfaatkan metode ini untuk mendapatkan informasi jaraknya, dan biasanya digunakan pada self-driving car. Namun, kekurangan dari metode ini adalah cukup rentan dibandingkan sensor lainnya karena dapat terganggu dari cahaya yang dipancarkan oleh sumber lain. **Gambar 2.11** merupakan visualisasi prinsip kerja dari metode ToF.



Gambar 2.11 Prinsip kerja *time of flight* dan LIDAR

2.8 Inertial Measurement Unit (IMU)

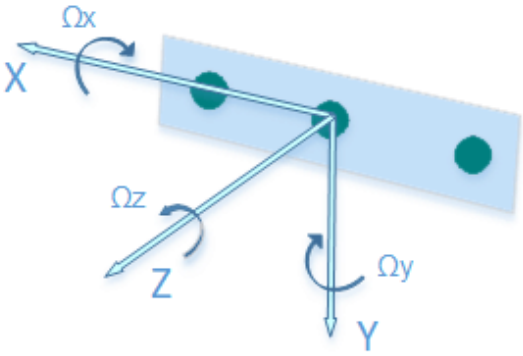
Inertial measurement unit atau disebut juga *motion sensor* merupakan perangkat elektronik yang mengukur gaya spesifik, tingkat sudut, dan terkadang orientasi arah menggunakan kombinasi akselerometer, giroskop, dan terkadang magnetometer. Secara umum IMU digunakan untuk manuver pesawat, termasuk pesawat tanpa awak, dan banyak lagi seperti pesawat ruang angkasa dan satelit.

Sebuah IMU sederhana yang terdiri dari akselerometer dan giroskop dapat mendeteksi akselerasi linear menggunakan akselerometer dan kecepatan sudut menggunakan giroskop. Pada beberapa IMU dilengkapi juga magnetometer yang biasanya digunakan untuk referensi arah. Konfigurasi dari dua atau tiga sensor ini digunakan untuk mengetahui rotasi dalam tiga dimensi, yang biasa disebut *yaw*, *pitch*, dan *roll*.

Secara lebih spesifik, terdapat tiga sumbu yang biasa digunakan dalam navigasi pesawat. Sumbu pertama yaitu sumbu normal atau disebut *yaw*. Sumbu ini digambarkan dari objek kebawah, dan perputaran pada sumbu ini menunjukkan arah tuju dari sebuah pesawat. Kedua adalah sumbu transversal atau disebut juga sumbu lateral, dan lebih dikenal dengan *pitch*. Sumbu ini digambarkan dari objek ke kanan, paralel dengan garis yang dibentuk saat kita merentangkan tangan ke kiri dan ke kanan. Perubahan pada sumbu ini menunjukkan elevasi dari sebuah pesawat, terutama saat lepas landas dan mendarat. Terakhir adalah sumbu longitudinal atau biasa disebut *roll*. Sumbu ini digambarkan ke arah depan dari sebuah objek. Pergeseran sudut pada sumbu ini disebut sebagai *bank*.

Pada perputaran sudut positif di sumbu ini, pesawat akan menaikkan sayap kiri dan menurunkan sayap kanan sehingga pesawat bermanuver ke kanan. Sebaliknya, perubahan sudut yang negatif pada pesawat mengakibatkan sayap kiri turun dan sayap kanan naik sehingga pesawat bermanuver ke kiri.

Berikut pada **Gambar 2.12** adalah ilustrasi sumbu *roll*, *pitch*, dan *yaw*, dimana sumbu X merupakan *pitch*, sumbu Y merupakan *yaw*, dan sumbu Z merupakan *roll*. Akan tetapi penamaan sumbu X Y Z ini tidaklah baku. Penggunaan kata *roll*, *pitch*, dan *yaw* lebih diterima secara universal untuk segala keadaan.



Gambar 2.12 Ilustrasi sumbu *pitch*, *yaw*, dan *roll* berturut-turut pada sumbu X, Y, dan Z

2.9 Intel® RealSense™

RealSense™ merupakan teknologi kamera *depth* dan *tracking* yang dikembangkan oleh Intel® dan didesain untuk memberi mesin dan berbagai perangkat lainnya kemampuan persepsi kedalaman (*depth*) agar dapat "melihat" dan mengerti berbagai hal. RealSense™ banyak digunakan dalam kapabilitasnya sebagai *computer vision* pada *autonomous drone*, robot, *AR/VR*, dan berbagai perangkat cerdas lainnya. Munculnya RealSense™ pada pasar kamera RGB-D membuat terobosan tersendiri karena harganya yang terjangkau serta kemudahan pengaplikasiannya dalam berbagai hal.

Hingga awal tahun 2020, Intel® RealSense™ memiliki 3 jenis produk, yaitu *stereo depth camera* yang memiliki seri D415, D435, dan D435i (untuk tipe modul seri D400, D410, D415, D420, dan D430),

kemudian *coded light depth camera* dengan seri SR305 (untuk tipe modul seri SR300), dan *tracking camera* T265 (untuk tipe modul seri T261). Pada tracking camera, informasi kedalaman (*depth*) tidak bisa didapatkan karena kamera ini didesain untuk *tracking* dan bukan merupakan *depth camera* pada umumnya. Selanjutnya untuk tipe *coded light*, penerapan dalam penelitian ini kurang sesuai karena *depth camera* jenis ini memiliki jarak kerja yang pendek, yaitu sekitar 1 meter dari kamera, sehingga meninggalkan pilihan yang sesuai yaitu *stereo depth camera*.

RealSense™ *Depth Camera* D435i memiliki *stereo depth module* D430 dari Intel® dan kamera RGB, juga 6 DoF (*Degree of Freedom*) IMU (*accelerometer* dan *gyroscope*), disertai *vision processor* D4 yang terintegrasi dalam sebuah perangkat dengan dimensi 90x25x25 mm dengan massa 72 gram. Berikut pada **Tabel 2.2** merupakan spesifikasi dari produk Intel® RealSense™ D435i. [29]

Tabel 2.2 Spesifikasi Intel® RealSense™ D435i

Fitur	Lingkungan Penggunaan:	*Indoor* dan *Outdoor*
	Teknologi Sensor Citra:	*Global Shutter* Ukuran piksel 3x3 µm
	Jarak Maksimum:	Sekitar 10 meter Akurasi dipengaruhi kalibrasi, keadaan, dan kondisi pencahayaan.
Kedalaman	Tekonologi Kedalaman:	*Active Stereo IR*
	Bidang Pandang HD Kedalaman:	Horisontal 87°±3° Vertikal 58°±1° Diagonal 95°±3°
	Jarak Kedalaman Minimum:	0.105 m
	Resolusi dan *Frame Rate* Gambar Kedalaman:	Hingga 1280 x 720 Hingga 90 fps
Warna (RGB)	Resolusi Sensor RGB:	1920 x 1080
	Frame Rate Gambar Warna:	30 fps
	Bidang Pandang RGB:	Horisontal 69.4°±3° Vertikal 42.5°±3° Diagonal 77°±3°

Fisik	Dimensi: (Panjang x Lebar x Tinggi)	90 mm x 25 mm x 25 mm
	Konektor:	USB-C 3.1 Gen 1
	Mekanisme *mounting*:	Satu ulir baut 1/4-20 UNC. Dua ulir baut ukuran M3

Intel® RealSense™ *Vision Processor* D4 untuk sistem teknologi kedalaman memberikan data kedalaman dengan kualitas tinggi untuk sistem yang diinginkan. Data kedalaman dihasilkan oleh teknologi *stereo imaging* dengan atau tanpa proyektor sinar inframerah (IR). *Vision processor* juga memiliki kemampuan untuk sinkronisasi dengan gambar warna (RGB). Berikut pada **Tabel 2.3** merupakan format gambar beserta resolusi dan *frame rate* yang tersedia untuk produk D435i untuk koneksi USB 3.1. [29]

Untuk bidang pandang kedalaman pada produk D435i, terdapat dua mode. Pertama adalah mode HD seperti pada **Tabel 2.2**, yaitu untuk horisontal 87° vertical 58° diagonal 95° dengan toleransi mekanik sebesar 5° dan toleransi antar perangkat yang berbeda (perbedaan antar lensa atau perbedaan antar modul) sebesar 3°. Kemudian kedua mode VGA dengan bidang pandang horisontal 75° vertical 62° diagonal 89° dengan toleransi mekanik sebesar 5° dan toleransi antar perangkat yang berbeda (perbedaan antar lensa atau perbedaan antar modul) sebesar 3°.

Bidang pandang kedalaman (*Depth FOV*) dalam berbagai kedalaman (*Z*) dapat dihitung dengan persamaan:

$$Depth\ FOV = \frac{HVOF}{2} + \tan^{-1}\left\{\tan\left(\frac{HFOV}{2}\right) - \frac{B}{Z}\right\} \quad \textbf{(2.6)}$$

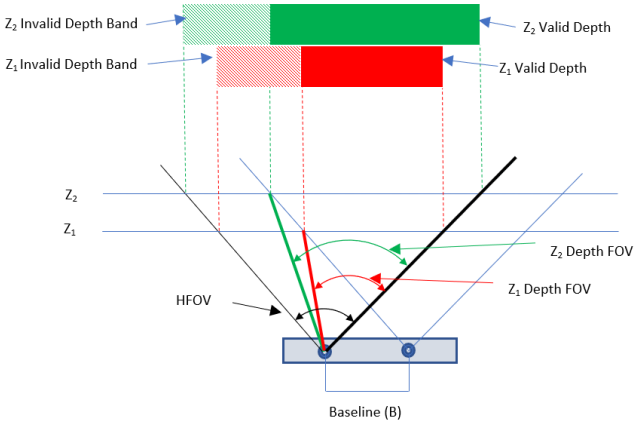
dengan:

Depth FOV : Bidang pandang kedalaman
HFOW : Bidang pandang horisontal untuk sensor kiri pada modul kedalaman
B : Jarak antar kedua sensor pada modul kedalaman (*baseline*)
Z : Jarak kedalaman dari modul kedalaman terhadap objek

Berikut ilustrasi perhitungan *Depth FOV* untuk gambar kedalaman pada **Gambar 2.13**.

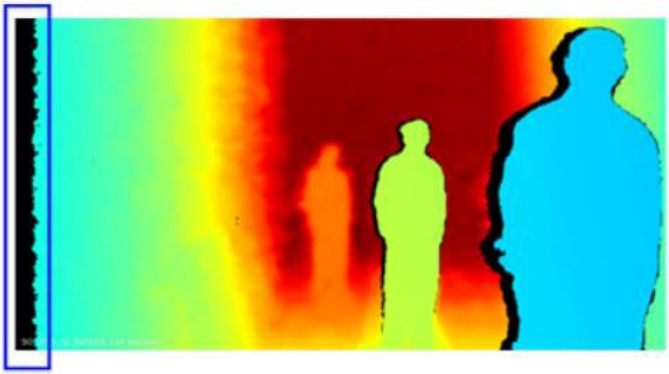
Tabel 2.3 Format gambar dengan resolusi dan *frame rate* untuk Intel® RealSense™ D435i

Format	Resolusi	*Frame Rate* (FPS)	Keterangan
Z (16-bit)	1280x720	6, 15, 30	Gambar Kedalaman
	848x480	6, 15, 30, 60, 90	
	640x480	6, 15, 30, 60, 90	
	640x360	6, 15, 30, 60, 90	
	480x270	6, 15, 30, 60, 90	
	424x240	6, 15, 30, 60, 90	
Y8 (8-bit)	1280x720	6, 15, 30	Gambar dari sensor stereo (kiri dan kanan)
	848x480	6, 15, 30, 60, 90	
	640x480	6, 15, 30, 60, 90	
	640x360	6, 15, 30, 60, 90	
	480x270	6, 15, 30, 60, 90	
	424x240	6, 15, 30, 60, 90	
YUY2 (16-bit)	1920x1080	6, 15, 30	Gambar Warna (RGB)
	1280x720	6, 15, 30	
	848x480	6, 15, 30, 60	
	640x480	6, 15, 30, 60	
	640x360	6, 15, 30, 60	
	480x270	6, 15, 30, 60	
	424x240	6, 15, 30, 60	
	320x240	6, 30, 60	
	320x180	6, 30, 60	
Kalibrasi	1280x800	15, 25	Untuk D435i



Gambar 2.13 Ilustrasi *Depth FOV* pada gambar kedalaman

Dengan demikian pada gambar kedalaman akan terdapat bagian yang disebut *invalid depth band*. Pada *stereo vision* dari RealSense™, data dari sensor kiri digunakan sebagai referensi untuk melakukan perhitungan kedalaman sehingga pada *invalid depth band* yang berada di sebelah kiri adalah bagian yang tidak tumpeng tindih dengan gambar yang dihasilkan oleh sensor kanan. Seperti jauh jarak pandang kamera, maka *invalid depth band* akan semakin berkurang pada gambar kedalaman yang dihasilkan. Secara keseluruhan gambar kedalaman merupakan peta kedalaman yang valid ditambah *invalid depth band* pada sebelah kiri, seperti pada **Gambar 2.14**.



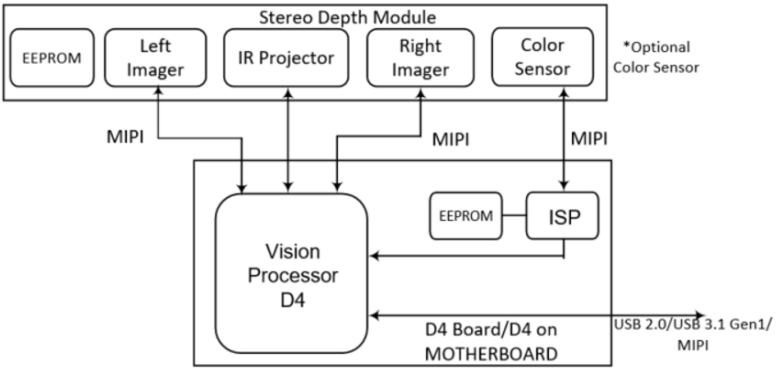
Gambar 2.14 Gambar kedalaman yang dengan *invalid depth band*

IMU yang terdapat pada produk D435i memiliki spesifikasi pada **Tabel 2.4**. [29]

Tabel 2.4 Spesifikasi IMU pada Intel® RealSense™ D435i

Kamera	Parameter	Properti
Intel® RealSense™ *Depth Camera* D435i (D435 + BMI055)	Derajat Kebebasan (*Degree of Freedom*)	6
	Rentang Akeslerasi	±4g
	Sample Rate Accelerometer	62.5, 250 (Hz) ±5%
	Rentang *Gyroscope*	±1000 deg/s
	Sample Rate Gyroscope	200, 400 (Hz) ±0.3%
	Akurasi *timestamp*	50 µsec

Secara keseluruhan, blok diagram dari RealSense™ (tanpa IMU) ditnjukkan pada **Gambar 2.15**.



Gambar 2.15 Blok diagram kamera RGB-D RealSense™

2.10 NVIDIA® Jetson™

Jetson™ merupakan produk *embedded computing board* dari NVIDIA® yang dirancang untuk berbagai aplikasi seperti *machine learning*, robot, dan penggunaan yang portable lainnya. Salah satu varian dari Jeton™ adalah tipe Nano. Untuk Nano dilengkapi juga dengan Developer Kit untuk mempermudah penggunaan. Berikut pada **Tabel 2.5** merupakan spesifikasi Nano Developer Kit.

Tabel 2.5 Spesifikasi NVIDIA® Jetson™ Nano Developer Kit

GPU	NVIDIA® Maxwell™ architecture with 128 NVIDIA CUDA® cores 0.5 TFLOPs (FP16)
CPU	Quad-core ARM® Cortex®–A57 MPCore processor @ 1.43 GHz
Memory	4 GB 64-bit LPDDR4 1600MHz – 25.6 GB/s
Storage	MicroSD (not included)
Video Encode	250 MP/sec 1x 4K @ 30 (HEVC) 2x 1080p @ 60 (HEVC) 4x 1080p @ 30 (HEVC)
Video Decode	500 MP/sec 1x 4K @ 60 (HEVC) 2x 4K @ 30 (HEVC) 4x 1080p @ 60 (HEVC) 8x 1080p @ 30 (HEVC)
Camera	12 lanes (3x4 or 4x2) MIPI CSI-2 DPHY 1.1 (18 Gbps)
Connectivity	Gigabit Ethernet, M.2 Key E
Display	HDMI 2.0 or DP1.2 | eDP 1.4 | DSI (1 x2) 2 simultaneous
UPHY	1 x1/2/4 PCIE, 1x USB 3.0, 3x USB 2.0
I/O	1x SDIO / 2x SPI / 4x I2C / 2x I2S / GPIOs -> I2C, I2S
USB	4x USB 3.0, USB 2.0 Micro-B
Mechanical	100 mm x 80 mm x 29 mm

2.11 Pemrograman Python

Python adalah bahasa pemrograman multifungsi yang dibuat oleh Guido van Rossum dan dirilis pada tahun 1991. GvR, begitu ia biasa disebut di komunitas Python, menciptakan Python untuk menjadi *interpreter* yang memiliki kemampuan penanganan kesalahan (*exception handling*) dan mengutamakan sintaksis yang mudah dibaca serta dimengerti (*readability*). Didesain untuk memudahkan dalam purwarupa, Python menjadi bahasa yang sangat mudah dipahami dan fleksibel.

Pada bahasa pemrograman Python, terdapat dua opsi untuk menjalankan program, yaitu:

1. Mode Interaktif
2. Mode Skrip

Pertama yaitu mode interaktif, atau dikenal dengan REPL (*Read Eval Print Loop*) memberikan kemudian untuk menjalankan sebagian kode atau bahkan sebaris kode Python. Kode dieksekusi via Python shell yang biasanya sudah disertakan saat peng-*install*-an Python. Mode interaktif sangat mudah digunakan untuk mempelajari dan mencoba berbagai fungsi dari Python atau sekedar untuk melakukan perhitungan.

Jika diperlukan kode Python yang sangat panjang, atau program Python ingin kita tulis dalam berbagai file, maka mode interaktif tidak disarankan. Pada mode skrip, kita dapat menulis kode secara langsung dan menyimpannya dalam ekstensi .py yang merupakan ekstensi standart Python. Kita dapat menggunakan berbagai *text editor* yang mudah dan nyaman untuk menulis program Python pada mode ini. Untuk menjalankannya, kita hanya butuh melakukan *run*.

Berbeda dengan C, Python merupakan bahasa pemrograman interpretasi (*interpreter language*), sedangkan C adalah *compiler language*. Perbedaa mendasar mengenai kedua bahasa ini adalah pada saat dijalankan. Pada *compiler language*, kode yang kita tulis adakan dilakukan pemrosesan dan penerjemanahan ke bahasa mesin yang dalam bentuk biner. Hasil terjemahan ke bahasa mesin ini dapat kita jalankan sewaktu-waktu dengan melakukan eksekusi program tersebut. Berbeda dengan *interpreter language*, dimana program tidak diterjemahkan seluruhnya ke bahasa mesin, melainkan diterjemahkan sebagian dalam waktu tertentu. Kekurangan dari *interpreter language* adalah waktu eksekusi yang lebih lambat ketimbang *compiler language*, akan tetapi pada saat mengembangan lebih cepat karena tidak memerlukan proses *compiling* yang terus menerus saat memperbaiki program yang kita tulis.

2.11.1 API dalam Package

Dalam Python, terdapat beberapa istilah yang harus dipahami. Pertama yaitu modul yang merupakan file yang didalamnya terdapat definisi dan statemen dalam Python. Modul berbentuk file Python dengan ekstensi .py pada nama file. Modul dapat di *import* ke dalam program Python yang lain baik dalam mode interaktif maupun script. Selanjutnya adalah istilah *package*. *Package* merupakan salah satu bentuk strukturisasi *namespace* dari modul Python dengan mengikuti aturan dan standar yang ada. Secara umum, fungsi modul sama dengan *package*,

hanya saja *package* digunakan sebagai bentuk standar dari *library* pada Python. [30]

API adalah singkatan dari *Application Programming Interface*, yaitu *interface* yang mendefinisikan interaksi antar berbagai *software* atau program. API mendefinisikan bagaimana cara untuk memanggil atau meminta data atau fungsi, bagaimana format data tersebut, dan lain-lain. Dalam penggunaannya pada sebuah program, biasanya API berhubungan erat dengan *software library*. Sebuah API dapat memiliki banyak implementasi dalam bentuk berbagai *library* dan berbagai bahasa pemrograman. Pemisahan API dari implementasinya memungkinkan program yang ditulis dalam satu bahasa menggunakan *library* dari bahasa pemrograman yang lainnya.

Dalam pemrograman antar bahasa, dikenal istilah *wrapping*, yaitu menyajikan fungsionalitas dari suatu program yang ditulis dalam suatu bahasa pemrograman ke bahasa pemrograman lain. Terdapat berbagai cara untuk melakukan *wrapping*, salah satunya adalah dengan cara manual. Di dalam Python, jika kita ingin menggunakan program dalam bahasa C atau C++ yang telah kita tulis untuk digunakan dalam program dalam bahasa Python, terdapat beberapa cara yang dijelaskan pada dokumentasi Python untuk Python/C API. [31]

2.11.2 Package yang Digunakan

Pada penelitian ini, dibutuhkan berbagai *package* yang menunjang untuk pembuatan algoritma dan pemrograman perangkat lunak. *Packages* yang akan digunakan adalah sebagai berikut

2.11.2.1 Librealsense

Penjelasan mengenai Intel® RealSense™ SDK 2.0 *librealsense* untuk bahasa pemrograman Python (*python wrapper*) atau disebut juga *pyrealsense2*. Pada *package pyrealsense2* disediakan fungsi untuk mengambil data dari berbagai produk RealSense™, dan untuk melakukan pemrosesan data.

2.11.2.2 OpenCV

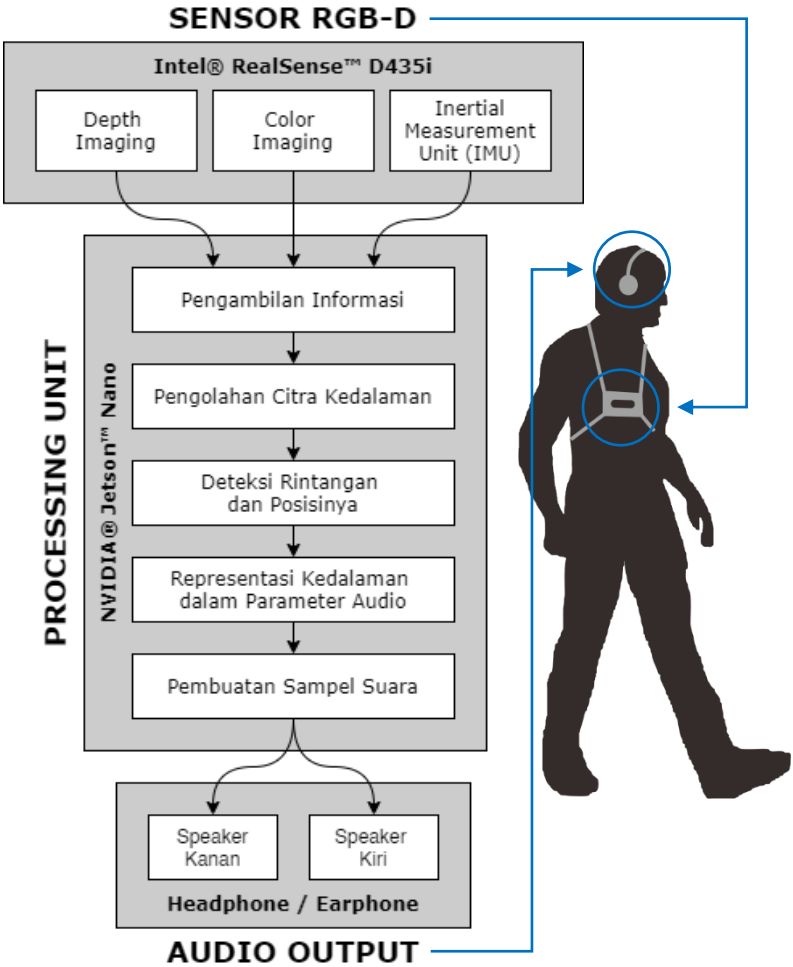
OpenCV merupakan *library* yang sangat dikenal dalam dunia *computer vision*. Dari namanya yang merupakan singkatan dari *Open Source Computer Vision*, OpenCV sangat dibutuhkan untuk melakukan pengolahan dan pemrosesan data yang berbentuk gambar. Pengolahan gambar ini dapat berupa mengubah ukuran, mengubah warna, melakukan transformasi dan morfologi, serta fungsi-fungsi lainnya yang sangat berguna.

2.11.2.3 PyAudio

PyAudio merupakan sebuah *package* Python yang berasal dari *binding library* C yaitu PortAudio. PyAudio berfungsi untuk melakukan *interface* dengan komponen audio pada komputer, seperti memutar audio, merekam audio, melakukan *stream* audio, dan sebagainya. Dalam pengolahan audio untuk dikeluarkan menjadi suara melalui *speaker* atau perangkat audio lainnya, data perlu diberikan dalam bentuk *frame*. *Frame* atau biasa disebut sebagai *chunk* ini didefinisikan sebelumnya. Selain itu PyAudio mendukung pemutaran audio *multi-channel* sehingga dapat digunakan memutar suara stereo.

BAB III
PERANCANGAN SISTEM

3.1 Diagram Blok Sistem



Gambar 3.1 Diagram blok sistem

Pada **Gambar 3.1** merupakan diagram blok dari rancangan sistem. Pada diagram blok tersebut terlihat informasi yang berasar dari sensor RGB-D berupa gambar kedalaman, gambar warna, dan IMU dilakukan pengambilan data ke CPU beruba *embedded computing board* dan dilakukan pengolahan citra yang dilanjutkan dengan deteksi rintangan dan posisinya, representasi informasi kedalaman dalam parameter audio, dan pembuatan sampel suara. Sampel suara yang telah diproduksi kemudian dikeluarkan dalam bentuk audio ke transduser suara stereo berupa *headphone* atau *earphone*.

Perangcangan sistem secara umum dibagi menjadi dua, yaitu perancangan perangkat keras dan perancangan perangkat lunak, yang akan dijelaskan pada subbab selanjutnya.

3.2 Sistem Navigasi untuk Tunanetra

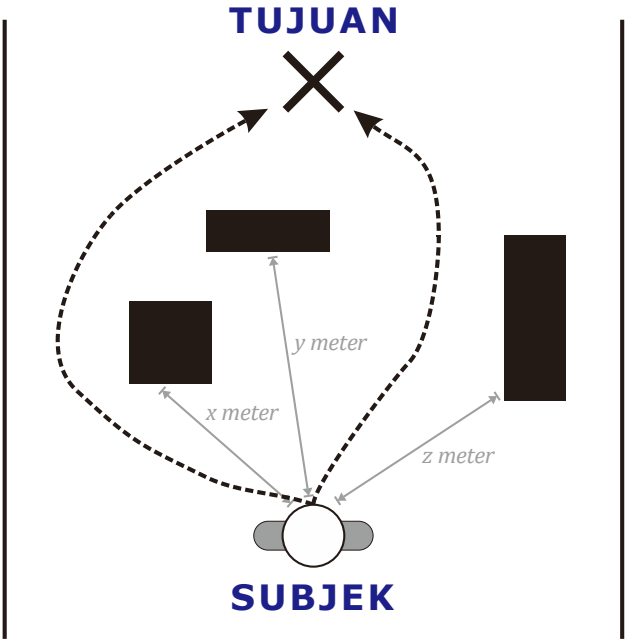
Alat bantu ini dirancang untuk mempermudah navigasi bagi tunanetra. Navigasi adalah sistem yang berfokus terhadap proses *monitoring* dan kontrol dari pergerakan dari suatu tempat ke tempat lain. Navigasi biasanya erat dikaitkan dengan pemetaan dan lokalisasi, dengan berbagai konsep mulai dari letak absolut berdasarkan *latitude* dan *longitude* maupun letak relatif terhadap objek yang lain atau titik acuan tertentu. Sistem navigasi yang seperti ini biasanya terdapat pada pesawat, kapal, dan kendaraan lainnya.

Berbeda dengan sistem navigasi pada kendaraan seperti pesawat ataupun kapal, sistem navigasi pada alat bantu ini digunakan pada manusia, yaitu tunanetra, dengan memberikan saran atau petunjuk arah bagi penggunanya melalui suara. Pada sistem navigasi untuk kendaraan maupun robot, diperlukan pemetaan dan lokalisasi untuk dapat melakukan navigasi. Tetapi, bagi tunanetra, navigasi yang diperlukan adalah untuk bermobilisasi dari satu tempat ke tempat yang lainnya. Hal ini dapat dicapai dengan mengetahui arah yang hendak dituju, yang secara tidak langsung diinformasikan dengan informasi posisi dan jarak rintangan yang ada didepannya. Dengan informasi ini, tunanetra dapat memilih jalan, arah, maupun rute mana yang akan ia ambil untuk bermobilisasi agar tidak tertabrak rintangan.

Pertama, yaitu rintangan. Dalam penelitian ini diberi batasan masalah berupa rintangan yang dimaksud adalah halangan dengan ketinggian lebih tinggi atau hampir sejajar dengan ketinggian sensor. Rintangan ini dapat dideteksi dengan melihat nilai kedalaman dari sensor.

Kedua yaitu posisi dari rintangan. Kamera kedalaman yang digunakan memiliki spesifikasi berupa FOV (*Field of View*). FOV ini membatasi rentang pandang dari kamera kedalaman ini sendiri. Saat digunakan dan diarahkan pada suatu pandangan, maka rintangan yang dapat terdeteksi adalah rintangan yang ada di dalam FOV, maka perlu dilakukan pemosisian rintangan dalam FOV.

Dengan kedua informasi tersebut, tunanetra diharapkan dapat memilih jalan, arah, ataupun rute yang tepat dan tidak tertabrak rintangan. Berikut pada **Gambar 3.2** merupakan ilustrasi navigasi dari subjek tunanetra. Ditunjukkan ada dua rute yang memungkinkan jika diketahui informasi jarak rintangan objek dan posisi relatifnya.



Gambar 3.2 Ilustrasi navigasi dari subjek tunanetra

Navigasi tetap dilakukan oleh tunanetra sebagai pengguna dari alat ini, akan tetapi dengan bantuan alat ini tunanetra dapat mengetahui tabrakan yang mungkin terjadi sehingga dapat memilih rute yang tepat untuk menghindari tabrakan tersebut.

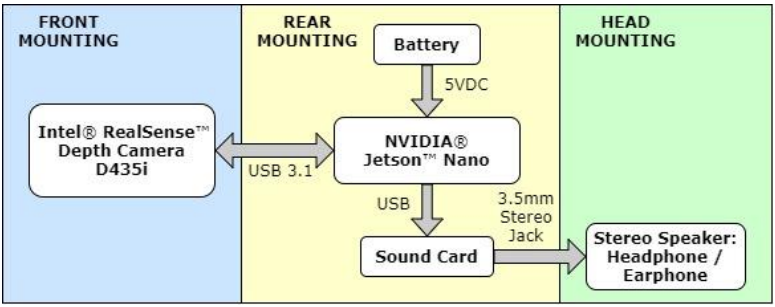
Pembagian posisi dan jarak dilakukan dengan melakukan percobaan pada subjek tunanetra. Percobaan ini untuk mengetahui pembagian yang tepat, serta dapat dipahami oleh tunanetra. Hasil dari pembagian posisi dan jarak dijelaskan lebih detail pada subbab perancangan perangkat lunak subsubbab deteksi rintangan dan posisinya (**3.5.3**). Untuk bagaimana cara menginformasikan informasi posisi dan jarak ini dalam audio kepada tunanetra, dijelaskan lebih detail pada subbab perancangan perangkat lunak subsubbab representasi informasi kedalaman dalam parameter audio (**3.5.4**)

3.3 Perancangan Perangkat Keras

Sesuai dengan diagram blok sistem, maka diperlukan rancangan perangkat keras yang mampu untuk menjalankan rancangan sistem dengan baik. Pertama adalah sensor untuk menangkap keadaan atau lingkungan di sekitar penggunanya, yaitu kamera RGB-D. Selanjutnya informasi yang diperoleh dari kamera RGB-D perlu diproses, sehingga dibutuhkan CPU berupa *Embedded Computing Board* untuk diolah dan diubah menjadi informasi audio. Dari pengolahan citra hingga produksi audio dilakukan oleh CPU. Kemudian informasi audio dikeluarkan dengan perangkat *speaker* seperti *headphone* atau *earphone*, agar dapat didengarkan oleh pengguna.

Selanjutnya adalah perangkat penunjang agar alat sesuai dengan tujuan penelitian yaitu pengembangan alat bantu navigasi untuk tunanetra yang *wearable* serta *portable*, *comfortable*, dan *easy to use*. Untuk menyediakan daya bagi perangkat-perangkat yang ada, diperlukan *power supply*. Untuk menjaga rancangan alat agar tetap *portable*, dipergunakan baterai. Kemudian diperlukan *wearable support* untuk meletakkan perangkat-perangkat yang lain pada pengguna ketika dikenakan, sehingga rancangan alat tidak hanya *portable*, tetapi juga *comfortable* dan benar-benar *wearable*.

Berikut pada **Gambar 3.3** adalah skema rangkaian perangkat keras dari sistem.



Gambar 3.3 Skema rancangan perangkat keras

Untuk tempat meletakkan perangkat-perangkat keras yang ada pada rancangan alat, dibagi menjadi tiga tempat, yang pertama adalah *front mounting* dimana kamera kedalaman RealSense™ diletakkan di posisi depan penggunanya, tepatnya di depan dada dengan bantuan *wearable support* seperti pada **Gambar 3.4**.

Gambar 3.4 *Front mounting wearable support*

Kemudian untuk *rear mounting*, perangkat yang terdiri dari CPU berupa Jetson™ Nano beserta pendukungnya diletakkan di bagian belakang penggunanya dalam sebuah kantongan atau tas untuk memudahkan pembawaan alat. Berikut pada **Gambar 3.5** adalah tas yang digunakan.

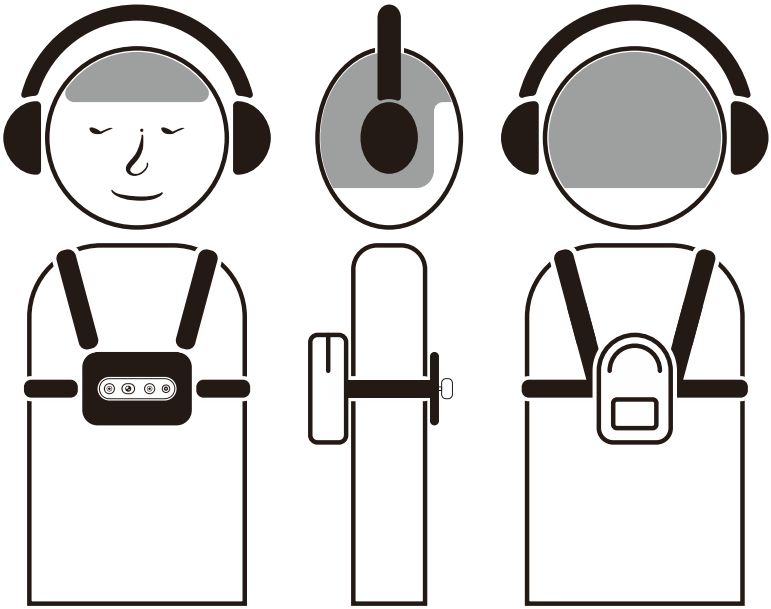
Gambar 3.5 Tas untuk *rear mounting*

Setelah tas diisi oleh perangkat-perangkat maka akan seperti pada **Gambar 3.6**.

Gambar 3.6 Tas setelah diisi oleh perangkat-perangkat

Terakhir adalah *head mounting*, yaitu perangkat yang diletakkan di bagian kepala penggunanya, tepatnya pada bagian telinga untuk dapat mendengarkan audio secara stereo untuk telinga kiri dan telinga kanan. Perangkat transduser suara yang digunakan adalah *headphone* dimana mudah untuk digunakan dan tidak memerlukan tambahan.

Penggunaan alat bantu navigasi yang dirancang secara keseluruhan seperti pada **Gambar 3.7**.



(a) (b) (c)

Gambar 3.7 Rancangan pemasangan alat bantu navigasi pada pengguna; (a) Tampak depan; (b) Tampak samping; (c) Tampak belakang

3.4 Pemilihan Perangkat Keras

Dari rancangan yang ada, dipilih perangkat-perangkat yang mendukung. Perangkat utama yaitu kamera RGB-D dan CPU. Perangkat pendukung yaitu *power supply* dan *wearable support*.

3.4.1 Sensor RGB-D

Pada penelitian ini, digunakan Intel® RealSense™ D435i. Pemilihan produk RealSense™ sendiri dikarenakan kualitas yang dimiliki oleh RealSense™ cukup baik jika dibandingkan dengan produk-produk serupa (contoh). Dari segi harga, sebagai pemain yang cukup baru di bidang *consumer depth camera*, RealSense™ memberikan harga yang cukup terjangkau dengan fitur yang cukup lengkap. Dimensi dari RealSense™ juga kecil dengan massa yang cukup ringan 72 gram, sehingga produk ini cocok untuk dipergunakan sebagai *wearable device*.

Dari produk-produk RealSense™ yang dijual di pasaran, pemilihan seri D435i dalam penelitian ini bukanlah tanpa alasan. Dari seri-seri yang lain, kelebihan pertama dari seri D435i adalah terdapat IMU (*Gyroscope* dan *Accelerometer*) tertanam yang terintegrasi di dalam produk ini. Fungsi IMU sendiri sangat penting dalam berbagai aplikasi robot dan wearable device, salah satunya yaitu untuk mengetahui pose dari perangkat apakah sudah sesuai dan seharusnya. Dalam rancang bangun perangkat ini, data dari *gyroscope* dan *accelerometer* dipergunakan untuk mentransformasi gambar kedalaman dan warna dari RealSense™ untuk mengatasi goncangan-goncangan kecil yang dihasilkan oleh pengguna saat bergerak sambil menggunakan alat ini.

Selanjutnya kelebihan dari RealSense™ D435i ini adalah penggunaan teknologi kedalaman berupa *Active Stereo IR*. Seperti yang dijelaskan pada tinjauan pustaka, teknologi stereo depth cocok digunakan untuk rentang kerja yang lebih jauh ketimbang *coded light*. Berikut pada **Gambar 3.8** merupakan foto dari RealSense™ yang digunakan dalam penelitian ini.

3.4.2 CPU Embedded Computing Board

Pada penelitian ini, digunakan NVIDIA® Jetson™ Nano Development Kit. Pemilihan *single-board computer* adalah untuk pemenuhan rancangan alat yang *portable* dan *wearable*. Meskipun memiliki kemampuan komputasi dan pemrosesan yang terbatas, *embedded computing board* mampu untuk memenuhi tugas dalam memproses sensor yang digunakan, dan menghasilkan output berupa suara melalui *external sound card* yang disambungkan ke perangkat suara (*headphone / earphone*).

(a)

(b)

(c)

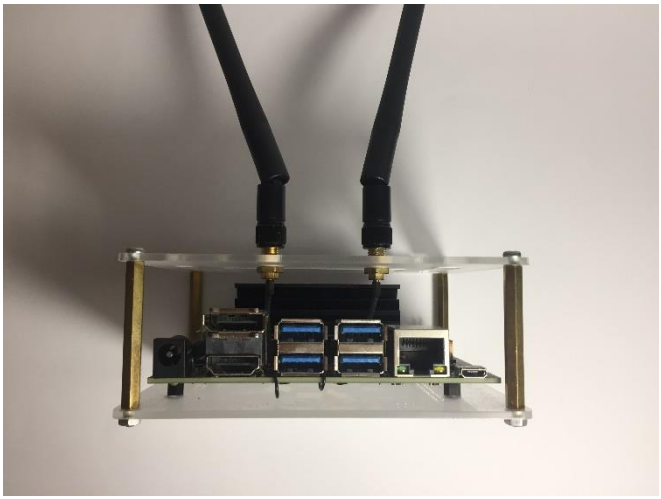
Gambar 3.8 RealSense yang digunakan; (a) Tampak depan; (b) Tampak belakang; (c) Tampak samping

Dalam pengaplikasiannya, Jetson™ memiliki performa yang sangat terbatas sehingga banyak keterbatasan-keterbatasan komputasi dan pemrosesan pada alat ini. Keterbatasan ini menyebabkan adanya kesulitan untuk mengembangkan perangkat ini dalam sisi fitur yang memerlukan komputasi dan pemrosesan yang lebih banyak dan kompleks. Untuk mengatasi hal tersebut, penggunaan *single-board computer* yang lebih *powerful* dapat menggantikan Jetson™ Nano ini dalam pengembangan yang lebih lanjut.

Dari sisi kelebihan, Jetson Nano memiliki *port* USB 3.0. Hal ini diperlukan untuk meng-*interface* RealSense™ D435i yang membutuhkan koneksi USB 3.0 Meskipun belum memiliki *Wireless Network Card* dan *Sound Card / Audio Jack* yang tertanam, penggunaan *Wireless Network* dapat dilakukan dengan menambahkan *Wireless Network Card* pada slot M.2 Key E yang tersedia pada Development Kit dan juga *Sound Card* pada slot USB.

Berikut pada **Gambar 3.9** dan **Gambar 3.10** merupakan foto dari Jetson™ Nano yang digunakan dalam penelitian ini.

Gambar 3.9 Jetson™ Nano yang digunakan



Gambar 3.10 Port dari Jetson™ Nano

3.4.3 Power Supply

Dalam penggunaan RealSense™ D435i dan Jetson™ Nano, diperlukan sumber daya yang mencukupi agar perangkat dapat bekerja dengan baik. Besar daya yang diperlukan oleh RealSense™ dan Jetson™ Nano bersifat variable tergantung dengan proses yang dilakukan, dengan daya antara 5 hingga 10 watt. Daya untuk RealSense™ diberikan melalui port USB 3.0 yang sekaligus berfungsi sebagai transmisi dan komunikasi data. Karena ditancapkan pada Jetson™ Nano, maka Jetson™ Nano membutuhkan tenaga untuk menghidupkan perangkatnya sendiri serta memiliki beban untuk menyediakan daya bagi RealSense™.

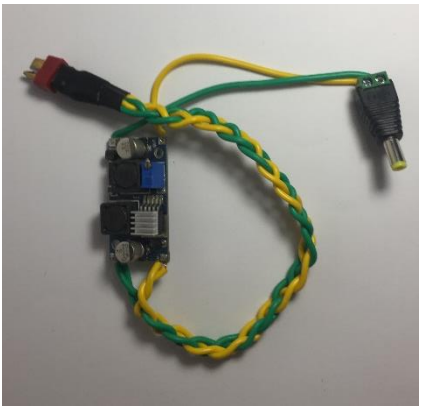
Pada Jetson™ Nano, terdapat 2 cara untuk memberi daya. Pertama, yaitu melalui port Micro USB, daya dapat diberikan dengan tegangan 5V, tetapi hanya memiliki batas arus maksimal sebesar 2A, sedangkan Jetson™ Nano memiliki daya maksimal 10W. Seharusnya daya cukup jika kita menghitung daya dari tegangan 5V dan arus 2A, akan tetapi banyak perangkat tambahan yang memerlukan daya seperti kamera, *mouse*, *keyboard*, dan lain sebagainya. Alternatif kedua yaitu melalui socket DC *barrel jack* (diameter luar 5.5mm, diameter dalam 2.1mm, Panjang 9.5mm). Dengan cara ini, Jetson™ Nano dapat menerima tegangan 5V dengan arus maksimal 4A. Untuk memberi daya dengan DC *barrel jack*, diperlukan *jumper* pada pin J48. Pemberian sumber daya

melalui MicroUSB tidak dapat digabung dengan DC *barrel jack* karena diperlukan jumper untuk memilih sumber daya dari salah satu *port/socket*.

Pada rancangan alat ini, dipergunakan sumber tenaga yang berasal dari baterai LiPo 3 sel dengan tegangan 9-12.6 volt. Karena tegangan yang diperlukan oleh Jetson™ Nano adalah tegangan 5V, maka digunakan *Step-Down Module* untuk menurunkan tegangan dari baterai ke nominal 5V. Untuk mencapai arus yang cukup, maka keluaran 5V dari Step-Down Module disambungkan melalui DC *barrel jack* ke Jetson™ Nano untuk menyediakan daya yang cukup bagi keseluruhan perangkat yang ada.

Berikut pada **Gambar 3.11** dan **Gambar 3.12** merupakan foto dari baterai LiPo dan *DC Step-Down Module* yang digunakan dalam penelitian ini. Seperti terlihat pada gambar, keluaran dari *DC Step-Module* dihubungkan ke DC *barrel jack* dan masukannya berasal dari baterai LiPo dengan konektor T-Plug atau Dean.

Gambar 3.11 Baterai LiPo yang digunakan



Gambar 3.12 *DC Step-Down Module* yang digunakan

3.4.4 Sound Card & Speaker

Seperti diketahui sebelumnya, Jetson™ Nano tidak memiliki *sound card* bawaan layaknya *single-board computer* pada umumnya. Pada sistem yang dirancang, tentunya diperlukan untuk mengeluarkan suara yang dihasilkan oleh pengolahan data. Karena hal itu, diperlukan *sound card external*. Pada perancangan perangkat keras, digunakan *sound card external* dengan port USB untuk mempermudah penggunaan. Dari *sound card external* ini kemudian dihubungkan ke *speaker* dengan *audio jack* 3.5mm pada umumnya. *Sound card external* yang digunakan pada sistem ini seperti pada **Gambar 3.13**.

Gambar 3.13 *Sound card external* yang digunakan

Untuk transduser suara digunakan *speaker* stereo. Untuk mempermudah penggunaan, maka dipilih perangkat berupa *headphone* atau *earphone* sebagai *output* suara. Nantinya *headphone* atau *earphone* akan dipasangkan seperti biasa pada pengguna alat ini.

3.5 Perancangan Perangkat Lunak

Pada penelitian ini, perangkat lunak memiliki peran yang paling penting untuk fungsionalitas alat. Data dari sensor dalam bentuk apapun akan tidak berguna jika tidak dapat diolah dan disampaikan dengan baik. Sesuai dengan blok diagram sistem, perangkat lunak akan dibagi menjadi lima langkah yang utama untuk menjalankan sistem. Tiap langkah akan dijelaskan dalam subsubbab selanjutnya.

3.5.1 Pengambilan Data RealSense™

Pengambilan informasi baik gambar kedalaman, gambar warna, maupun data IMU dapat diambil dari RealSense™ dengan menggunakan SDK yang disediakan oleh Intel® yang dikembangkan secara *open source*.

Intel® RealSense™ SDK 2.0 yang diberi nama *librealsense* sudah dilengkapi dengan *library* antar platform yang dapat digunakan untuk berbagai produk kamera kedalaman RealSense™ [32]. Pada penelitian ini, digunakan Python *wrapper* dari *librealsense* dalam lingkungan pemrograman Python 3.0.

Pertama, untuk mengakses API, perlu dilakukan pemanggilan *package librealsense* yang dilakukan dengan memanggil perintah

```
import pyrealsense as rs
```

Digunakan *namespace* `rs` sebagai metode untuk memanggil fungsi dan objek yang terdapat pada *librealsense*.

Setelah itu untuk melakukan pengambilan data, diperlukan sebuah koneksi antara program yang dibuat dengan RealSense™ yang disebut dengan *pipeline*. Data yang ingin diambil beserta semua konfigurasinya dapat diatur dalam *pipeline* agar kita dapat menerima informasi sesuai dengan apa yang kita inginkan. Secara teoritis, kita dapat membuat satu atau lebih *pipeline* dengan satu atau lebih RealSense™ untuk mengakses informasi yang kita perlukan, dengan batasan jika sebuah *pipeline* sudah mengakses sebuah *stream* (contohnya *stream* gambar kedalaman) dalam satu kamera, maka *pipeline* yang lain tidak bisa mengakses *stream* gambar kedalaman pada RealSense™ tersebut dalam waktu yang bersamaan. Tetapi setelah percobaan, didapatkan hasil bahwa terdapat masalah saat membuah lebih dari satu *pipeline* untuk RealSense™ yang sama meskipun data yang diambil dari *stream* yang berbeda. Untuk permasalahan ini berasal dari *librealsense* dan merupakan *bug* [33] yang sudah diketahui.

Pada cuplikan kode berikut adalah *pipeline* yang digunakan dalam program Python beserta konfigurasinya:

Terlihat bahwa pada konfigurasi cuplikan kode, dilakukan *stream* untuk gambar kedalaman dengan resolusi 640 x 480 dan *frame rate* 30 fps, gambar warna dengan resolusi dan *frame rate* yang sama dengan gambar kedalaman, dan data IMU berupa *gyroscope* dengan *frame rate* 200 fps serta *accelerometer* dengan *frame rate* 63 fps.

```
pipeline = rs.pipeline()
config = rs.config()
config.enable_stream(rs.stream.depth, 640,
      360, rs.format.z16, 30)
config.enable_stream(rs.stream.color, 640,
      360, rs.format.bgr8, 30)
config.enable_stream(rs.stream.accel,
      rs.format.motion_xyz32f, 63)
config.enable_stream(rs.stream.gyro,
      rs.format.motion_xyz32f, 200)
profile = pipeline.start(config)
```

Selanjutnya adalah menerima informasi yang diberikan oleh RealSense™. Terdapat 2 teknik dalam menerima informasi yang ada, yaitu dengan *wait_for_frames()* atau *poll_for_frames()*, dan *callback function*. Pada *wait_for_frames()*, kita menunggu *frameset* (*frameset* merupakan objek pada Python yang berisikan sejumlah *frame*, dimana tiap *frame* dapat berisi informasi gambar kedalaman atau gambar warna atau IMU). Setelah *frameset* diterima maka akan disimpan pada variable dan program akan dilanjutkan. Seperti hal nya *wait_for_frames()*, *poll_for_frames()* juga mengambil *frameset* tetapi tidak menunggu untuk *frameset* yang baru, melainkan mengecek jika ada *frameset* baru jika tidak maka akan mengambil *frameset* sebelumnya yang belum diambil datanya. Metode ini sangat praktis untuk menerima data dari RealSense™, akan tetapi memiliki kekurangan yaitu jika terdapat beberapa *stream* yang tidak sama *frame rate*-nya, maka harus menunggu semua data *stream* tersedia baru akan diambil dalam bentuk *frameset*. Selain itu jika program kita memiliki waktu pengulangan yang cukup lama, maka akan mempengaruhi kecepatan data yang diambil sehingga tidak dapat diperoleh *frame rate* yang dinginkan sesuai spesifikasi kita.

Untuk mengatasi ini maka digunakan metode *callback function*. Pada metode ini, setiap ada *frame* baru yang tersedia maka akan langsung diproses dengan cara memanggil fungsi *callback* yang ditentukan. Proses ini bersifat *concurrent*, dimana saat program kita berjalan dimana saja dan kapan saja, setiap ada *frame* maka fungsi *callback* akan dipanggil dan tidak mempengaruhi proses utama dalam program kita. Dengan demikian, dapat diperoleh *frame rate* yang hampir sama dengan spesifikasi yang kita inginkan.

Data yang diambil sesuai konfigurasi adalah:

1. Gambar kedalaman (*depth image*) dengan resolusi 640x360 piksel dan *frame rate* 30 fps dalam format integer 16 bit 1 channel (Z16).
2. Gambar warna (*color image*) dengan resolusi 640x360 piksel dan *frame rate* 30 fps dalam format integer 8 bit 3 channel untuk *blue*, *green*, dan *red* (BGR8)
3. Informasi akselerasi atau percepatan (dalam m/s^2) dengan *frame rate* 63 fps untuk sumbu X, Y, dan Z.
4. Informasi kecepatan sudut (dalam radian/detik) dengan *frame rate* 200 fps untuk sumbu X, Y, dan Z.

3.5.2 Pengolahan Citra Kedalaman

Produk Intel® RealSense™ sudah dilengkapi dengan API yang cukup mudah digunakan untuk berbagai keperluan, baik dengan GUI maupun dalam bentuk *library*, untuk mengambil data baik citra kedalaman maupun citra warna, dalam satuan yang baku (milimeter untuk citra kedalaman dan 8-bit RGB untuk citra warna). Akan tetapi, hasil yang didapatkan belum dapat dipergunakan secara langsung. Diperlukan pengolahan citra agar gambar dapat digunakan untuk langkah selanjutnya. Ada beberapa masalah yang terdapat pada gambar kedalaman yang akan dijelaskan secara terperinci beserta cara untuk menanggulanginya, antara lain citra kedalaman dan citra warna yang memiliki sudut pandang yang berbeda, adanya nilai kedalaman yang tidak terbaca, serta bagaimana mengambil nilai yang merepresentasikan suatu daerah dengan ukuran tertentu pada gambar kedalaman.

3.5.2.1 Align

Pada Intel® RealSense™ API, kita dapat menentukan resolusi dari gambar yang ingin kita ambil, baik gambar kedalaman maupun gambar warna. Resolusi ini tentunya sesuai dengan spesifikasi kamera yang digunakan, pada penelitian yang menggunakan Intel® RealSense™ sesuai dengan **Tabel 2.3**. Resolusi yang tidak sama antara gambar kedalaman dan gambar warna tentunya menyebabkan nilai piksel di lokasi yang sama pada kedua gambar merepresentasikan titik yang berbeda. Akan tetapi, meskipun resolusi dari gambar kedalaman dan gambar warna sama, tidak berarti nilai piksel di lokasi yang sama pada kedua gambar merepresentasikan titik yang sama. Perbedaan ini disebabkan karena sudut pandang kedua gambar yang berbeda, sehingga nilai dari kedua gambar tidak dapat digabungkan begitu saja, contohnya dengan menambah sebuah *channel* pada gambar warna dan memasukkan

gambar kedalaman pada *channel* tersebut sebagai nilai kedalaman, akan menyebabkan kesalahan yang fatal. Maka dari itu, perlu dilakukan sebuah pengolahan untuk mencari lokasi piksel dari kedua gambar yang merepresentasikan titik yang sama.

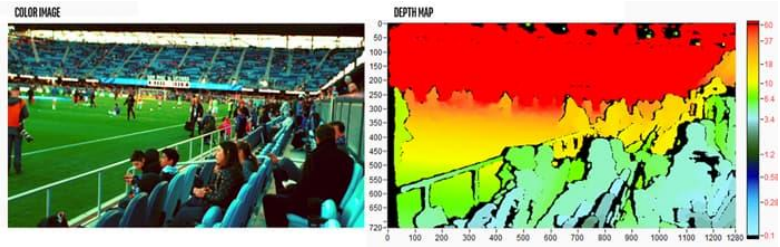
Align adalah istilah yang sering digunakan untuk membuat sebuah gambar baru dimana nilai piksel yang sama merepresentasikan titik yang sama. Tentunya diperlukan algoritma khusus seperti *feature detection* dan transformasi linear untuk menggabungkan nilai kedua gambar. Pada penelitian ini, digunakan fitur *align* yang disediakan dalam API *librealsense* untuk *python wrapper* dengan fungsi seperti pada cuplikan kode berikut:

```
align = rs.align(rs.stream.color)
```

Align yang dilakukan adalah dengan menyamakan nilai kedalaman pada gambar warna sebagai acuan. Untuk melakukan *align* diperlukan parameter-parameter dari pengaturan gambar kedalaman yang didapatkan melalui perintah pada cuplikan kode berikut.

```
depth_sensor =
    profile.get_device().first_depth_sensor()
depth_scale = depth_sensor.get_depth_scale()
```

Pada gambar X terlihat kedua gambar kedalaman dan warna untuk objek yang sama dengan resolusi yang sama, akan tetapi tidak dilakukan *align*. Setelah dilakukan *align*, maka akan terbentuk gambar yang memiliki lokasi piksel yang sama untuk titik objek yang sama, seperti pada gambar X. Kedua gambar pada **Gambar 3.14** kemudian dapat disatukan menjadi sebuah gambar dengan 4 *channels*, untuk masing-masing nilai warna merah (*red*), warna hijau (*green*), warna biru (*blue*), dan nilai kedalaman (*depth*) atau RGB-D.



Gambar 3.14 Gambar yang akan disatukan menjadi gambar RGB-D

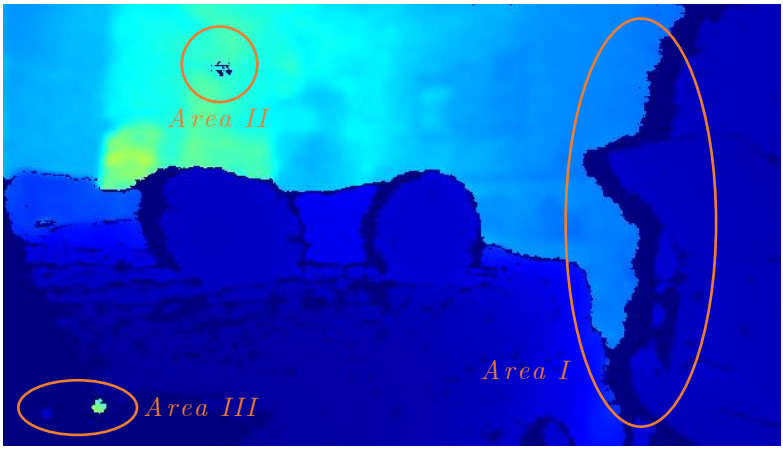
3.5.2.2 Morphological Transformation

Gambar kedalaman yang digunakan merupakan gambar kedalaman dengan format Z16, yaitu gambar 1 *channel* dengan nilai ukuran 16-bit yang memiliki nilai *unsigned integer*, dimana nilai *integer* sesuai dengan jarak antara kamera dengan titik tersebut dalam satuan milimeter (mm). Data yang didapatkan akan sulit divisualisasikan jika tidak dilakukan *scalling* Maka dari itu, dilakukan scaling dan *colormapping* untuk memudahkan visualisasi.

Untuk memvisualisasikan gambar kedalaman seperti dijelaskan pada tinjauan pustaka, digunakan kode Python seperti pada kode berikut.

```
# Scalling
depth_scaleabs = cv2.convertScaleAbs(
                   depth_image,
                   alpha=0.03
                 )
# Colormapping
depth_colormap = cv2.applyColorMap(
                   depth_scaleabs,
                   cv2.COLORMAP_JET
                 )
# Show Image
cv2.imshow("COLORMAP", depth_colormap)
```

Pada tahap ini, gambar yang diperoleh sudah ter-*align*, kemudian kita lihat salah satu contoh gambar kedalaman dalam *colormap* pada **Gambar 3.15**.



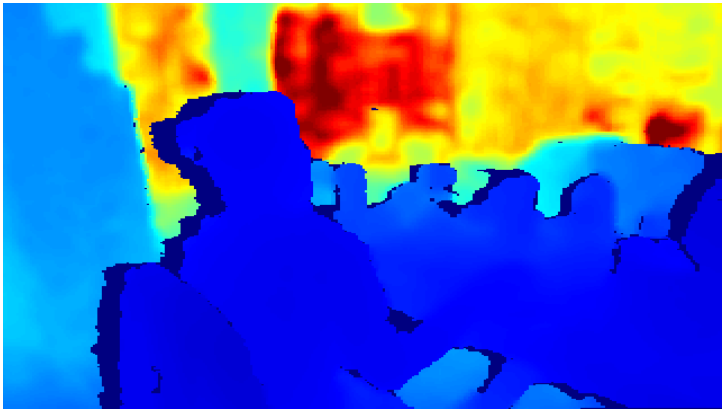
Gambar 3.15 Gambar kedalaman dengan *colormap* yang belum diolah

Terlihat pada **Gambar 3.15** terdapat beberapa area yang memiliki nilai yang tidak sesuai. Pertama yaitu *Area I*, dimana nilai tersebut merupakan nilai kedalaman yang tidak dapat terbaca dari sebuah objek. Penyebab dari tidak dapat terbacanya nilai pada *Area I* disebabkan adanya titik yang terlihat dari salah satu *stereo depth imaging* tetapi tidak terlihat dari *stereo depth imaging* yang lain, sehingga titik tersebut tidak dapat dikalkukasikan jarak kedalamannya. Dua buah sensor untuk melakukan *imaging* tentu memiliki sudut pandang yang berbeda, sesuai dengan jarak *baseline* yang ditentukan, agar dapat dihitung nilai kedalamannya, sehingga titik yang tidak terbaca ini tidak dapat dihindari keberadaannya. Nilai yang tidak terbaca ini akan bernilai nol (0) pada gambar kedalaman.

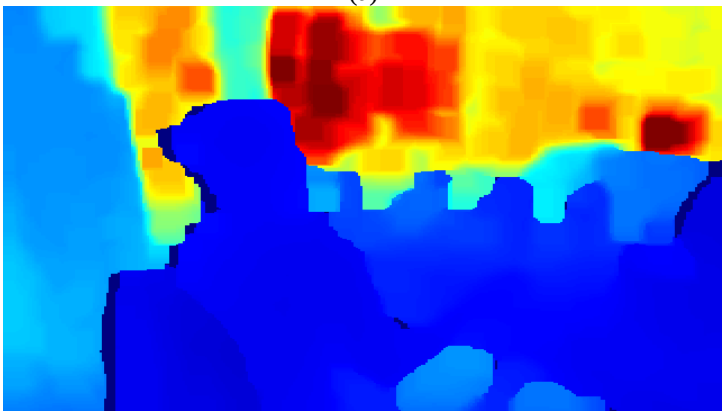
Area selanjutnya pada **Gambar 3.15** adalah *Area II*. Pada *Area II* terlihat titik berwarna gelap yang merupakan nilai nol (0). Nilai yang bernilai nol ini merupakan *noise* yang terjadi akibat berbagai hal lain seperti sinar inframerah (IR) tidak dapat tertangkap oleh sensor, atau adanya gangguan-gangguan lain. Nilai yang bernilai 0 ini berada pada area yang seharusnya memiliki nilai kedalaman yang cukup jauh, sehingga akan terjadi kesalahan menafsiran nilai jika diambil nilai rata-rata atau nilai minimum untuk merepresentasikan suatu area atau mengecilkan gambar (*pooling*).

Seperti halnya *Area II*, pada *Area III* **Gambar 3.15** terjadi *noise* berupa nilai yang sangat tinggi pada area yang seharusnya bernilai rendah atau bahkan nol (0). Nilai ini dapat berasal dari berbagai gangguan,

terutama jika objek berada pada batas yang lebih dekat dari nilai minimum kedalaman yang dapat dibaca. Sama halnya dengan *Area II*, nilai ini akan mengganggu perhitungan jika diambil nilai rata-rata atau nilai minimum untuk mengecilkan gambar.



(a)



(b)

Gambar 3.16 Perbandingan hasil gambar kedalaman dengan *morphological transformation*; (a) sebelum; (b) sesudah

Untuk mengatasi berbagai *noise* dan gangguan yang lain pada gambar kedalaman tersebut, maka diperlukan pengolahan citra yang disebut *morphological transformation*. Transformasi ini secara dasar ada

dua, yaitu erosi dan dilasi. Pada gambar kedalaman, untuk menghilangkan *noise* dan mengatasi nilai yang tidak terbaca, dilakukan proses kombinasi erosi dan dilasi dengan urutan dilasi, kemudian erosi dengan iterasi 2x, dilanjutkan dengan dilasi kembali dengan iterasi 2x.

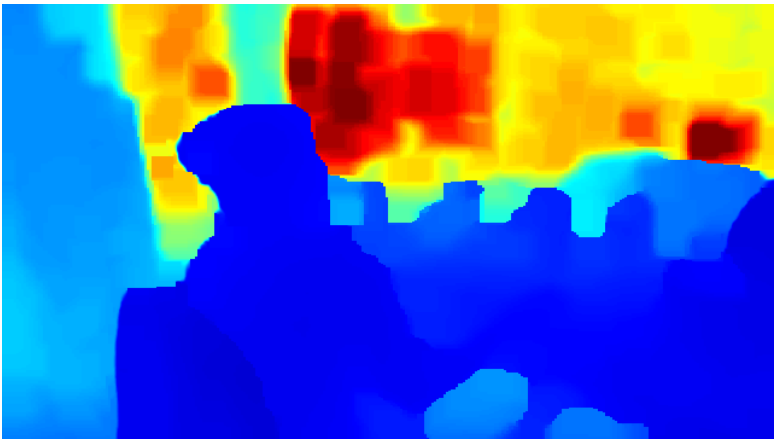
Hasil dari kombinasi *morphological transformation* akan seperti pada **Gambar 3.16**. Terlihat bahwa pada **Gambar 3.16** (b) sudah tidak terdapat *noise* dan nilai yang tidak terbaca berkurang areanya jika dibandingkan sebelumnya pada.**Gambar 3.16** (a).

3.5.2.3 Inpaint

Dengan adanya *morphological transformation*, area yang tidak terbaca berkurang, akan tetapi tidak menghilangkan area tersebut. Nilai dari area tersebut masih ada yang bernilai nol (0). Seperti sebelumnya, nilai ini akan mengganggu perhitungan jika diambil nilai rata-rata atau nilai minimum untuk merepresentasikan suatu area atau mengecilkan gambar (*pooling*). Sebagai contoh, terdapat sebuah tembok dengan jarak kedalaman 1 meter atau 1000 mm yang didepannya terdapat sebuah objek dengan jarak kedalaman 0.5 meter atau 500 mm. Jika terdapat nilai yang tidak terbaca pada tepi objek, maka aka nada area dengan nilai 0 di antara nilai 1000 dan 500. Jika diambil nilai rata-rata pada area yang terdapat nilai 1000, 500, dan 0, maka nilai yang didapat tidak akan merepresentasikan area tersebut akibat nilai 0 yang bisa dikatakan sebagai *outlier*. Maka dari itu diperlukan sebuah algoritma untuk menghilangkan nilai tersebut.

Dari berbagai gambar kedalaman yang diambil, digunakan berbagai jenis algoritma seperti filter untuk nilai diferensial yang tinggi, *blurring*, dan sebagainya. Setelah melakukan percobaan, didapatkan kesimpulan jika algoritma berdasarkan aturan (*rule-based*) kurang sesuai untuk aplikasi *real-time*, dimana gambar terus-menerus diambil dalam jangka waktu tertentu. Waktu untuk pemrosesan yang cepat dibutuhkan, sehingga untuk aplikasi ini digunakan metode *inpaint* yang disediakan oleh OpenCV untuk menghilangkan nilai nol ini.

Pertama dilakukan *masking* untuk mengetahui lokasi piksel yang memiliki nilai nol. Kemudian masking ini digunakan sebagai acuan area yang perlu dilakukan *inpaint*, hingga didapatkan hasil dimana nilai yang tidak terbaca ini hilang dan tersamarkan dengan objek di sekitarnya. Contoh hasil dari *inpaint* adalah seperti pada **Gambar 3.17**.



Gambar 3.17 Gambar kedalaman yang telah diolah dengan *inpaint*

3.5.2.4 IMU Based Stabilizer

Untuk melengkapi fungsi dari fitur IMU yang terdapat pada Intel® RealSense™ D435i, maka gambar akan lebih baik jika diberi *stabilizer*. Pada langkah ini, gambar yang telah diolah akan distabilisasi menggunakan informasi yang didapat dari IMU berupa akselerasi linear dan percepatan sudut. Stabilisasi dilakukan pada sumbu *roll* untuk mengkompensasi guncangan-guncangan yang terjadi karena cara berjalan penggunanya.

Pertama, untuk mendapatkan nilai *roll* diperlukan perhitungan dari sensor yang tersedia, yaitu akselerometer dan *gyroscope*. Nilai yang didapatkan adalah percepatan linear (m/s^2) dan kecepatan sudut (rad/s). Seperti dijelaskan sebelumnya, karena nilai *frame rate* yang berbeda, digunakan fungsi *callback* untuk menyimpan nilai setiap kali data sudah tersedia. Ketika data percepatan linear dan kecepatan sudut sudah didapatkan keduanya, akan dilakukan perhitungan *roll*, *pitch*, dan *yaw*. Cuplikan kode berikut adalah program untuk menghitung nilai *roll*, *pitch*, dan *yaw*.

```
# Timestamp dari Gyroscope untuk menghitung dt
ts_gyro = gyro_frame.get_timestamp()

# Menyimpan nilai Gyroscope dan Accelerometer
accel_data = accel_frame.as_motion_frame()
             .get_motion_data()
gyro_data = gyro_frame.as_motion_frame()
            .get_motion_data()

# Menghitung roll dan pitch dari accelerometer
resultant = np.math.sqrt(accel_data.y*accel_data.y
            + accel_data.z*accel_data.z)
accel_roll = np.math.atan2(accel_data.x, resultant)
accel_pitch = np.math.atan2(accel_data.y,
              accel_data.z)

# Menghitung roll pitch yaw dari gabungan IMU
if last_ts_gyro is not None:
    dt = (ts_gyro - last_ts_gyro) / 1000.0
    theta[0] = (theta[0] + gyro_data.x * dt)*alpha
               + accel_pitch*(1 - alpha)
    theta[1] = (theta[1] - gyro_data.y * dt
    theta[2] = (theta[2] - gyro_data.z * dt)*alpha
               + accel_roll * (1 - alpha)
else:
    theta[0] = accel_pitch
    theta[1] = np.math.pi
    theta[2] = accel_roll
    last_ts_gyro = ts_gyro
```

Untuk nilai theta[0] merupakan nilai *pitch*, nilai theta[1] adalah *yaw*, dan theta[2] untuk *roll*. Arah dari *pitch*, *yaw*, dan *roll* terhadap RealSense™ ditunjukkan pada **Gambar 2.12**.

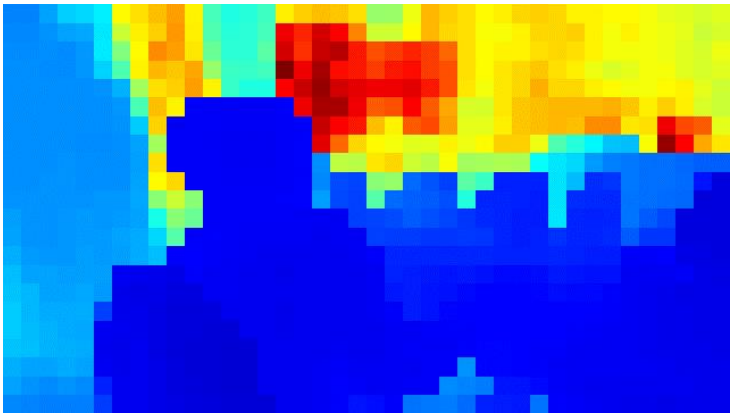
Setelah didapatkan nilai *roll* akan digunakan untuk melakukan pemutaran gambar agar tetap stabil saat digunakan saat berjalan. Dengan bantuan *package imutils*, gambar dapat dengan mudah diputar sesuai dengan sudut yang diberikan oleh *roll*. Akan tetapi perlu diingat jika sudut yang dihitung pada **Error! Reference source not found.** dalam satuan radian, sedangkan fungsi dari *imutils* menerima parameter sudut dalam satuan derajat. Maka dari itu perlu dilakukan perhitungan nilai sudut dapat derajat terlebih dahulu. Berikut adalah kode perhitungan yang dilakukan beserta fungsi untuk melakukan pemutaran gambar.

```
# Mengkonversi nilai sudut - rad ke derajat
rotate_angle = theta[2]*180/np.math.pi
# Rotate sesuai Yaw
depth_image = imutils.rotate(depth_image,
              rotate_angle)
```

3.5.3 Deteksi Rintangan dan Posisinya

Tahap ini sangat penting untuk menentukan spesifikasi visual dari rancangan sistem. Dari gambar yang telah diolah, akan didapatkan hasil seperti pada **Gambar 3.17**. Selanjutnya gambar akan diproses untuk mendeteksi rintangan yang ada. Pada penelitian ini, rintangan dibatasi posisinya dalam 3 bagian, yaitu yang pertama depan kiri, kedua depan tengah, dan ketiga depan kanan. Tiap posisi merepresentasikan area dengan sejumlah piksel pada gambar kedalaman.

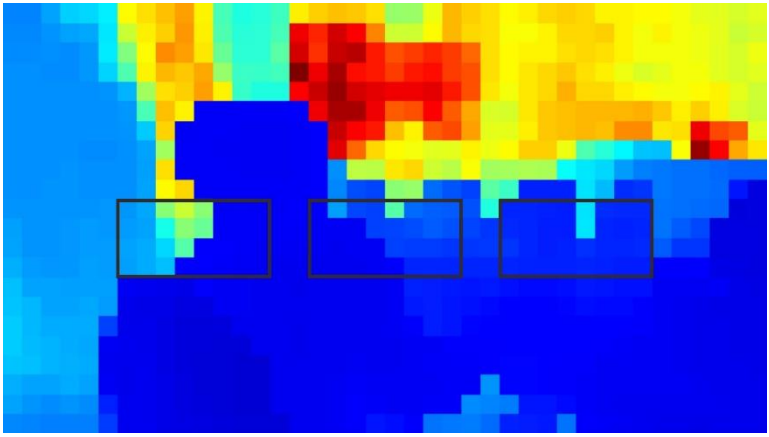
Pertama untuk mempermudah pengambilan nilai yang merepresentasikan suatu area atau beberapa piksel, dilakukan pengecilan resolusi gambar atau *resize*. Gambar yang semula berukuran 640 x 360 piksel akan dikecilkan menjadi 40 x 22 piksel. Pengambilan nilai yang merepresentasikan area tiap 16 x 16 piksel adalah dengan menggunakan *minpooling* dimana setiap area 16 x 16 piksel diambil nilai paling kecil sebagai nilai piksel yang baru. Hasil dari *resize* seperti pada **Gambar 3.18**.



Gambar 3.18 Gambar kedalaman setelah dilakukan *resize*

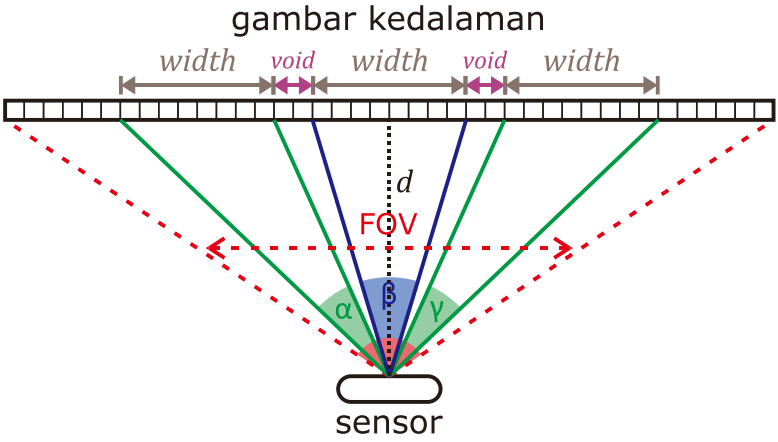
Setelah didapatkan informasi kedalaman dalam ukuran 40 x 22 piksel, kemudian diambil daerah yang ingin diambil atau *region of interest*. Sesuai dengan posisi yang telah ditentukan, pertama adalah

posisi depan kiri. Posisi ini akan merepresentasikan area pada piksel baris 11 hingga 14 dan kolom 7 hingga 14. Selanjutnya untuk posisi depan tengah diambil baris 11 hingga 14 dan kolom 17 hingga 24. Terakhir untuk posisi depan kanan diambil piksel baris 11 hingga 14 dan kolom 27 hingga 34. Masing masing dari area berukuran 8 x 4 piksel. Area ditunjukkan pada **Gambar 3.19**.



Gambar 3.19 Area dari tiga posisi yang ditentukan

Untuk spesifikasi dari posisi, direpresentasikan dalam rentang sudut. Berikut pada **Gambar 3.20** adalah ilustrasi besar sudut tiap posisinya dengan perhitungan sudut berdasarkan FOV. Untuk nilai FOV didapatkan dari **persamaan (2.6)** atau dapat dilakukan pengujian secara manual dengan mengukur lebar objek yang datar pada gambar kedalaman dan jarak antara kamera kedalaman dengan objek tersebut.



Gambar 3.20 Ilustrasi besar sudut tiap posisi pada gambar kedalaman

$$\angle\frac{\beta}{2} = \tan^{-1}\frac{\frac{1}{2}\cdot width}{d}$$
$$\angle\beta = 2\cdot\tan^{-1}\frac{\frac{1}{2}\cdot width}{d} \quad \textbf{(3.1)}$$

$$\angle\alpha = \angle\gamma = \tan^{-1}\frac{\frac{3}{2}\cdot width + void}{d} - \tan^{-1}\frac{\frac{1}{2}\cdot width + void}{d} \quad \textbf{(3.2)}$$

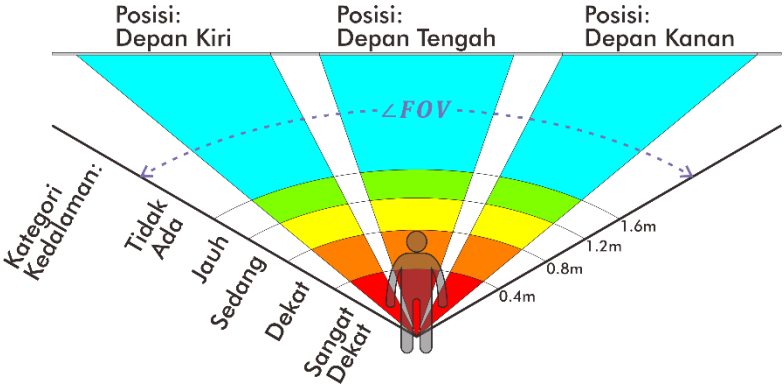
Sudut α merupakan besar sudut untuk posisi depan kiri, sudut β untuk posisi depan tengah, dan sudut γ untuk posisi depan kanan. Karena simetris, maka nilai sudut α sama dengan sudut γ. Perlu diingat bahwa perhitungan ini adalah untuk sudut horisontal. Untuk sudut vertikal sangat dipengaruhi oleh cara dan posisi penggunaan alat.

Selanjutnya area tiap posisi yang berukuran 8 x 4 piksel diambil nilai paling rendahnya (*minpooling*) untuk merepresentasikan area tersebut. Dari ketiga posisi ini akan didapatkan tiga nilai yang merepresentasikan berturur turut depan kiri, depan tengah, dan depan kanan.

Untuk jarak kedalaman dalam penelitian ini ditentkan jarak kedalaman minimum yaitu 40 cm, dan jarak kedalaman maksimum yaitu 160 cm. Maksud dari jarak kedalaman minimum adalah jika objek berada

lebih dekat dari jarak kedalaman minimum maka objek akan dianggap sangat dekat hingga batas minimum pengukuran kedalaman kamera. Maksud dari jarak kedalaman maksimum adalah jika objek berada lebih jauh dari jarak kedalaman maksimum, maka objek akan diabaikan atau dianggap tidak ada objek. Nilai ini dapat diubah dan diatur sesuai dengan kebutuhan dan kenyamanan tiap penggunanya, akan tetapi dalam penelitian ini digunakan nilai yang ditentukan tersebut. Jarak juga ditentukan berdasarkan ketelitian kedalaman dalam sistem ini. Pada penelitian ini ditentukan ketelitian kedalaman sisten sebesar 40 cm, yang berarti informasi kedalaman dalam rentang 40 cm akan dianggap sama. Dengan demikian maka akan ada 5 kategori. Pertama yaitu kedalaman diluar batas maksimum kedalaman (160 cm) yang tidak terdeteksi. Kedua yaitu kedalaman antara batas maksimum hingga ketelitian kedalaman dalam hal ini antara 120 cm hingga 160 cm. Ketiga yaitu antara 80 cm hingga 120 cm. Keempat antara 40 cm hingga 80 cm. Terakhir adalah kedalaman yang lebih kecil dari batas minimum kedalaman (40 cm).

Pada tahap ini data yang didapatkan berupa tiga informasi kedalaman (depan kiri, depan tengah, dan depan kanan) yang telah digolongkan ke kategorinya masing-masing (dalam hal ini kategori 1 hingga 5). Data ini membentuk spesifikasi visual dari rancangan sistem sesuai dengan **Gambar 3.21**.



Gambar 3.21 Spesifikasi visual dari rancangan sistem

3.5.4 Representasi Informasi Kedalaman dalam Parameter Audio

Dari langkah sebelumnya, diketahui bahwa terdapat lima kategori, dimana kategori yang pertama merupakan kedalaman yang tidak

dianggap rintangan dan yang terakhir kategori yang kurang dari batas minimum. Informasi ini kemudian diubah dalam parameter audio.

Parameter audio yang digunakan dalam penelitian ini ada 3, yang pertama adalah audio stereo, yaitu parameter kekerasan suara (volume) untuk dua *channel* yang merupakan suara kiri dan suara kanan. Kedua adalah parameter frekuensi fundamental dari nada yang dihasilkan. Kemudian ketiga adalah parameter waktu, yaitu durasi dari nada yang dihasilkan. Durasi ini dapat berupa lamanya nada dimainkan, dapat juga berupa durasi diam (*silence*) antar nada, sehingga bisa dihasilkan berbagai pola nada.

Hal terpenting dari langkah ini adalah memilih parameter yang tepat untuk merepresentasikan informasi kedalaman yang ada. Pada percobaan ini, melalui *try and error* didapatkan pengaturan parameter sebagai berikut:

1. Untuk audio stereo, digunakan pengaturan volume untuk *channel* kanan dan *channel* kiri. Ketika objek berada pada depan kiri, maka audio akan memberikan volume penuh pada *channel* kiri. Ketika objek berada pada depan tengah, maka audio akan memberikan volume setengah pada *channel* kanan dan *channel* kiri. Begitu juga ketika objek berapa pada depan kanan, maka audio akan memberikan volume penuh pada *channel* kanan.
2. Untuk pemilihan frekuensi, setelah melalui proses *try and error* digunakan tiga jenis frekuensi yang merupakan nada C5, D5, dan E5. Masing masing frekuensi adalah 523.25 Hz, 587.33 Hz, dan 659.26 Hz untuk merepresentasikan posisi masing-masing depan kiri, depan tengah, dan depan kanan. Diperoleh pengamatan bahwa penggunaan nada do re mi lebih mudah dipahami dan diingat oleh pengguna.
3. Untuk durasi, digunakan pola *beep… beep…* untuk memberikan persepsi jarak pada penggunanya. Semakin jauh objek maka *beep* akan semakin lambat, begitu juga semakin dekat objek maka *beep* akan semakin cepat dan jeda antar *beep* juga semakin cepat. Karena telah ditentukan ada 5 kategori, dan untuk kategori pertama dianggap tidak ada objek, maka ada 4 pola suara *beep… beep…* yang masing-masing merepresentasikan kategori jarak kedalaman. Pada penulisan laporan ini, untuk mempermudah penyebutan kategori kedalaman, maka digunakan penamaan sederhana yaitu “tidak ada objek” untuk kategori kedalaman lebih jauh dari batas maksimum, selanjutnya berturut-turut “jauh”, “sedang”, “dekat”, dan terakhir “sangat dekat” untuk kategori dimana objek lebih dekat dari batas minimum.

Selanjutnya parameter suara yang telah didapatkan dari informasi kedalaman akan diproses untuk menghasilkan suara. Parameter ini masing-masing akan digunakan dalam memanggil fungsi yang bertugas menciptakan sampel nada yang diinginkan, dan kemudian digabungkan.

3.5.5 Pembuatan Sampel Suara

Dalam pembuatan suara, pertama hal yang dilakukan adalah membuat sampel. Pada penelitian ini, digunakan jenis sampel sinuioid, dengan persamaan sesuai **Persamaan (2.3)**. Digunakan *sampling rate* 44100 Hz. Pemrograman pada Python untuk membuat sampel suara seperti pada cuplikan kode berikut.

```
fs = 44100  # sampling rate (Hz)
duration = 1  # duration of sampe (s)
volume = 0.8  # 0.00-1.00

# Phase Calculation
phase = 2*np.pi*np.arange(fs*duration)*
        frequency/fs

# Sample Calculation
sample = volume*np.sin(phase).astype(
         np.float32)

# Keep the last phase
last_phase = phase[-1]-(int(phase[-1]/
             (2*np.pi))*2*np.pi)
```

Jika diperhatikan dalam cuplikan kode terdapat variable untuk menyimpan fase terakhir. Hal ini dilakukan untuk menjadi acuan fase dari sampel selanjutnya yang akan digabungkan. Penggunaan fase terakhir ini digunakan agar tidak terdapat diskontinuitas antara satu sampel dengan sampel selanjutnya. Jika terdapat diskontinuitas, maka akan terdengan suara klik pada saat audio dikeluarkan di titik diskontinuitas tersebut. Maka dari itu penting untuk melakukan penyimpanan nilai fase terakhir.

Selanjutnya adalah pembuatan harmonisa. Seperti kita tahu dalam sebuah suara secara alami sangat jarang ditemui suara sinusoid murni. Hampir dalam semua suara terdapat harmonisa. Harmonisa adalah sinyal dengan frekuensi kelipatan bilangan bulat dari frekuensi fundamentalnya. Pada beberapa aplikasi, harmonisa merupakan sebuah gangguan. Dalam

dunia musik, harmonisa sendiri sangat dimanfaatkan untuk menghasilkan harmoni dan membedakan satu alat musik dengan yang lainnya, juga membuat sebuah nada lebih kaya dan nyaman didengar. Untuk itu, pada penelitian ini digunakan harmonisa agar suara sinusoid nyaman didengarkan oleh penggunanya.

Terdapat dua buah harmonisa yang digunakan, yang pertama harmonisa dengan frekuensi 0.5 kali dari frekuensi fundamental. Dalam dunia musik, kelipatan frekuensi dua kali adalah sebuah kenaikan oktaf, sehingga frekuensi 0.5 berarti nada satu oktaf di bawah frekuensi fundamental. Kedua adalah harmonisa dengan frekuensi 2 kali dari frekuensi fundamentalnya. Frekuensi ini merupakan nada dengan satu oktaf lebih tinggi dari frekuensi fundamentalnya.

Selanjutnya adalah *edge smoothing*. Sampel dari langkah sebelumnya sudah dapat dikeluarkan menjadi sebuah audio, akan tetapi jika didengarkan maka aka nada sebuah suara klik pada awal dan akhir audio. Hal ini disebabkan oleh perubahan yang kurang halus pada amplitudo suara. Untuk mengurani suara yang kurang nyaman ini, dilakukan *edge smoothing*. *Edge smoothing* dilakukan dengan meningkatkan amplitudo dari nol sampai volume yang dinginkan dengan linear dalam waktu tertentu untuk awalan dan sebaliknya untuk akhiran. Kombinasi waktu transisi yang tepat akan menghilangkan suara klik yang ada sebelumnya.

Setelah didapat sampel audio, perlu diingat bahwa sinyal tersebut hanya memiliki satu *channel* atau mono. Untuk mendapatkan suara stereofonik dengan amplitudo yang berbeda antara suara kiri dengan suara kanan, perlu dilakukan perhitungan masing-masing *channel* untuk menghasilkan sampel yang berbeda. Sampel kemudian disusun menggunakan fungsi *column_stack((left_sample, right_sample))* untuk menghasilkan sampel yang baru dengan dua *channel*.

BAB IV
PENGUJIAN DAN ANALISIS

Pada bab ini, hasil dari rancangan sistem diimplementasikan dan diujikan secara *real* pada subjek dan dilakukan analisis.

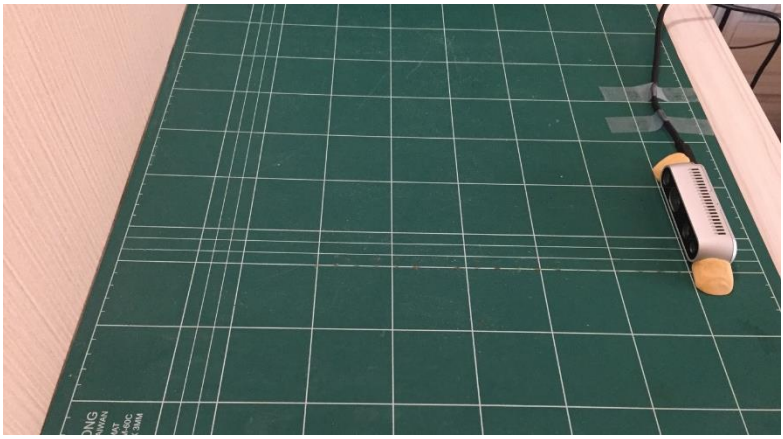
4.1 Pengujian Spesifikasi Sensor

Untuk mengetahui apakah sensor yang digunakan sesuai dengan spesifikasi, dan untuk menguji apakah hasil pembacaan kedalaman pada sensor akurat atau tidak (beserta nilai galat), maka sensor berupa kamera kedalaman Intel® RealSense™ D435i perlu diuji.

4.1.1 Pengujian Nilai FOV

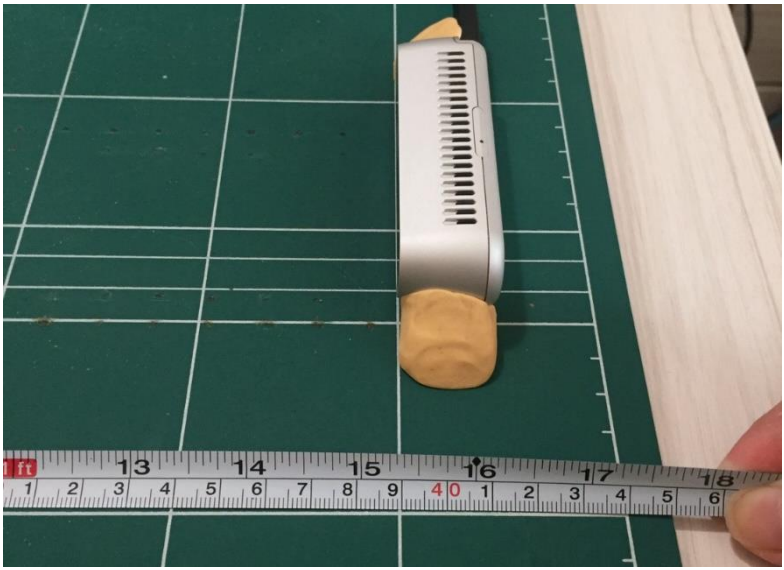
Pertama sebelum melakukan pengujian alat dilakukan pengujian nilai FOV oleh kamera kedalaman yang digunakan. Untuk mengukur FOV (*Field of View*) seperti diilustrasikan pada **Gambar 2.13**, kamera diletakkan tegak lurus dengan bidang yang datar seperti pada **Gambar 4.1** dan **Gambar 4.2**.

Gambar 4.1 Peletakkan kamera pada bidang datar tampak belakang



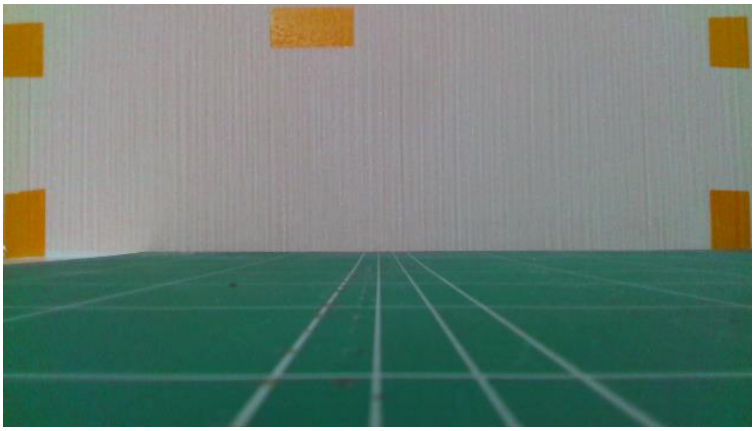
Gambar 4.2 Peletakkan kamera pada bidang datar tampak samping

Jarak antara bagian depan kamera dengan bidang tembok didepannya adalah 39 cm, ditunjukkan oleh pengukuran pada **Gambar 4.3**.

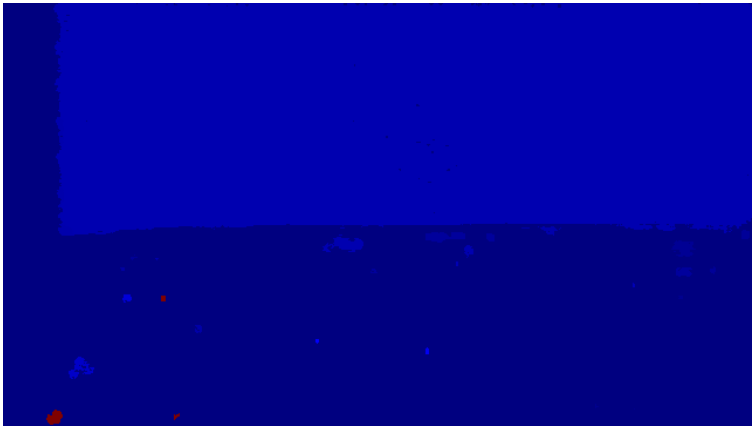


Gambar 4.3 Jarak antara kamera dengan bidang tembok di depannya

Setelah itu diambil data gambar kedalaman dan warna dari RealSense™. Hasil gambar kedalaman dan warna ditunjuukan pada **Gambar 4.4** dan **Gambar 4.5**.



Gambar 4.4 Gambar warna hasil pengujian

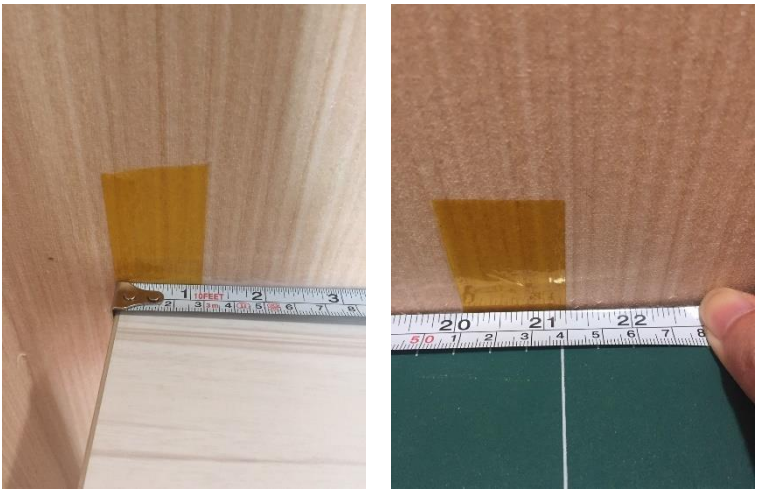


Gambar 4.5 Gambar kedalaman hasil pengujian

Terlihat pada **Gambar 4.5** terdapat *invalid depth band* di sebelah kiri. Untuk mengatasi *invalid depth band* tersebut, dilakukan proses *align* gambar kedalaman pada gambar warna sehingga didapatkan hasil gambar kedalaman *aligned* seperti pada **Gambar 4.6**.

Gambar 4.6 Gambar kedalaman *aligned* hasil pengujian

Setelah itu dicari jarak dari ujung kiri ke ujung kanan untuk mengetahui lebar pandang, kemudian dapat dihitung sudut FOV dari nilai lebar pandang pada bidang tembok dan jarak antara kamera dengan bidang tembok. Dari pengujian, didapatkan nilai lebar pandang pada bidang tembok sebesar 54 cm sesuai pada **Gambar 4.7**.



(a) (b)

Gambar 4.7 Lebar pandang pada bidang tembok; (a) Ujung kiri; (b) Ujung kanan

Dari data diatas, maka dapat dihitung nilai FOV hasil pengujian dengan rumus segitiga sama kaki:

$$FOV = 2 \cdot \tan^{-1} \frac{\frac{1}{2} l}{d} \tag{4.1}$$

$$FOV = 2 \cdot \tan^{-1} \frac{\frac{1}{2} \cdot 54\ cm}{39\ cm} = 1.211\ rad = 69.39°$$

Dimana l merupakan lebar pandang pada bidang tembok dan d adalah jarak antara sensor dengan bidang tembok.

Dari perhitungan maka diperoleh nilai FOV sebesar **1.211 radian** atau **69.39°**. Hal ini sesuai dengan spesifikasi dari rentang pandang gambar RGB pada **Tabel 2.2** yang menunjukkan nilai rentang pandang horisontal 69.4°±3°. FOV tidak sesuai dengan rentang pandang horisontal dari gambar kedalaman dikarenakan sebenarnya pada gambar kedalaman terdapat *valid depth band* disebelah kiri dan tidak ter-*align* dengan gambar warna. Ketika dilakukan proses *align* gambar kedalaman akan dipetakan pada gambar RGB sehingga rentang pandangnya menjadi sama dengan gambar RGB.

4.1.2 Pengujian Nilai Kedalaman

Selanjutnya hal yang terpending dari kamera kedalaman adalah nilai kedalaman itu sendiri. Pengujian ini dilakukan untuk mengetahui apakah pembacaan nilai kedalaman dari RealSense® cukup akurat atau seberapa nilai galat dari pembacaan nilai kedalaman.

Untuk mengetahui hal ini, dilakukan pengujian pembacaan nilai kedalaman atau jarak antara sensor dengan suatu objek dengan jarak nyata yang diketahui atau diukur dengan alat ukur lain. Perbandingan nilai antara pembacaan kedalaman dengan pengukuran yang lain yang handal akan menunjukkan keakuratan dan nilai galat dari kamera kedalaman yang digunakan

Pertama, dilakukan peletakkan kamera RealSense® pada suatu bidang atau benda yang kokoh untuk menahan RealSense® sehingga tidak mudah tergoyang dan berubah posisi / arah. Setelah itu diletakkan sebuah benda dengan bidang yang datar sejajar dengan permukaan depan kamera seperti pada **Gambar 4.8**.

Gambar 4.8 Posisi pengujian nilai kedalaman

Benda pada **Gambar 4.8** diubah-ubah jaraknya terhadap sensor dan diukur jaraknya menggunakan *rangefinder* CP-100S dengan spesifikasi pengukuran jarak antara 0.05 meter hingga 100 meter (tipe laser 635 nm, < 1 mW) seperti pada **Gambar 4.9**.

Gambar 4.9 *Rangefinder* untuk mengukur jarak

Berikutnya, pada setiap posisi dari benda, gambar kedalaman diambil dari RealSense® dan dilakukan pembacaan nilai kedalaman antara sensor dengan permukaan objek tersebut. Perlu diingat bahwa permukaan objek datar dan sejajar dengan permukaan depan kamera. Nilai kedalaman diambil pada titik tengah dari objek yang terbaca pada gambar kedalaman. Dari pengujian ini, didapatkan nilai kedalaman dari pembacaan Intel® RealSense™ sesuai dengan **Tabel 4.1**.

Tabel 4.1 Pengujian nilai kedalaman dari kamera kedalaman terhadap *rangefinder*

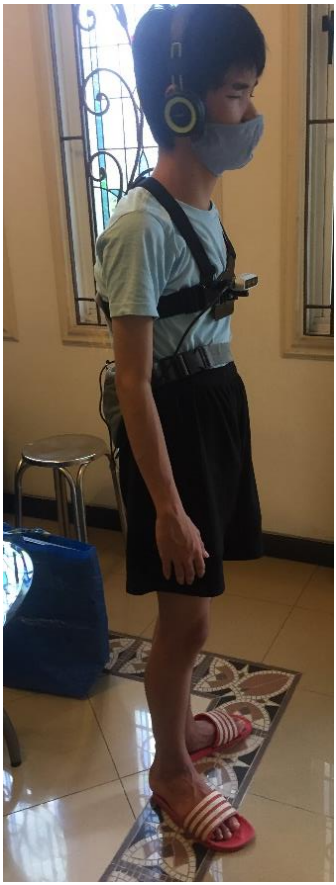
Percobaan	**Nilai Kedalaman diukur dengan *rangefinder***	**Nilai Kedalaman dari kamera kedalaman**	**Nilai Galat Absolut**
1	200 mm	204 mm	4 mm
2	300 mm	302 mm	2 mm
3	400 mm	408 mm	8 mm
4	500 mm	498 mm	2 mm
5	600 mm	596 mm	4 mm
6	700 mm	702 mm	2 mm
7	800 mm	800 mm	0 mm
8	900 mm	906 mm	6 mm
9	1000 mm	996 mm	4 mm
10	1200 mm	1200 mm	0 mm
11	1400 mm	1404 mm	4 mm
12	1600 mm	1600 mm	0 mm
13	1800 mm	1802 mm	2 mm
14	2000 mm	2002 mm	2 mm
		Rata-rata:	2.86 mm

Dari pengujian yang telah dilakukan, didapatkan nilai rata-rata galat absolut atau biasa disebut MAE (*Mean Absolute Error*) sebesar **2.86 mm**. Dari nilai ini, pembacaan nilai kedalaman dari kamera kedalaman RealSense® dapat dikatakan cukup akurat untuk aplikasi pada sistem dari penelitian ini. Nilai galat ini lebih dari cukup untuk mendeteksi jarak antara subjek tunanetra dengan rintangan yang ada didepannya, dimana sebenarnya penglihatan manusia normal memiliki ketelitian jarak yang jauh lebih buruk tetapi masih dapat memperkirakan jarak rintangan tersebut dan menghindarinya.

4.2 Pemasangan Alat Bantu Navigasi pada Tunanetra

Rancangan alat bantu navigasi untuk tunanetra yang telah dirangkai dipasangkan pada tunanetra untuk mengetahui hasil desain. Hasil dari pemasangan alat bantu navigasi pada subjek tunanetra adalah seperti pada **Gambar 4.10**, **Gambar 4.11**, dan **Gambar 4.12**.

Gambar 4.10 Tampak depan pemakaian alat



Gambar 4.11 Tampak samping pemakaian alat

Gambar 4.12 Tampak belakang pemakaian alat

Pada **Gambar 4.10** adalah tampak depan pemasangan, **Gambar 4.11** untuk tampak samping, dan **Gambar 4.12** merupakan tampak belakang pemakaian alat. Dari pemasangan ini, diuji portabilitas, kenyamanan, dan kemudahan penggunaan dari perangkat pada subjek tunanetra. Setelah penggunaan beberapa saat, subjek tunanetra diminta untuk memberikan penilaian terhadap nilai *portability*, *comfortability*, dan *ease of use* dengan beberapa pertanyaan sesuai pada **Tabel 4.2**.

Tabel 4.2 Daftar pertanyaan kualitatif penggunaan alat

No. Pertanyaan	Pertanyaan	Aspek yang Dinilai
I	Apakah alat nyaman ketika digunakan?	*Comfortability*
II	Apakah alat memberatkan ketika digunakan?	*Portability*
III	Apakah alat dapat digunakan sehari-hari?	*Portability & Comfortability*
IV	Apakah ada kesulitan ketika menggunakan alat tersebut?	*Ease of use*
V	Apakah ada kesulitan ketika melepaskan alat tersebut?	*Ease of use*
VI	Apakah alat ketika digunakan membatasi gerak?	*Comfortability*
VII	Apakah penggunaan *headphone* / *earphone* mengganggu?	*Comfortability*

Jawaban dari pertanyaan pada **Tabel 4.2** digolongkan menjadi jawaban positif, jawaban negatif, atau jawaban netral. Jawaban yang merupakan jawaban positif meliputi nyaman ketika digunakan, tidak memberatkan, mudah digunakan sehari-hari, tidak ada kesulitan dalam menggunakan maupun melepas alat, tidak membatasi gerak, tidak mengganggu, dan sejenisnya. Untuk jawaban yang berupakan jawaban negatif meliputi tidak nyaman ketika digunakan, memberatkan, sulit digunakan, tidak bisa digunakan sehari-hari, sulit untuk memasang dan / atau melepas alat, alat membatasi pergerakan, pemakaian *headphone* / *earphone* menganggu. Jawaban yang merupakan jawaban netral adalah jawaban yang tidak termasuk jawaban positif maupun negative meliputi jawaban dengan alasan atau syarat tertentu seperti alat nyaman digunakan tapi dalam jangka waktu tertentu, pemakaian alat tidak membatasi gerak jika dipakai pada saat-saat tertentu saja, dan sejenisnya.

Dari percobaan pemakaian yang dilakukan ke dua subjek tunanetra (data pribadi dari subjek berupa jenis kelamin, umur, dan tingkat kepekaan cahaya, dan penyebab gangguan penglihatan terdapat pada lampiran), diperoleh hasil pada **Tabel 4.3**:

Tabel 4.3 Hasil penilaian kualitatif penggunaan alat oleh subjek tunanetra

No. Pertanyaan	Aspek yang Dinilai	Subjek 1	Subjek 2
I	*Comfortability*	Positif	Positif
II	*Portability*	Positif	Positif
III	*Portability & Comfortability*	Positif	Positif
IV	*Ease of use*	Positif	Netral
V	*Ease of use*	Positif	Netral
VI	*Comfortability*	Positif	Positif
VII	*Comfortability*	Positif	Positif

Selain pertanyaan yang bersifat kualitatif, subjek juga diminta untuk memberikan penilaian dalam bentuk kuantitatif untuk nilai *portability*, *comfortability*, dan *ease of use*. Subjek diminta untuk memberi penilaian dalam bentuk angka antara satu hingga sepuluh (1-10) dengan nilai 1 paling rendah dan nilai 10 paling tinggi.

Hasil dari penilaian kuantitatif dari dua subjek tunanetra diperolah pada **Tabel 4.4**.

Tabel 4.4 Hasil penilaian kuantitatif penggunaan alat oleh subjek tunanetra

Aspek yang Dinilai	Subjek 1	Subjek 2
Portability	10	10
Comfortability	10	9
Ease of use	9	9

Dari hasil yang pengujian pemasangan perangkat keras yang dilakukan, diperoleh nilai *portability*, *comfortability*, dan *ease of use* **rata-rata positif** oleh kedua subjek.

Selanjutnya untuk penilaian yang bersifat kuantitatif dengan rentang penilaian satu hingga sepuluh (1-10) dengan nilai 1 paling rendah dan nilai 10 paling tinggi, diperoleh rata-rata nilai **10** untuk *portability*, **9.5** untuk *comfortability*, dan **9** untuk *ease of use*.

4.3 Pengujian Fungsionalitas Alat

Sistem secara keseluruhan berupa alat bantu navigasi untuk tunanetra diujikan pada subjek tunanetra secara fungsionalitas.

4.3.1 Deteksi Posisi Rintangan

Pertama sesuai dengan spesifikasi visual rancangan sistem (merujuk pada **Gambar 3.21**) , deteksi rintangan dikelompokkan menjadi tiga area, yaitu rintangan di depan kiri, rintangan di depan tengah, dan rintangan di depan kanan. Dari tiga area tersebut, diujikan masing-masing rintangan pada masing-masing area secara satu per satu dan juga kombinasi untuk mengetahui apakah subjek tunanetra dapat mengetahui ada tidak nya rintangan pada area tersebut.

Pengujian dilakukan dengan kombinasi peletakkan rintangan pada area sesuai **Tabel 4.5**.

Tabel 4.5 Kombinasi peletakkan rintangan untuk pengujian

Letak Benda		
Depan Kiri	**Depan Tengah**	**Depan Kanan**
Tidak Ada	Tidak Ada	Tidak Ada
Ada	Tidak Ada	Tidak Ada
Tidak Ada	Ada	Tidak Ada
Tidak Ada	Tidak Ada	Ada
Ada	Ada	Tidak Ada
Ada	Tidak Ada	Ada
Tidak Ada	Ada	Ada
Ada	Ada	Ada

Untuk mengukur tingkat keberhasilan pada pengujian ini, digunakan pengukuran akurasi dan *F1 score*. Pertama-tama dari hasil pengujian akan dihitung nilai tiap pengujian sesuai dengan matriks *confusion* pada **Tabel 4.6**.

True positive (TP) merupakan kondisi dimana terdapat rintangan dan dapat dideteksi oleh subjek yang diuji. *True negative* (TN) yaitu kondisi dimana tidak terdapat rintangan dan subjek juga tidak mendeteksi adanya rintangan. Selanjutnya yaitu *false negative* (FN) dimana subjek tidak mendeteksi adanya rintangan padahal sebenarnya ada rintangan. Sebaliknya untuk *false positive* (FP), subjek mendeteksi adanya rintangan padahal kenyataanya tidak ada rintangan.

Tabel 4.6 Matriks *confusion*

		Prediksi Rintangan oleh Subjek	
		Ada	Tidak Ada
Kenyataan Rintangan	Ada	*True Positive* (TP)	*False Negative* (FN)
	Tidak Ada	*False Positive* (FP)	*True Negative* (TN)

Kemudian setelah didapatkan nilai dari tiap elemen pada matriks *confusion*, dicar nilai akurasi. Nilai akurasi adalah prediksi yang berhasil dilakukan oleh tunanetra dengan benar, dalam hal ini maka semua deteksi tunanetra baik ada maupun tidak ada rintangan yang benar dan sesuai dengan kenyataan terhadap semua percobaan.

Akurasi ada pengukuran yang sangat lazim digunakan untuk mengukur tingkat keberhasilan, akan tetapi dibutuhkan metrik lain untuk mengetahui *behaviour* dari sistem kita secara lebih spesifik. Maka dari itu digunakan metrik berikutnya yaitu *precision* dan *recall*. Secara umum, *precision* berarti seberapa banyak item yang dipilih yang relevan. Secara lebih spesifik, *precision* adalah seberapa banyak prediksi yang benar untuk semua prediksi positif dari sistem. Untuk *recall*, secara umum *recall* berarti seberapa banyak item yang relevan yang dipilih. Dalam hal ini *recall* mengindikasikan seberapa banyak prediksi positif yang benar dari semua kondisi positif yang sebenarnya.

Setelah didapatkan nilai *precision* dan *recall*, selanjutnya digunakan sebuah metrik untuk menggabungkan kedua nilai ini. *F1 score* adalah rata-rata harmonis dari *precision* dan *recall*. Karena dalam sistem ini bobot dari *precision* dan *recall* sama, maka digunakan F1.

Rumus perhitungan untuk akurasi, *precision*, *recall*, dan *F1 score* ditunjukkan pada persamaan berikut:

$$accuracy = \frac{True\ Prediction}{Total\ Population} = \frac{TP + TN}{TP + TN + FP + FN} \quad \textbf{(4.2)}$$

$$precision = \frac{True\ Positive}{All\ Positive\ Prediction} = \frac{TP}{TP + FP} \quad \textbf{(4.3)}$$

$$recall\ (sensitivity) = \frac{True\ Positive}{All\ Positive\ Condition} = \frac{TP}{TP + FN} \quad \textbf{(4.4)}$$

$$F1\ score = \frac{2}{recall^{-1} + precision^{-1}} = 2 \cdot \frac{precision \cdot recall}{precision + recall} \quad \textbf{(4.5)}$$

Dari pengujian yang dilakukan oleh dua subjek tunanetra, diperoleh hasil pada **Tabel 4.7**

Tabel 4.7 Hasil pengujian deteksi letak rintangan pada subjek tunanetra

Subjek Pengujian	**Subjek 1**	**Subjek 2**
***True Positive* (TP)**	12	9
***True Negative* (TN)**	10	10
***False Positive* (FP)**	2	4
***False Negative* (FN)**	0	3
Accuracy	0,92	0,73
Precision	0,86	0,69
Recall	1	0,75
F1 Score	0,9247	0,7188

Hasil dari pengujian ini, didapatkan maksimum akurasi sebesar **92.47%** yang berarti kesalahan deteksi rintangan hanya terjadi satu dari sepuluh percobaan. Dari nilai akurasi yang didapat, alat ini dapat berfungsi dengan baik untuk mendeteksi ada tidaknya dan posisi dari rintangan.

Untuk *precision* didapatkan nilai maksimal sebesar **86%**, dan *recall* maksimal sebesar **100%**. Dari kedua nilai tersebut didapatkan nilai *F1 score* sebesar **92%**.

4.3.2 Prediksi Jarak dan Posisi Rintangan

Selanjutnya, dilakukan pengujian untuk mengetahui perkiraan jarak antara subjek dengan rintangan yang ada. Sesuai dengan perancangan sisten, karena nilai kedalaman minimum dan maksimum yang dipilih adalah 0.4 meter dan 1.6 meter dengan ketelitian kedalaman pada rancangan sistem 0.4 meter, sehingga kedalaman dibagi menjadi 5 kategori, yaitu jarak diatas 1.6 meter yang dideteksi sebagai bukan rintangan, jarak antara 1.6 meter dan 1.2 meter, jarak antara 1.2 meter dan 0.8 meter, jarak antara 0.8 meter dan 0.4 meter, dan juga jarak yang lebih

dekat dari 0.4 meter. Spesifikasi visual rancangan sistem ini merujuk pada **Gambar 3.21**.

Karena jarak memiliki 5 kategori dan bukan merupakan klasifikasi biner, maka tidak dapat digunakan pengukuran akurasi dengan *F1 score* seperti pada pengujian sebelumnya. Pada dasarnya jarak merupakan nilai numerik. Pengkategorian nilai numerik berdasarkan rentang nilai tertentu disebut juga *bucketization*.

Ada berbagai metrik untuk mengukur keakuratan atau kesalahan (*error*) secara numerik menggunakan metrik jarak (*distance metrics*). Sebuah fungsi jarak yaitu $d(x, y)$ didefinisikan sebagai jarak antara elemen pada pasangan x dan y sebagai bilangan real bukan negatif. Jika jarak sama dengan nol, maka kedua elemen tersebut bernilai sama dalam metrik tersebut. Fungsi jarak digunakan untuk mengukur kedekatan dari dua elemen tersebut, dimana elemen dapat berupa angka, vektor, matriks, atau objek tertentu.

Pada penelitian ini digunakan metrik MAE (*Mean Absolute Error*) untuk nilai galat/kesalahan (*error*). Berikut fungsi jarak untuk menghitung MAE:

$$d_{MAE}: (x, y) \mapsto \frac{\|x - y\|_1}{n} = \frac{1}{n}\sum_{i=1}^{n}|x_i - y_i| \qquad \textbf{(4.6)}$$

Berikut adalah kategori dari jarak yang diujikan dalam bentuk tabel pada **Tabel 4.8**

Tabel 4.8 Pengkategorian jarak antara subjek dengan objek rintangan

Kategori	Jarak	
	Lebih Dari	Kurang Dari
1 (Sangat Dekat)	-	0.4 meter
2 (Dekat)	0.4 meter	0.8 meter
3 (Sedang)	0.8 meter	1.2 meter
4 (Jauh)	1.2 meter	1.6 meter
5 (Tidak Ada)	1.6 meter	-

Dari pengujian yang dilakukan pada dua subjek tunanetra, diperoleh hasil pada **Tabel 4.9**

Tabel 4.9 Hasil pengujian deteksi jarak dan lokasi rintangan pada subjek tunanetra

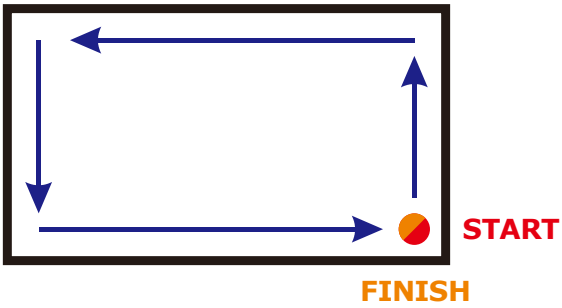
Subjek		Subjek 1	Subjek 2
Mean Absolute Error (MAE)	Kategori 1	1	1
	Kategori 2	1	2
	Kategori 3	1	3
	Kategori 4	1	1
	Kategori 5	0	0
	Rata-rata	0.8	1.6

Dari pengujian yang telah dilakukan, didapatkan nilai MAE atau *mean absolute error* terbaik sebesar **0.8**. Nilai ini cukup baik untuk tingkat kesalahan dalam memprediksi jarak. Nilai MAE yang kecil menandakan kesalahan deteksi jarak terjadi untuk perbedaan kategori yang dekat dan tidak terlalu jauh.

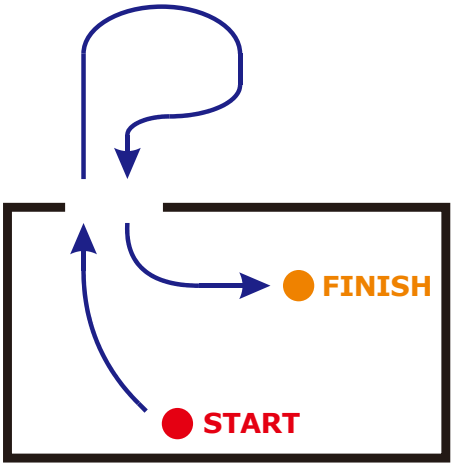
4.3.3 Pengujian Navigasi Tunanetra pada Jalur yang Ditentukan

Pada pengujian ini, subjek tunanetra mengaplikasikan alat bantu tunanetra dalam berjalan pada rute yang telah ditentukan, untuk mensimulasikan beberapa keadaan yang terjadi pada navigasi sehari-hari.

Dalam pengujian ini ada dua rute yang disediakan untuk subjek tunanetra bernavigasi. Rute seperti pada **Gambar 4.13**



(a)



(b)

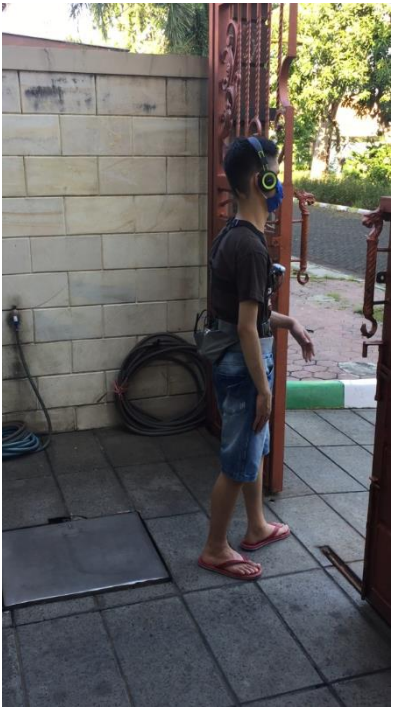
Gambar 4.13 Rute pengujian navigasi pada subjek tunanetra; (a) Rute pertama; (b) Rute kedua

Dalam pengujian ini, ada dua poin yang diuji. Pertama adalah apakah subjek tunanetra dapat menghindari tabrakan dengan tembok dengan berhenti tepat sebelum tembok tanpa alat bantu lainnya. Kedua adalah apakah subjek tunanetra dapat mengetahui cela di antara dua buah rintangan dan memasuki cela tersebut tanpa tertabrak. Cela yang dimaksud adalah seperti pintu yang terbuka atau pagar yang terbuka cukup untuk dimasuki dengan lancar.

Berikut pada **Gambar 4.14** dan **Gambar 4.15** adalah pengujian pada dua poin yang telah dijelaskan.

Dari pengujian diketahui bahwa subjek tunanetra dapat berhenti sebelum terjadi tabrakan dengan rintangan di depannya, dalam hal ini adalah tembok. Selain itu, subjek tunanetra juga dapat melewati cela di antara dua buah rintangan, dalam hal ini adalah cela pagar yang terbuka.

Gambar 4.14 Subjek tunanetra dapat berhenti sebelum tertabrak dengan tembok



Gambar 4.15 Subjek tunanetra dalam melewati cela pagar yang terbuka

BAB V
PENUTUP

Dari perancangan sistem, pengujian, dan analisis yang telah dilakukan, penelitian ini dapat dirangkum dalam sebuah kesimpulan. Hal-hal mengenai kendala dan kekurangan yang dihadapi selama proses penelitian ini akan ditulis dalam subbab saran untuk membantu penelitian selanjutnya.

5.1 Kesimpulan

Setelah melakukan pengujian dari perancangan sistem alat bantu navigasi untuk tunanetra menggunakan sensor RGB-D, dapat disimpulkan sebagai berikut:

1. Rancangan alat bantu navigasi dapat digunakan oleh tunanetra dengan relatif mudah, nyaman, dan tidak memberatkan ataupun membatasi aktivitas dari penggunanya.
2. Dari pengolahan gambar kedalaman, informasi kedalaman diambil dari area yang kecil dari gambar kedalaman, sehingga objek rintangan yang dapat dideteksi terbatas.
3. Pengujian fungsionalitas alat bantu navigasi menunjukkan akurasi pendeteksian rintangan dengan pembagian tiga posisi sebesar 92.47% dan kesalahan MAE (*mean absolute error*) perkiraaan jarak terharap rintangan sebesar 0.8 untuk pengaturan jarak rintangan yang kurang dari 1.6 meter dari penggunanya, serta dapat berhenti sebelum terjadi tabrakan dengan rintangan di depannya.

5.2 Saran

Dari proses perancangan sistem hingga pengujian, terdapat beberapa kendala dan kesulitan yang dapat menjadi saran sebagai berikut:

1. Pengambilan informasi kedalaman yang dilakukan hanya terbatas pada objek dengan ketinggian setara dengan sensor yang digunakan. Jika dapat dilakukan deteksi bidang lantai, alat bantu navigasi akan lebih handal dalam melakukan deteksi rintangan yang pendek seperti kardus atau anak tangga.
2. Pada penelitian ini digunakan CPU berupa Jetson™ Nano, dimana *embedded computing board* ini memiliki spesifikasi yang kurang untuk penambahan fitur atau deteksi objek yang lebih kompleks. Untuk selanjutnya mungkin dapat digunakan CPU yang lebih *powerful* dan memiliki kemampuan komputasi

yang tinggi tetapi tetap dalam ukuran yang *portable* dan konsumsi daya yang cukup dengan baterai.

3. Kamera kedalaman yang digunakan yaitu Intel® RealSense™ D435i dilengkapi dengan IMU (*Inertial Measurement Unit*). Tetapi dalam penelitian ini, IMU hanya digunakan sebagai *stabilizer* dari gambar yang didapatkan dan kurang dimaksimalkan fungsinya. Dengan menambahkan filter Kalman, hasil IMU dapat digunakan untuk penambahan fitur-fitur lainnya yang berguna bagi tunanetra.
4. Sistem yang dirancang pada penelitian ini menggunakan keluaran berupa audio stereo. Pada uji coba yang dilakukan, subjek tunanetra kesulitan untuk membedakan frekuensi suara dan membayangkan posisi spasial dari suara stereo. Pada penelitian selanjutnya dapat dicoba untuk penggunaan aktuator yang lain seperti *haptic*.

DAFTAR PUSTAKA

[1] D. Pascolini dan S. P. Mariotti, "Global estimates of visual impairment: 2010," *Br. J. Ophthalmol.*, vol. 96, no. 5, hal. 614–618, Mei 2012.

[2] D. A. L. Maberley *et al.*, "The prevalence of low vision and blindness in Canada," *Eye*, vol. 20, no. 3. Nature Publishing Group, hal. 341–346, 2006.

[3] "Daily Life Problems Faced by Blind People." [Daring]. Tersedia pada: https://wecapable.com/problems-faced-by-blind-people/. [Diakses: 30-Okt-2019].

[4] E. Khoirunisa dan D. Aries Himawanto, "The comparison of guide texture tiles for blind people in public areas between Surakarta and Nagoya city," *J. Kaji. Wil.*, vol. 9, no. 1, hal. 34, 2018.

[5] World Health Organization, "International Classification of Diseases for Mortality and Morbidity Statistics, 11th edition (ICD-11 MMS)," vol. 11, Apr 2019.

[6] S. Stevens, "Test distance vision using a Snellen chart," *Community Eye Heal. J.*, vol. 20, no. 63, hal. 52, 2007.

[7] N. K. Reefani, *Panduan Anak Berkebutuhan Khusus*. Yogyakarta: Imperium, 2013.

[8] A. Arora dan A. Shetty, "Common Problems Faced By Visually Impaired People," *Int. J. Sci. Res.*, vol. 3, no. 10, 2014.

[9] A. Parant, S. Schiano-Lomoriello, dan F. Marchan, "How would I live with a disability? Expectations of bio-psychosocial consequences and assistive technology use," *Disabil. Rehabil. Assist. Technol.*, vol. 12, no. 7, hal. 681–685, Okt 2017.

[10] "Adaptive Technology Versus Assistive Technology." [Daring]. Tersedia pada: https://www.assistivetech.com/adaptive-technology-versus-assistive-technology/. [Diakses: 28-Mei-2020].

[11] B. Bhanushali, A. Dhoot, P. Gandhi, dan K. Mehta, "Refreshable Braille Displays," *Int. J. Comput. Appl.*, vol. 180, no. 37, hal. 1–4, Apr 2018.

[12] I. Akbar dan A. F. Misman, "Research on semantics used in GPS based mobile phone applications for blind pedestrian navigation in an outdoor environment," in *Proceedings* -

International Conference on Information and Communication Technology for the Muslim World 2018, ICT4M 2018, 2018, hal. 196–201.

[13] Y. Shiizu, Y. Hirahara, K. Yanashima, dan K. Magatani, "The development of a white cane which navigates the visually impaired," in *Annual International Conference of the IEEE Engineering in Medicine and Biology - Proceedings*, 2007, hal. 5005–5008.

[14] I. P. Adi, "Rancang Bangun Sistem Pemetaan Halangan Pada Ruang Sebagai Alat Bantu Navigasi Tunanetra," Institut Teknologi Sepuluh Nopember Surabaya, 2019.

[15] S. Schmidt, C. Tinti, M. Fantino, I. C. Mammarella, dan C. Cornoldi, "Spatial representations in blind people: The role of strategies and mobility skills," *Acta Psychol. (Amst).*, vol. 142, no. 1, hal. 43–50, Jan 2013.

[16] L. B. Merabet dan A. Pascual-Leone, "Neural reorganization following sensory loss: The opportunity of change," *Nature Reviews Neuroscience*, vol. 11, no. 1. hal. 44–52, Jan-2010.

[17] M. Gori, G. Cappagli, G. Baud-Bovy, dan S. Finocchietti, "Shape Perception and Navigation in Blind Adults," *Front. Psychol.*, vol. 8, Jan 2017.

[18] C. E. Shannon, "Communication in the Presence of Noise," *Proc. IRE*, vol. 37, no. 1, hal. 10–21, 1949.

[19] IEC 61672-1, "International Standard IEC 61672-1 Edition 2.0 - Electroacoustics - Sound Level Meters," *Int. Electrotech. Commision*, 2013.

[20] A. M. Muntasir Rahman, S. Mahmud Khandaker, N. N. Saleheen, T. Nobi Afee, N. Afrin, dan M. A. Alam, "A portable braille refreshable display using micro servos," in *2018 Joint 7th International Conference on Informatics, Electronics and Vision and 2nd International Conference on Imaging, Vision and Pattern Recognition, ICIEV-IVPR 2018*, 2019, hal. 212–217.

[21] J. Yu dan K. Moeller, "Assistive Navigation Device for Visually Impaired—A Study on Reaction Time to Tactile Modality Stimuli," *Engineering*, vol. 05, no. 10, hal. 195–198, 2013.

[22] P. B. L. Meijer, "An Experimental System for Auditory Image Representations," *IEEE Trans. Biomed. Eng.*, vol. 39, no. 2, hal. 112–121, 1992.

[23] "The vOICe - New Frontiers in Sensory Substitution." [Daring].

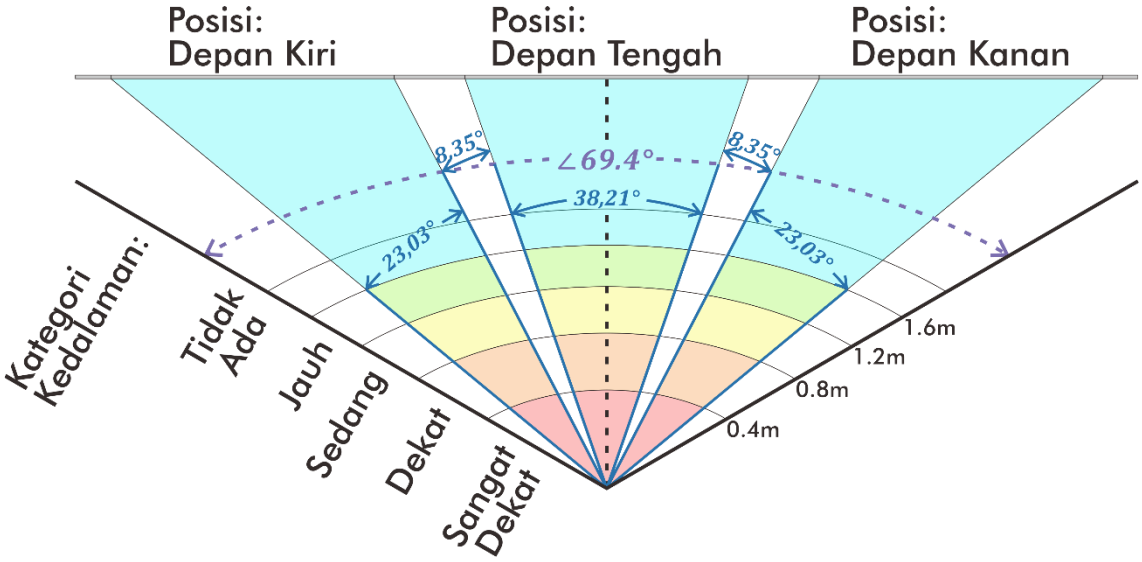
Tersedia pada: https://www.seeingwithsound.com/voicebme.html. [Diakses: 19-Nov-2019].
[24] "PMD Tech." [Daring]. Tersedia pada: https://www.pmdtec.com/. [Diakses: 26-Nov-2019].
[25] "SR4000 Data Sheet Rev. 4.9," 2011.
[26] "PrimeSense." [Daring]. Tersedia pada: https://www.primesense.com/. [Diakses: 26-Nov-2019].
[27] J. Garcia dan Z. Zalevsky, "US7433024B2 - Range mapping using speckle decorrelation," 2008.
[28] J. C. P. Villota dan A. H. R. Costa, "Aligning RGB-D Point Clouds through Adaptive Integration of Color and Depth Cues," in *Proceedings - 12th LARS Latin American Robotics Symposium and 3rd SBR Brazilian Robotics Symposium, LARS-SBR 2015 - Part of the Robotics Conferences 2015*, 2016, hal. 309–314.
[29] Intel, "Intel® RealSense™ Camera D400 series Product Family Datasheet Rev. 01/2019," 2019.
[30] "Python 3.8.3 Documentation." [Daring]. Tersedia pada: https://docs.python.org/3/. [Diakses: 03-Jun-2020].
[31] "Python/C API Reference Manual — Python 3.8.3 documentation." [Daring]. Tersedia pada: https://docs.python.org/3/c-api/index.html. [Diakses: 29-Mei-2020].
[32] "IntelRealSense/librealsense: Intel® RealSense™ SDK." [Daring]. Tersedia pada: https://github.com/IntelRealSense/librealsense. [Diakses: 03-Jun-2020].
[33] "Error Calibrating IMU and failed to receive IMU frames when stream Depth Color IMU at the same time · Issue #6370 · IntelRealSense/librealsense." [Daring]. Tersedia pada: https://github.com/IntelRealSense/librealsense/issues/6370. [Diakses: 03-Jun-2020].

Halaman ini sengaja dikosongkan

LAMPIRAN 1
SPESIFIKASI ALAT BANTU NAVIGASI

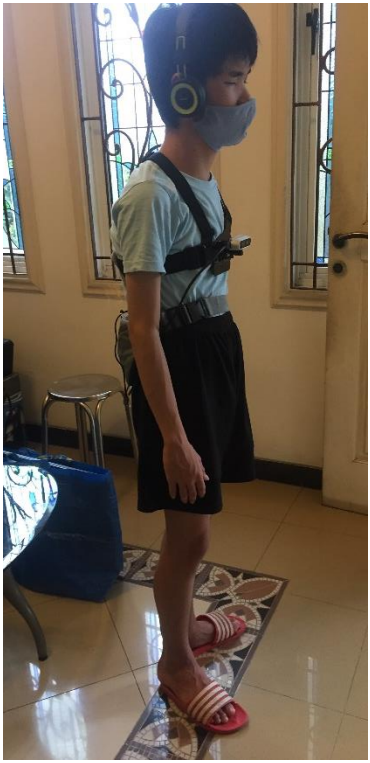
Spesifikasi Visual Perangkat:

Pembacaan Kedalaman Minimal Sensor	10.5 cm
Jarak Minimal Objek Rintangan	40 cm
Jarak Maksimal Objek Rintangan	160 cm
Pengkategorian Jarak Objek Rintangan (Lihat Ilustrasi)	< 40 cm (Sangat Dekat) 40 – 80 cm (Dekat) 80 – 120 cm (Sedang) 120 – 160 cm (Jauh) > 160 cm (Tidak Ada)
Rentang Pandang Horizontal (FOV)	69.4°±3°
Pembagian Posisi Rintangan (Lihat Ilustrasi)	Rentang 38.21° dengan sudut simetris terhadap garis tengah FOV (Depan Tengah) Rentang 23.03° dengan cela sudut 8.35° terhadap garis kanan posisi Depan Tengah (Depan Kanan) Rentang 23.03° dengan cela sudut 8.35° terhadap garis kiri posisi Depan Tengah (Depan Kiri)


Posisi:
Depan Kiri
Posisi:
Depan Tengah
Posisi:
Depan Kanan
8,35°
∠69.4°
8,35°
38,21°
23,03°
23,03°
Kategori
Kedalaman:
Tidak
Ada
Jauh
Sedang
Dekat
Sangat
Dekat
1.6m
1.2m
0.8m
0.4m

LAMPIRAN 2
BIODATA SUBJEK PENGUJIAN

Subjek 1:

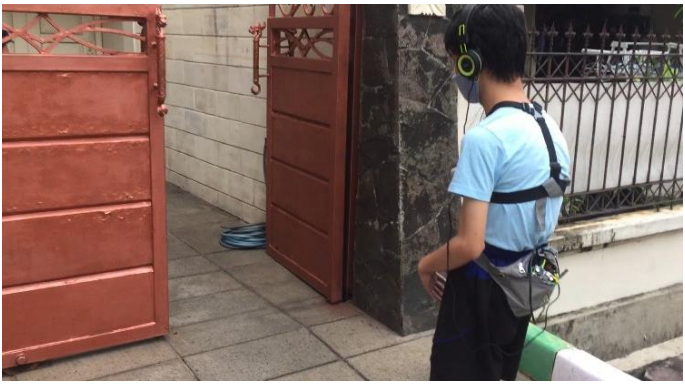


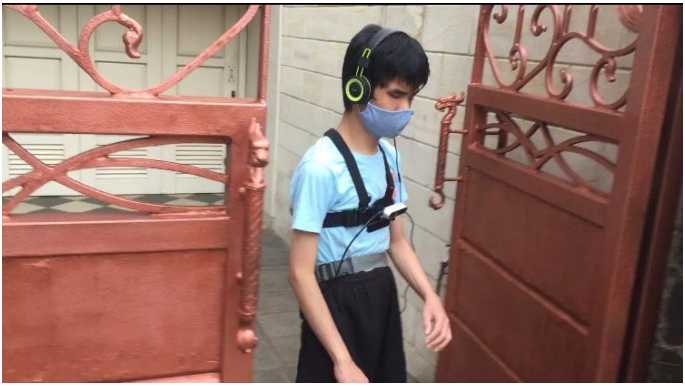
Jenis Kelamin	: Laki-laki
Umur	: 18 tahun
Tingkat gangguan penglihatan	: Kebutaan Total
Penyebab gangguan penglihatan	: Congenital

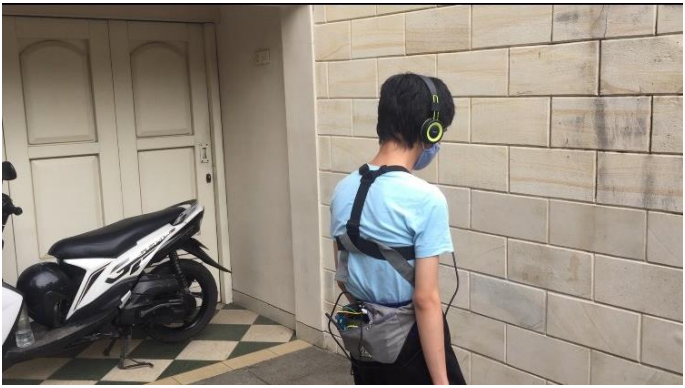
Subjek 2:

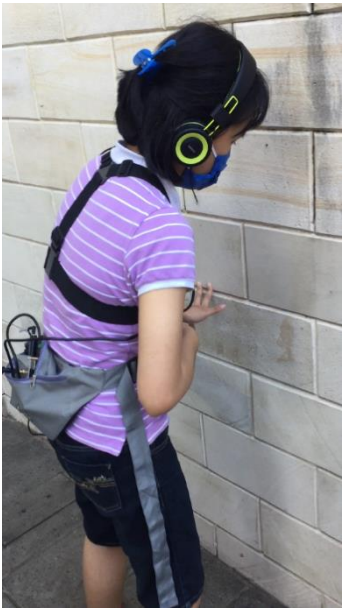
Jenis Kelamin	: Perempuan
Umur	: 18 tahun
Tingkat gangguan penglihatan	: Kebutaan Total
Penyebab gangguan penglihatan	: Congenital

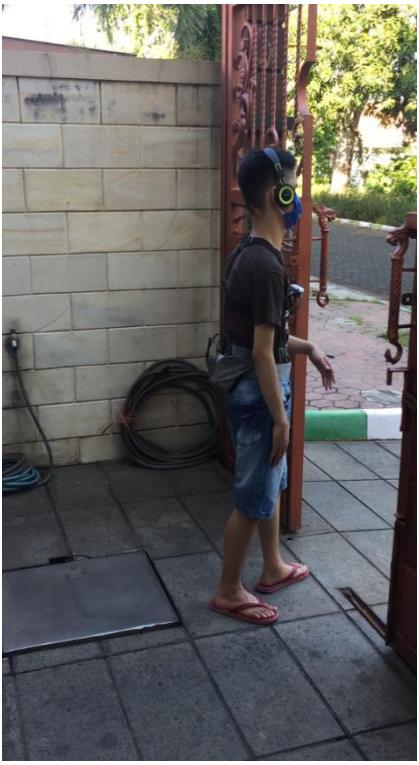
LAMPIRAN 3
DOKUMENTASI PENGUJIAN











LAMPIRAN 4
SKRIP PENGKODEAN PERANGKAT LUNAK

Source Code dapat diakses pada laman:
https://github.com/juliussin/TA/
Untuk izin akses mohon menghubungi penulis di alamat email:
julius.sintara@gmail.com

Berikut skrip utama program:

Catatan: Dibutuhkan modul tambahan yang terdapat pada *github repository* dan berbagai *packages* yang mendukung untuk dapat menjalankan skrip ini.

```
import pyrealsense2 as rs
import cv2
import pyaudio
import numpy as np
import time
# from scipy import stats
from skimage.measure import block_reduce
from multiprocessing import Lock
from scipy.io.wavfile import write
import threading
import imutils
# import ctypes
# import Jetson.GPIO
from diy_function import *
from diy_sound import *

mutex = Lock()


def convert2audio(list_of_depth, mode=1, show_time=False):
    global temp_sample
    # Error Handling
    if mode > 3:
        raise ValueError('mode should be 0~1 !!!')

    if show_time:
        time0 = time.time()
```

```
    freq_list = [tone_freq['c5'], tone_freq['d5'],
tone_freq['e5']]
    if mode == 0:
        beep_duration = [0.05, 0.08, 0.12, 0.18, 0.23]
        number_of_beep = [2, 2, 2, 2, 2]
        silence_duration = [0.07, 0.09, 0.12, 0.18, 0.20]
        number_of_pattern = [2, 2, 2, 1, 1]
        pattern_silence = [0.07, 0.09, 0.12, 0.18, 0.20]
        volume_left = [0.80, 0.45, 0.05]
        volume_right = [0.05, 0.45, 0.80]
    elif mode == 1:
        beep_duration = [0.05, 0.07, 0.1, 0.2, 0.35]
        number_of_beep = [5, 4, 3, 2, 1]
        silence_duration = [0.05, 0.05, 0.09, 0.18, 0.30]
        number of pattern = [1, 1, 1, 1, 1]
        pattern_silence = [0.1, 0.1, 0.1, 0.1, 0.1]
        volume_left = [0.80, 0.45, 0.05]
        volume_right = [0.05, 0.45, 0.80]
    else:
        beep_duration = [0.05, 0.08, 0.12, 0.18, 0.23]
        number of beep = [2, 2, 2, 2, 2]
        silence_duration = [0.07, 0.09, 0.12, 0.18, 0.20]
        number_of_pattern = [2, 2, 2, 1, 1]
        pattern_silence = [0.07, 0.09, 0.12, 0.18, 0.20]
        volume_left = [0.80, 0.45, 0.05]
        volume_right = [0.05, 0.45, 0.80]
    audio_sample = np.zeros((1, 2)).astype(np.float32)

    # Create Category Threshold
    d_distance = int((max_distance - min_distance) /
n_category)
    category_value = []
    for n in range(n_category):
        category_value.append(min_distance + d_distance * n)
    category_value.append(max_distance)

    # Iterate over list_of_depth
    for n in range(len(list_of_depth)):
        if list_of_depth[n] < category_value[0]:
            # Below min_distance
            print(0)
            audio_sample = np.vstack((audio_sample,

beep_beep(freq_list[n], number_of_beep[0], beep_duration[0],

silence_duration[0], number_of_pattern[0],

pattern_silence[0], volume_left[n], volume_right[n])))
        else:
```

```
            for m in range(1, len(category_value)):
                if category_value[m-1] <= list_of_depth[n] <
category_value[m]:
                    # Between Categories
                    print(m)
                    audio_sample = np.vstack((audio_sample,

beep_beep(freq_list[n], number_of_beep[m], beep_duration[m],

silence_duration[m], number_of_pattern[m],

pattern_silence[m], volume_left[n], volume_right[n])))
                    break
                elif list_of_depth[n] >= category_value[-1]:
                    # Above max distance
                    print('JAUH')
                    audio_sample = np.vstack((audio_sample,
np.zeros((4410, 2)).astype(np.float32)))
                    break

    audio_sample = np.vstack((audio_sample, np.zeros((4410,
2)).astype(np.float32)))
    temp_sample = audio_sample

    if show_time:
        print('convert2audio time: {0:0.3f}
ms'.format((time.time() - time0) * 1000))
    return True


def beep_beep(frequency,
              number_of_beep,
              beep_duration,
              silence_duration,
              number_of_pattern,
              pattern_silence,
              volume_left=1.0,
              volume_right=1.0,
              show_time=False):
    # Error Handling
    if volume_left > 1.0:
        raise ValueError('volume_left should be
0.0~1.0 !!!')
    if volume_right > 1.0:
        raise ValueError('volume_right should be
0.0~1.0 !!!')

    if show_time:
        time0 = time.time()
```

```
    # Fundamental Frequency
    beep = 2 * np.pi * (np.arange(fs * beep_duration) + 1) *
frequency / fs
    beep = np.sin(beep).astype(np.float32)
    # Harmonics Frequency
    beep_harmonic_2 = 2 * np.pi * (np.arange(fs *
beep_duration) + 1) * frequency * 2 / fs
    beep_harmonic_2 = 0.2 *
np.sin(beep_harmonic_2).astype(np.float32)
    beep_harmonic_4 = 2 * np.pi * (np.arange(fs *
beep_duration) + 1) * frequency * 4 / fs
    beep_harmonic_4 = 0.05 *
np.sin(beep_harmonic_4).astype(np.float32)
    # Combine + Process Beep
    beep_total = beep + beep_harmonic_2 + beep_harmonic_4
    beep_total = beep_total / max(beep_total)
    beep_total = smoothMaker(beep_total, int(beep_duration *
fs / 8))
    # Silence Sample
    silence = np.zeros(int(silence_duration *
fs)).astype(np.float32)
    # Extra Silence Sample
    extra_silence = np.zeros(int(pattern_silence *
fs)).astype(np.float32)
    # Sample Initialization
    beep_beep_sample = np.zeros(1).astype(np.float32)
    for j in range(number_of_pattern):
        for i in range(number_of_beep):
            beep_beep_sample = np.append(beep_beep_sample,
beep_total)
            beep_beep_sample = np.append(beep_beep_sample,
silence)
        beep_beep_sample = np.append(beep_beep_sample,
extra_silence)
    # Stereo Maker
    final_sample =
np.column_stack((volume_left*beep_beep_sample,

volume_right*beep_beep_sample))

    if show_time:
        print('beep_beep time: {0:0.3f}
ms'.format((time.time() - time0) * 1000))
    return final_sample


def callback(in_data, frame_count, time_info, status):  #
PyAudio Stream Callback Function
```

```
    global temp_sample, dummy_index, is_dummy, audio_save
    try:
        # print("Before Pop:  ", test.check(), frame_count)
        x = test.pop(frame_count, channel=2)
        out_data = x.tobytes()
        return out_data, pyaudio.paContinue
    except ValueError:
        # _ = test.pop(len(test.sample), channel=2)
        if temp_sample is not None:
            test.add(temp_sample)
            # audio_save = np.vstack((audio_save,
temp_sample))
            if is_dummy:
                dummy_index = dummy_index + 1
            temp sample = None
            # print(temp_sample.shape)
            x = test.pop(frame_count, channel=2)
            out_data = x.tobytes()
            return out_data, pyaudio.paContinue
        else:
            return np.zeros((frame_count, 2)).tobytes(),
pyaudio.paContinue


def realsense_callback(frames):
    global frame, frame_new, frameset, frameset_new,
accel_frame, accel_new, gyro_frame, gyro_new
    with mutex:
        if frames.get_profile().stream_type() ==
rs.stream.depth:  # and not frameset_new:
            frameset = frames
            frameset_new = True
        if frames.get_profile().stream_type() ==
rs.stream.accel:
            accel_frame = frames
            accel_new = True
        if frames.get_profile().stream_type() ==
rs.stream.gyro:
            gyro_frame = frames
            gyro_new = True
    # frame = frames
    # frame_new = True


# Audio Initiation
p = pyaudio.PyAudio()  # PyAudio Initiation
fs = 44100  # Sampling Rate 44.1khz

# Stream Setting
```

```
device_id = None  # Default: None
accel_fps = 63
gyro_fps = 200
depth_resolution = (640, 352)
depth_fps = 30
rgb_resolution = (640, 352)
rgb_fps = 30

# RealSense Initiation
pipeline = rs.pipeline()
config = rs.config()
if device_id is not None:
    config.enable_device(device_id)
config.enable_stream(rs.stream.depth, depth_resolution[0],
360, rs.format.z16, depth fps)
config.enable_stream(rs.stream.color, rgb_resolution[0],
360, rs.format.bgr8, rgb_fps)
config.enable_stream(rs.stream.accel,
rs.format.motion_xyz32f, accel_fps)
config.enable_stream(rs.stream.gyro,
rs.format.motion xyz32f, gyro fps)
profile = pipeline.start(config, realsense_callback)

# Align Depth Image to Color Image
depth_sensor = profile.get_device().first_depth_sensor()
depth_scale = depth_sensor.get_depth_scale()
align_to = rs.stream.color
align = rs.align(align_to)

# Start Sound Stream & Play Initial Tone
init_freq = [220, 246.94, 277.18, 293.66, 329.63, 369.99,
415.3,
             440]  # , 493.88, 554.37, 587.33, 659.25,
739.99, 830.61, 880]
sample, _ = phaseMaker(init_freq,  # Calculate the Phases of
the Frequency Sequence
                       total_duration=3,
                       transition_time=0.05,
                       transition_n=10)
sample = smoothMaker(sample,  # Smooth the Beginning and the
End of the Sample
                     transition_time=441 * 8)
sample_stereo = stereoMaker(sample,  # Convert Sample into
2-Channels Surround Sound
                            start_volume=0.1,
                            direction=1,
                            mode=1)
audio_save = np.copy(sample_stereo)
test = StreamOut(sample_stereo)  # Initiate StreamOut Class
```

```
stream = p.open(format=pyaudio.paFloat32,  # Open Stream
with Callback Mode
                channels=2,
                rate=fs,
                output=True,
                frames_per_buffer=2048,
                stream_callback=callback)
stream.start_stream()  # Start Stream

# Global Variables
frame = None
frame_new = False
frameset = None
frameset_new = False
accel_frame = None
accel_new = False
gyro_frame = None
gyro_new = False
theta = np.array([0, 0, 0]).astype(np.float32)
alpha = 0.98
counter = 0
counter_max = 6  # frames
cycle_duration = 0.7  # second
last_ts_accel = None
last_ts_gyro = None
pooling_size = (16, 16)  # Resize to 40 x 22.5 (40 x 22)
middle_crop = (10, 14)
bottom_crop = (14, 18)
warning_counter = 0
warning_memory = np.array([False, False])
temp_sample = None
rotate_angle = 0
# audio_save = np.zeros((1, 2)).astype(np.float32)
data_dummy = [[1800, 1800, 1800],
              [1500, 1800, 1800],
              [1000, 1800, 1800],
              [500, 1800, 1800],
              [300, 1800, 1800],
              [1800, 1000, 1800],
              [1800, 300, 1800],
              [1800, 500, 1800],
              [1800, 1800, 500],
              [1800, 1800, 1000],
              [1800, 1800, 300]]
dummy_index = 0
dummy_n = 3
is_dummy = True
# temp_sample = Array(ctypes.c_float, )
```

```
# Distance Parameter
min_distance = 400  # millimeter
max_distance = 1600  # millimeter
n_category = 3  # step = (max_distance - min_distance) /
category

orig = ImageShow('ORIG')
inpaint2 = ImageShow('INPAINT2ND')

time.sleep(0.5)  # Delay

try:
    while True:
        time1 = time.time()
        if frameset new:
            time_frameset = time.time()
            # Align Depth Image to Color Image
            composite_frame = rs.composite_frame(frameset)
            aligned_frames = align.process(composite_frame)
            aligned_depth_frame =
aligned frames.get depth frame()
            aligned_color_frame =
aligned_frames.get_color_frame()

            # Convert into Numpy
            depth_image =
np.asanyarray(aligned_depth_frame.get_data())
            color_image =
np.asanyarray(aligned_color_frame.get_data())

            # Crop
            depth_image = depth_image[8:, :]
            color_image = color_image[8:, :]

            # Morphological Transformation
            kernel = np.ones((9, 9))
            img_dilation = cv2.dilate(depth_image, kernel,
iterations=1)
            img_erosion = cv2.erode(img_dilation, kernel,
iterations=2)
            img_dilation2 = cv2.dilate(img_erosion, kernel,
iterations=2)

            # First Inpaint (Small Contours on Original
Depth Image)
            is_mask = (img_dilation2 == 0)
            mask =
np.zeros(img_dilation2.shape).astype(np.uint8)
            mask[is_mask] = 255
```

```
            contours, hierarchy = cv2.findContours(mask,
cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_NONE)
            area = 0
            for cnt in contours:  # Do inpaint only for
small contours
                if cv2.contourArea(cnt) > 1000:
                    area = area + cv2.contourArea(cnt)
                    cv2.drawContours(mask, [cnt], -1, 0,
cv2.FILLED)
                    if area > 10000:  # If total contours
area exceed certain value, stop.
                        break
            img_inpaint =
cv2.inpaint(img_dilation2.astype(np.uint16), mask, 5, 0)

            # Resize with MinPool
            img_minpool = block_reduce(img_inpaint,
pooling_size, np.nanmin)
            img_minpool = img_minpool[:int(rgb_resolution[1]
/ pooling_size[0]),
                          :int(rgb_resolution[0] /
pooling_size[1])]
            img_minpool_resize = cv2.resize(img_minpool,
rgb_resolution,  # Optional

interpolation=cv2.INTER_NEAREST)

            # Raise Error for Abnormal Image
            abnormal_percentage = 0.3
            total_abnormal = np.count_nonzero(img_minpool ==
0)
            if total_abnormal > abnormal_percentage *
img_minpool.shape[0] * img_minpool.shape[1]:
                print('Abnormal!')
                warning = True
            else:
                warning = False

            # Second Inpaint (Big Contours after Resized)
            is_mask2 = (img_minpool == 0)
            mask2 =
np.zeros(img_minpool.shape).astype(np.uint8)
            mask2[is_mask2] = 255
            img_inpaint2 =
cv2.inpaint(img_minpool.astype(np.uint16), mask2, 1, 0)

            # Outlier Detection
            # for row in range(img_inpaint2.shape[0]):
            #     d0 = None
```

```
            #     for col in range(img_inpaint2.shape[1] -
1):
            #         if col == 0: continue
            #         d1 = img_inpaint2[row, col-1] -
img_inpaint2[row, col]
            #         d2 = img_inpaint2[row, col] -
img_inpaint2[row, col+1]
            #         if d1 > 500 and d2 < -500:  # Large
differential (1 element, very low value)
            #             img_inpaint2[row, col] =
img_inpaint2[row, col+1]
            #         if d0 is not None and d0 > 500 and d2
< -500:  # Large differential (2 elements, very low value)
            #             img_inpaint2[row, col] =
img inpaint2[row, col+1]
            #             img_inpaint2[row, col-1] =
img_inpaint2[row, col-2]
            #         d0 = d1

            img_inpaint2_resize = cv2.resize(img_inpaint2,
rgb_resolution,  # Optional

interpolation=cv2.INTER_NEAREST)

            # Too Close Warning
            front_crop = img_inpaint2[:,
int(img_inpaint2.shape[1] / 2) - 6:int(img_inpaint2.shape[1]
/ 2) + 6]
            # less_than_min = (front_crop <
minimum_distance)
            # if less_than_min.any():
            #     print('Warning! Too Close')
            #     warning = True
            # else:
            #     warning = False
            # OR
            less_than_min_threshold = 5
            less_than_min = front_crop[front_crop <
min_distance - 100]
            if len(less_than_min) > less_than_min_threshold:
                print('Warning! Too Close > 5')
                warning = True
            else:
                warning = False

            # Crop Middle and Bottom
            # if not warning:
            processing_time = time.time()
```

```
            middle =
img_inpaint2[middle_crop[0]:middle_crop[1], :]  # Crop
Middle
            is_error = np.less(middle, np.ones(middle.shape)
* 50)  # Check for Value < 50
            if middle.dtype != np.float:
                middle = middle.astype(np.float)
            middle[is_error] = np.nan  # Set Value < 50 to
NaN
            middle = block_reduce(middle, (4, 2), np.nanmin)
# Min Pooling (Ignore NaN)
            # print('Middle Size: {}'.format(middle.shape))
            left_val = int(np.nanmin(middle[:, 3:7]))
            middle_val = int(np.nanmin(middle[:, 8:12]))
            right_val = int(np.nanmin(middle[:, 13:17]))
            print('Left, Middle, Right: {}, {},
{}'.format(left_val, middle_val, right_val))

            # Convert to Audio
            info = [left_val, middle_val, right_val]
            if is_dummy:
                try:
                    info =
data_dummy[int(dummy_index/dummy_n)]
                except IndexError:
                    info = [1800, 1800, 1800]
            convert2audio(info)

            # Warning Counter
            if warning:
                warning_counter = warning_counter + 1
            elif not warning_counter and
warning_memory.any():
                warning_counter = warning_counter + 1
            else:
                warning_counter = 0
            warning_memory = np.insert(warning_memory, 0,
warning)
            warning_memory = warning_memory[:-1]

            # Warning Handing
            if warning_counter > 4:
                # Send Audio Warning
                print('MEGA WARNING!')
                # Make Tiiiiiittt

            # ColorMap for Visualization
```

```
            img_colormap =
cv2.applyColorMap(cv2.convertScaleAbs(depth_image,
alpha=0.03), cv2.COLORMAP_JET)
            # img_colormap2 =
cv2.applyColorMap(cv2.convertScaleAbs(img_dilation2,
alpha=0.03), cv2.COLORMAP_JET)
            # img_colormap3 =
cv2.applyColorMap(cv2.convertScaleAbs(img_inpaint,
alpha=0.03), cv2.COLORMAP_JET)
            # img_colormap4 =
cv2.applyColorMap(cv2.convertScaleAbs(img_minpool_resize,
alpha=0.03), cv2.COLORMAP_JET)
            img_colormap5 =
cv2.applyColorMap(cv2.convertScaleAbs(img_inpaint2_resize,
alpha=0.03),

cv2.COLORMAP_JET)
            # img_colormap6 =
cv2.applyColorMap(cv2.convertScaleAbs(, alpha=0.03),
cv2.COLORMAP_JET)

            # Draw Rectangle Middle and Bottom
            color_image = cv2.rectangle(color_image,
                                        (0 * 16,
middle_crop[0] * 16),
                                        (8 * 16,
middle_crop[1] * 16),
                                        (0, 255, 0),
                                        1)
            color_image = cv2.rectangle(color_image,
                                        (8 * 16,
middle_crop[0] * 16),
                                        (16 * 16,
middle_crop[1] * 16),
                                        (0, 255, 0),
                                        1)
            color_image = cv2.rectangle(color_image,
                                        (16 * 16,
middle_crop[0] * 16),
                                        (24 * 16,
middle_crop[1] * 16),
                                        (0, 255, 0),
                                        1)
            color_image = cv2.rectangle(color_image,
                                        (24 * 16,
middle_crop[0] * 16),
                                        (32 * 16,
middle_crop[1] * 16),
                                        (0, 255, 0),
```

```
                                        1)
            color_image = cv2.rectangle(color_image,
                                        (32 * 16,
middle_crop[0] * 16),
                                        (40 * 16,
middle_crop[1] * 16),
                                        (0, 255, 0),
                                        1)
            # color_image = cv2.rectangle(color_image,
            #                             (5 * 16,
bottom_crop[0] * 16),
            #                             (35 * 16,
bottom_crop[1] * 16),
            #                             (0, 255, 0),
            #                             1)
            # img_colormap5 = cv2.rectangle(img_colormap5,
            #                               (5 * 16,
middle_crop[0] * 16),
            #                               (35 * 16,
middle_crop[1] * 16),
            #                               (0, 255, 0),
            #                               1)
            # img_colormap5 = cv2.rectangle(img_colormap5,
            #                               (5 * 16,
bottom_crop[0] * 16),
            #                               (35 * 16,
bottom_crop[1] * 16),
            #                               (0, 255, 0),
            #                               1)

            # Rotate for show
            # color_image = imutils.rotate(color_image,
rotate_angle)

            # Show the Images
            # cv2.imshow("ORIGINAL", img_colormap)
            # cv2.imshow("MORPH", img_colormap2)
            # cv2.imshow("INPAINT", img_colormap3)
            # cv2.imshow("MINPOOL", img_colormap4)
            # cv2.imshow("INPAINT2", img_colormap5)
            cv2.imshow("COLOR", color_image)
            # minpool = ImageShow('MINPOOL', img_colormap4,
img_minpool_resize)
            # inpaint2 = ImageShow('INPAINT2ND',
img_colormap5, img_inpaint2_resize)
            # orig = ImageShow('ORIG', img_colormap,
depth_image)
            orig.setValue(depth_image)
            orig.show(img_colormap)
```

```
            inpaint2.setValue(img_inpaint2_resize)
            inpaint2.show(img_colormap5)

            # print('Time: {0:0.3f}ms & FPS:
{1:0.2f}'.format(
            #     (time.time() - time_frameset) * 1000,
            #     1 / (time.time() - time_frameset)))

            key = cv2.waitKey(1)

            if key == 27:
                break
            elif key == ord('s'):
                while cv2.waitKey(1) != ord('r'):
                    None
            elif key == ord('c'):
                save_image(color_image, 'satu',
os.path.join('test_alat_1', 'color'), uid='datetime')
                save_image(img_colormap, 'satu',
os.path.join('test_alat_1', 'colormap'), uid='datetime')
                save csv(depth image, 'satu',
os.path.join('test_alat_1', 'depth'), uid='datetime')
            frameset_new = False

        if accel_new:
            ts_gyro = gyro_frame.get_timestamp()
            accel_data =
accel_frame.as_motion_frame().get_motion_data()
            # print('Accel X, Y, Z:
{}\t{}\t{}'.format(accel_data.x, accel_data.y, accel_data))
            gyro_data =
gyro_frame.as_motion_frame().get_motion_data()
            # print('Gyro  X, Y, Z:
{}\t{}\t{}'.format(gyro_data.x, gyro_data.y, gyro_data))
            # Process here to Theta
            resultant = np.math.sqrt(accel_data.y *
accel_data.y
                                     + accel_data.z *
accel_data.z)
            accel_roll = np.math.atan2(accel_data.x,
resultant)
            accel_pitch = np.math.atan2(accel_data.y,
accel_data.z)
            # accel_roll = np.math.acos(accel_value[0] /
resultant)
            # accel_roll = np.math.acos(accel_value[0] /
resultant)
            # accel_yaw = np.math.acos(accel_value[1] /
resultant)
```

```
            # accel_pitch = np.math.acos(accel_value[2] /
resultant)
            if last_ts_gyro is not None:
                dt = (ts_gyro - last_ts_gyro) / 1000.0
                theta[0] = (theta[0] + gyro_data.x * dt) *
alpha + accel_pitch * (1 - alpha)
                theta[1] = (theta[1] - gyro_data.y * dt)  #
* alpha + accel_yaw * (1-alpha)
                theta[2] = (theta[2] - gyro_data.z * dt) *
alpha + accel_roll * (1 - alpha)
            else:
                theta[0] = accel_pitch
                theta[1] = np.math.pi  # accel_yaw
                theta[2] = accel_roll
            last ts gyro = ts gyro
            # print('Pitch: {}  Yaw: {}  Roll:
{}'.format(theta[0]*180/np.math.pi,
            #
theta[1]*180/np.math.pi,
            #
theta[2]*180/np.math.pi))
            rotate_angle = theta[2]*180/np.math.pi + 1

    # audio_save = np.vstack((audio_save, np.zeros((44100,
2))))
    # write('save_sound.wav', 44100, audio_save)

finally:
    # Stop Realsense Pipeline
    pipeline.stop()
    # Stop Sound Stream
    stream.stop_stream()
    stream.close()
    # Close PyAudio
    p.terminate()
    # Close OpenCV Windows
    cv2.destroyAllWindows()

# 554 lines
```