TESIS – EE185401

RANCANG BANGUN SISTEM PEMETAAN HALANGAN PADA RUANG SEBAGAI ALAT BANTU NAVIGASI TUNANETRA

ICHSAN PRATAMA ADI
NRP 07111750040004

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

PROGRAM MAGISTER
BIDANG KEAHLIAN TEKNIK ELEKTRONIKA
DEPARTEMEN TEKNIK ELEKTRO
FAKULTAS TEKNOLOGI ELEKTRO
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA

TESIS – EE185401

RANCANG BANGUN SISTEM PEMETAAN HALANGAN PADA RUANG SEBAGAI ALAT BANTU NAVIGASI TUNANETRA

ICHSAN PRATAMA ADI
NRP 07111750040004

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

PROGRAM MAGISTER
BIDANG KEAHLIAN TEKNIK ELEKTRONIKA
DEPARTEMEN TEKNIK ELEKTRO
FAKULTAS TEKNOLOGI ELEKTRO
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA
2019

THESIS – EE185401

DESIGN AND DEVELOPMENT OF OBSTACLE MAPPING SYSTEM IN A ROOM AS A NAVIGATION TOOL FOR BLIND

ICHSAN PRATAMA ADI
NRP 07111750040004

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

MASTER PROGRAM
FIELD OF ELECTRONICS ENGINEERING
DEPARTMENT OF ELECTRICAL ENGINEERING
ELECTRICAL TECHNOLOGY FACULTY
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA
2019

LEMBAR PENGESAHAN TESIS

Tesis disusun untuk memenuhi salah satu syarat memperoleh gelar
Magister Teknik (MT)
di
Institut Teknologi Sepuluh Nopember
Oleh:

ICHSAN PRATAMA ADI
NRP: 07111750040004

Tanggal Ujian: 02 Juli 2019
Periode Wisuda: September 2019

Disetujui oleh:
Pembimbing:

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

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

Penguji:

1. Dr. Ir. Djoko Purwanto, M.Eng. .....
   NIP: 196512111990021002

2. Dr. Ir. Totok Mujiono, M.IKom. .....
   NIP: 196504221989031001

3. Dr. Achmad Arifin, ST., M.Eng. .....
   NIP: 197103141997021001

4. Dr. Tri Arief Sardjono, S.T., M.T. .....
   NIP: 197002121995121001

Kepala Departemen Teknik Elektro
Fakultas Teknologi Elektro

Dr.Eng. Ardyono Priyadi, ST., M.Eng.
NIP: 197309271998031004

Halaman ini sengaja dikosongkan

PERNYATAAN KEASLIAN TESIS

Dengan ini saya menyatakan bahwa isi keseluruhan Tesis saya dengan judul "**RANCANG BANGUN SISTEM PEMETAAN HALANGAN PADA RUANG SEBAGAI ALAT BANTU NAVIGASI TUNANETRA**" adalah benar-benar hasil karya intelektual mandiri, diselesaikan tanpa menggunakan bahan-bahan yang tidak diizinkan 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, Maret 2019

Ichsan Pratama Adi

NRP. 07111750040004

Halaman ini sengaja dikosongkan

RANCANG BANGUN SISTEM PEMETAAN HALANGAN PADA RUANG SEBAGAI ALAT BANTU NAVIGASI TUNANETRA

Nama mahasiswa : Ichsan Pratama Adi
NRP : 07111750040004
Pembimbing : 1. Dr. Ir. Hendra Kusuma, M.Eng.,Sc.
2. Muhammad Attamimi, B.Eng., M.Eng., Ph.D.

ABSTRAK

Seorang tunanetra membutuhkan sebuah alat untuk navigasi ketika berjalan di tempat yang asing atau di tempat yang tidak familier. Biasanya, tongkat adalah salah satu solusi yang mampu memandu mereka memetakan jalan yang akan dilaluinya. Perjalanan mereka cukup lambat dikarenakan mereka harus memetakan dengan tongkat terlebih dahulu, dan jangkauan pemetaannya hanya sebatas panjang tongkat yang digunakan. Masalah lainnya adalah ketika ada objek yang tidak menyentuh tanah seperti pintu atau pagar yang tidak dapat dideteksi dengan tongkat biasa. Untuk mengatasi keterbatasan itu, pada penelitian ini kami merancang sistem pembantu berbasis stereo kamera yang dapat mendeteksi objek dan atau penghalang di sekitar pengguna tunanetra. Dengan stereo kamera, jarak dari objek di sekitar sistem dapat dideteksi dan dapat diinformasikan kepada pengguna tunanetra melalui suara stereo. Dengan pendekatan ini, kami menganggap bahwa para tunanetra dapat berjalan lebih cepat. Dari hasil eksperimen, tunanetra yang menggunakan sistem ini dapat berjalan menghindari 83.16% dari halangan yang diletakkan di depannya, dan dapat memandu tunanetra untuk mencari jalan yang kosong dengan penuh percaya diri.

Kata kunci: Navigasi, stereo kamera, suara stereo, tunanetra.

Halaman ini sengaja dikosongkan

DESIGN AND DEVELOPMENT OF OBSTACLE MAPPING SYSTEM IN A ROOM AS A NAVIGATION TOOL FOR BLIND

By : Ichsan Pratama Adi
Student Identity Number : 07111750040004
Supervisors : 1. Dr. Ir. Hendra Kusuma, M.Eng.Sc
2. Muhammad Attamimi, B.Eng., M.Eng., Ph.D

ABSTRACT

Blind people need a tool for navigation when walking in an unfamiliar place. Usually, a stick is one of solution to guide them map the walking path to be passed. Their movement is slow because they have to map with stick, which the radius of the map is only along to the stick. Another problem is if there are objects that do not touch the ground, such as railroad crossing gate cannot be detected with simple stick. To that limitation, in this research we propose a stereo camera based assistive-system that can detect objects and or obstacles around the blind people. With Stereo Camera, the distance of objects around the system can be detected and will be informed to the blind-user via stereo sound. By this approach, we expected that the blind people are able to walk faster. From experimental results, blind people that use this system can avoid 83.16% of collisions placed in front of him, and able to guide blind people to find the empty path confidently.

Key words: Blind, navigation, stereo camera, stereo sound.

Halaman ini sengaja dikosongkan

KATA PENGANTAR

Puji syukur kepada Allah Subhanahuwata'ala, yang selalu memberikan hidayah-Nya sehingga tesis ini dapat diselesaikan. Tesis berjudul "RANCANG BANGUN SISTEM PEMETAAN HALANGAN PADA RUANG SEBAGAI ALAT BANTU NAVIGASI TUNANETRA" ini disusun untuk memenuhi sebagian persyaratan memperoleh gelar Magister Teknik (M.T.) pada jurusan Teknik Elektro, Fakultas Teknologi Elektro, Institut Teknologi Sepuluh Nopember.

Penulis menyadari bahwa dalam penyusunan tesis ini tidak terlepas dari bantuan berbagai pihak. Oleh karena itu, dengan segala ketulusan dan kerendahan hati penulis menyampaikan terima kasih kepada:

1. Bapak, ibu, dan seluruh keluarga yang telah memberikan dukungan serta motivasi untuk penulis dalam menempuh studi pascasarjana.
2. Bapak Dr. Ir. Hendra Kusuma, M.Eng.Sc. selaku dosen pembimbing I yang telah sabar, banyak memberikan saran, bantuan, bimbingan, arahan, serta motivasi kepada penulis untuk menyelesaikan penelitian.
3. Bapak Muhammad Attamimi, B.Eng., M.Eng., Ph.D. selaku dosen pembimbing II yang telah banyak memberi masukan, saran, arahan, dukungan, serta semangat kepada penulis untuk menyelesaikan penelitian.
4. Seluruh dosen Teknik Elektro yang telah memberikan ilmunya dalam kuliah yang mendukung terselesaikannya penelitian ini.
5. Rekan-rekan pascasarjana jurusan Teknik Elektro yang telah banyak membantu dan mendukung dalam menyelesaikan penelitian ini.

Penulis sadar bahwa penelitian ini masih belum sempurna dan masih banyak hal yang perlu diperbaiki.

Surabaya, 2 Juli 2019

Ichsan Pratama Adi

Halaman ini sengaja dikosongkan

DAFTAR ISI

DAFTAR GAMBAR

Halaman ini sengaja dikosongkan

DAFTAR TABEL

Halaman ini sengaja dikosongkan

BAB 1
PENDAHULUAN

1.1 Latar Belakang

Pada tahun 2017, diperkirakan dari 253 juta orang yang hidup dengan gangguan penglihatan di seluruh dunia, 36 juta orang mengalami kebutaan [1]. Artinya 4,8% dari penduduk dunia yang berjumlah 7,5 milyar [2] adalah tunanetra. Kebutaan adalah salah satu permasalahan yang rumit pada kategori disabilitas manusia. Penglihatan merupakan indra yang sangat berperan penting dalam mengenali dan merasakan lingkungan sekitarnya. Kurangnya atau tidak adanya penglihatan dari seseorang, maka orang tersebut tidak dapat mengenali dan merasakan lingkungan sekitarnya dengan baik. Dengan mata juga manusia dapat melihat adanya halangan di sekitarnya untuk memandu jalan. Orang yang memiliki kekurangan dalam indra penglihatan akan sangat kesulitan dalam melakukan aktivitas sehari-hari, seperti berkendara, membaca, menjahit, dan berjalan [3]. Dalam hal berjalan, mereka akan kesulitan berjalan tanpa bantuan alat khusus. Sehingga diperlukan sebuah alat khusus yang mampu membantu tunanetra mengenali dan menavigasikan lingkungannya dengan baik.

Perkembangan teknologi yang semakin maju ini menciptakan harapan kepada orang-orang yang memiliki kebutuhan khusus. Untuk tunanetra sendiri sudah banyak alat bantu yang menggunakan teknologi terkini. *Assistive Robot* [4] dapat membantu tunanetra untuk menghindari halangan, lubang, dan tangga. *Smart Walking Stick* [5] memandu jalan dengan memberikan sedikit penglihatan artifisial. *GuideCane*, robot dengan pegangan yang memandu jalan dengan sonar [6]. Tetapi alat-alat tersebut memiliki kekurangan yang hanya melakukan tugas secara khusus saja, dan sudah tertinggal zaman. Beberapa di antaranya juga masih menggunakan sensor yang kurang efektif, misalnya sensor ultrasonik yang hanya mendeteksi objek dalam 1 dimensi.

Pada penelitian ini, kami mengimplementasikan kamera stereo untuk mendeteksi jarak suatu objek di sekitarnya. Kamera stereo memiliki kelebihan mendeteksi jarak pada bidang matriks. Ukuran matriks jarak dalam sekali ambil

sama besarnya dengan resolusi gambar. Tujuannya adalah untuk mengembangkan riset tentang perangkat bantu untuk tuna netra yang menginformasikan lokasi objek terdekat ke penggunanya melalui suara.

1.2 Rumusan Masalah

Berdasarkan latar belakang yang telah dipaparkan, maka dapat dibuat rumusan permasalahan sebagai berikut:

1. Bagaimana citra objek yang ditangkap oleh stereo kamera dapat digunakan untuk mengetahui jarak objek terhadap kamera.
2. Bagaimana cara memberikan informasi ke tunanetra berdasarkan dari gambar stereo yang diambil.
3. Apakah informasi suara yang disampaikan ke pengguna tunanetra dapat dipahami.

1.3 Tujuan

Tujuan dari dilakukannya penelitian ini berdasarkan dari perumusan masalah di atas adalah:

1. Mengetahui jarak suatu objek menggunakan kamera stereo dan mengubah menjadi *depth image*.
2. Menginformasikan pengguna tunanetra untuk menghindari halangan di depannya.
3. Pengguna tunanetra dapat memahami instruksi navigasi dari sistem dan dapat mengikuti instruksinya.

1.4 Batasan Masalah

Dalam perancangan penelitian ini, ada beberapa hal yang dibatasi oleh penulis. Di antaranya adalah:

1. Kamera yang digunakan adalah kamera stereo ZED.
2. Objek yang dideteksi merupakan dinding ruangan, dan perabotan berukuran besar dengan jarak 0.5 – 8 meter.
3. Digunakan oleh tunanetra.

1.5 Kontribusi

Kontribusi hasil dari penelitian ini adalah menerapkan *depth image* yang diperoleh dari kamera stereo, dimana informasi *depth* akan dikonversikan menjadi nada stereo yang sebanding dengan informasi *depth*. Suara nada stereo ini yang akan menjadi panduan penyandang tunanetra dalam bernavigasi dengan memetakan halangan di sekitarnya. Hal ini akan membuat mereka lebih cepat dan leluasa dalam berjalan.

Halaman ini sengaja dikosongkan

BAB 2
KAJIAN PUSTAKA

Tesis ini disusun berdasarkan beberapa penelitian yang sudah dilakukan oleh orang lain, yang diperbaiki dan digabungkan sehingga menghasilkan suatu terobosan baru. Beberapa referensi terkait penelitian yang sudah dilakukan dan teori pendukung dalam pembuatan tesis ini disebutkan di dalam bab ini.

2.1 Kajian Penelitian Terkait

Penelitian ini dibuat berdasarkan referensi beberapa penelitian yang sudah dilakukan sebelumnya. Penelitian dengan bidang yang berbeda-beda kemudian digabungkan sehingga dapat membantu menyelesaikan permasalahan pada penelitian ini. Berbagai bidang penelitian terkait penelitian ini yang sudah dilakukan disebutkan pada poin-poin selanjutnya.

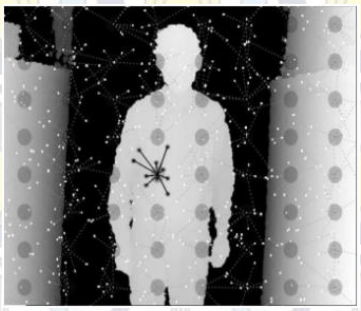
2.1.1 Penelitian Terkait Alat Pemandu Tunanetra

Ada beberapa penelitian yang sudah dilakukan yang menciptakan alat untuk membantu tunanetra berjalan. Salah satu di antaranya adalah *GuideCane*, yang memberikan bantuan navigasi kepada tunanetra berbentuk robot yang berjalan memandu melalui jalan-jalan yang aman untuk dilewati [6]. Alat ini berbentuk robot 2 roda yang menggunakan beberapa sensor ultrasonik untuk mendeteksi jarak halangan di sekitarnya. Robot ini tidak memiliki daya untuk memutar roda yang berjalan sehingga harus didorong penggunanya. Ketika robot ini berjalan, ia akan mendeteksi halangan di sekitarnya, sehingga dapat mendapatkan jalur mana yang bisa dilalui oleh robot dan penggunanya. Beberapa sensor ultrasonik yang digunakan agar robot dapat mendeteksi halangan 120° di sekitarnya. Data yang diambil dari ultrasonik kemudian diproses menggunakan teknik yang dinamakan *Vector Field Histogram* (VFH) yang dikombinasikan dengan metode *Error Eliminating Rapid Ultrasonic Firing* (EERUF) untuk menembakkan sonar. Metode ini diklaim dapat digunakan jalan yang cepat [6]. Keluaran dari metode ini digunakan untuk mengontrol motor servo yang berfungsi mengarahkan jalannya ke arah yang aman. Sehingga ketika pengguna mendorong robot ini, robot otomatis membelokkan penggunanya menuju ke arah yang aman. Pengguna dari *GuideCane*

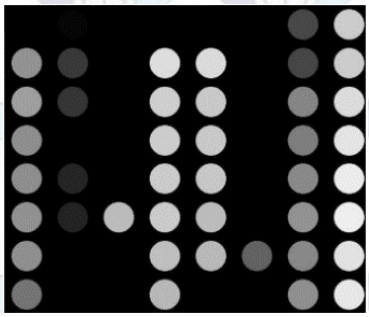
ini merasakan perintah kemudi yang sangat terasa tenaga fisiknya pada pegangan robot, dan dapat dengan mudah mengikuti jalur dari *GuideCane* tanpa kesulitan yang berarti.

MeloSee merupakan salah satu alat portabel yang berfungsi untuk menavigasikan seseorang melalui gambar kedalaman (*Depth Image*) ke dalam bentuk suara. *MeloSee* menggunakan sensor tipe Kinect untuk mengambil gambar kedalaman 2-dimensi, kemudian gambar tersebut dikonversikan ke dalam bentuk melodi secara *real-time*. Gambar yang ditangkap oleh sensor memiliki resolusi VGA kemudian dibagi menjadi 64 titik *Receptive Field* (RF) menjadi ukuran 8x8. Masing-masing titik tersebut mewakili kedalaman kamera dari beberapa piksel di sekitarnya. 64 titik RF tersebut kemudian direpresentasikan ke dalam bentuk intensitas suara, modulasi stereo yang merepresentasikan objek secara lateral, dan modulasi frekuensi nada yang merepresentasikan objek secara vertikal [7]. Berikut konversi gambar yang dilakukan, ditampilkan pada Gambar 2.1. Konversi suara dari masing-masing titik memiliki frekuensi dan amplitudo yang berbeda-beda.

a) Gambar kedalaman



b) Aktivitas penghitungan

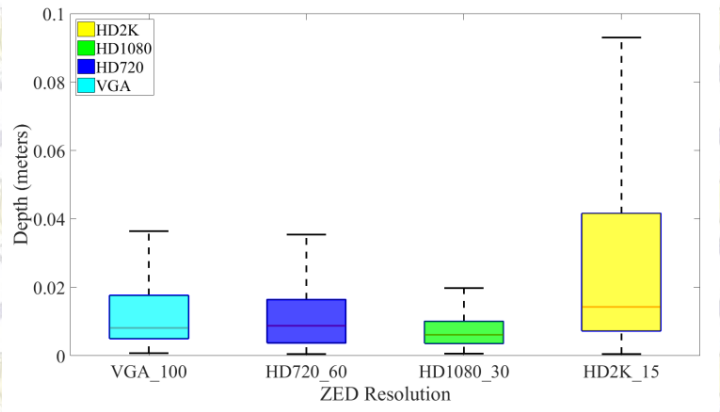


c) Aktivitas RF: semakin dekat objeknya, semakin terang lingkarannya.

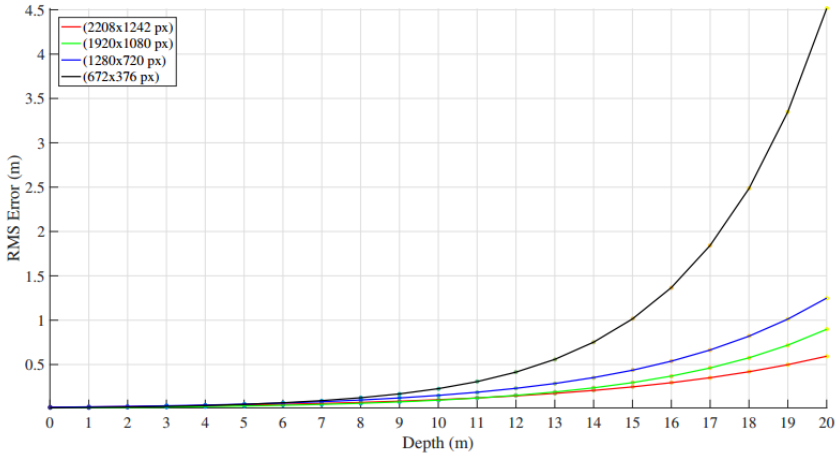
Gambar 2.1 Pengolahan enkode gambar kedalaman dari alat *Melosee* [7]

2.1.2 Penelitian Terkait Stereo Kamera

Penelitian terkait kamera stereo masih cukup jarang dilakukan. Terutama kamera stereo yang sudah dapat dipakai langsung tanpa menyusun dua buah kamera secara manual, salah satunya adalah kamera ZED. Kamera ZED ini sendiri baru pertama kali dijual di pasaran pada tahun 2015 [8]. Sehingga belum banyak riset yang sudah dilakukan terkait kamera ini. Beberapa riset yang sudah dilakukan adalah memodelkan *error* dari data kedalaman (*depth data*) dari kamera ZED yang dilakukan oleh Luiz Ortiz [9]. Kamera ini memiliki 4 jenis resolusi yang digunakan, yaitu WVGA, HD720, HD1080, dan HD2K. Masing-masing resolusi tersebut diukur standar deviasi jarak yang diperoleh ketika kamera tersebut diletakkan di tanah yang datar dengan jarak 4 meter dari tembok yang terletak di depannya. Hasil



(a)

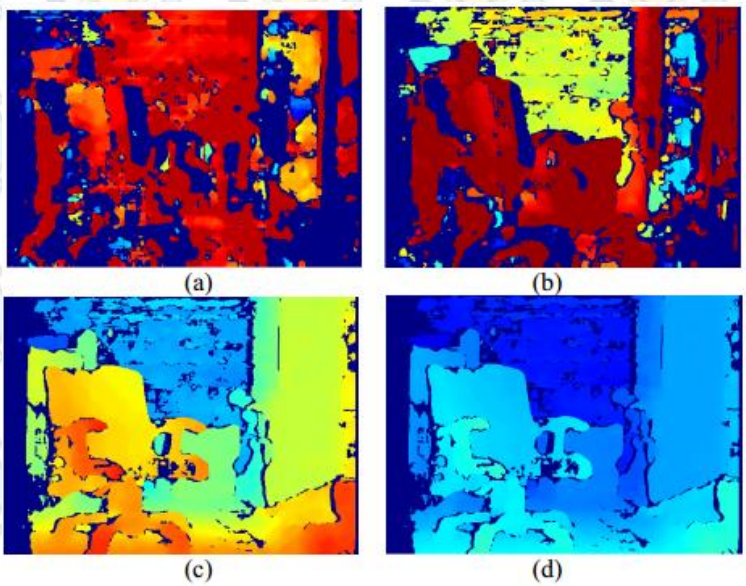


(b)

Gambar 2.2 Grafik analisis error gambar kedalaman kamera ZED. (a) Grafik standar deviasi, dan (b) Grafik RMS error tiap jarak

standar deviasi tiap resolusi bermacam-macam dari 0 hingga 0.9 meter, dengan gambar penyebaran yang ditampilkan pada Gambar 2.2 (a). Selain standar deviasi, juga diambil error RMS pada tiap-tiap resolusi dan tiap jarak. Dengan data ini, Luis [9] mengklaim bahwa kamera ZED dapat digunakan dengan presisi yang baik hingga 16 meter. Kemudian error yang diperoleh dari resolusi yang semakin besar, maka errornya semakin kecil, akan tetapi proses kalkulasinya juga semakin besar. Sehingga keseimbangan error dan waktu kalkulasi diperlukan bergantung pada aplikasinya.

Beberapa penelitian terkait konstruksi 3-dimensi dari stereo kamera sudah banyak dilakukan. Algoritme *Semi-Global Matching* (SGM) adalah salah satu cara dalam menentukan jarak menggunakan stereo kamera. Caranya adalah dengan menentukan persamaan dari kedua buah gambar, kemudian menentukan titik-titik piksel tetangganya (*neighbor pixel*) sebagai bantuan untuk meningkatkan akurasi jaraknya. Untuk menentukan jarak objek setiap piksel terhadap kamera, dibutuhkan kalibrasi gambar dari kedua kamera kiri dan kanan. Gambar tersebut disejajarkan hingga tinggi dari gambar kiri dan kanan benar-benar tepat. Kemudian gambar tersebut ditransformasi berdasaran pergeseran perbedaan antara kamera kanan dan kiri. Gambar yang sudah ditransformasi tersebut yang diolah menggunakan algoritme SGM lalu diukur *disparity map*-nya untuk menentukan jaraknya. Dalam



Gambar 2.3 *Disparity map* dengan bermacam-macam neighbor pixel. (a) 0-16 pixel, (b) 0-32 pixel, (c) 0-64 pixel, (d) 0-128 pixel

proses *disparity mapping*, jarak objek ke kamera ditentukan berdasarkan parameter dari kamera yang mengandung parameter intrinsik, dan parameter ekstrinsik. Isi dari parameter intrinsik mencakup *focal length*, titik tengah optik, *skew* masing-masing kamera. Kemudian parameter intrinsik berisi nilai perbedaan kamera kanan dan kiri, salah satunya adalah jarak kedua kamera [10].

Riset lain terkait kamera ZED adalah akuisisi data 3-dimensi (3D) dan pemetaan dalam ruangan yang dilakukan untuk *Smart Cities*. Pada pemetaan 3D, kamera ini digunakan untuk merekonstruksi sebuah objek dari bentuk gambar berwarna dan *depth image*-nya ke bentuk 3D [11]. Kamera ini juga dapat digunakan untuk pemetaan 3D dengan menjalankannya untuk mengelilingi suatu tempat, sehingga didapatkan nilai warna dan kedalaman dari objek di depannya. Teknologi ini berpotensi untuk digunakan pada kebutuhan komersial, terutama untuk melacak kemajuan dari konstruksi bangunan, dalam hal keamanan, manajemen fasilitas, retail dan aplikasi *Augmented Reality* [12].

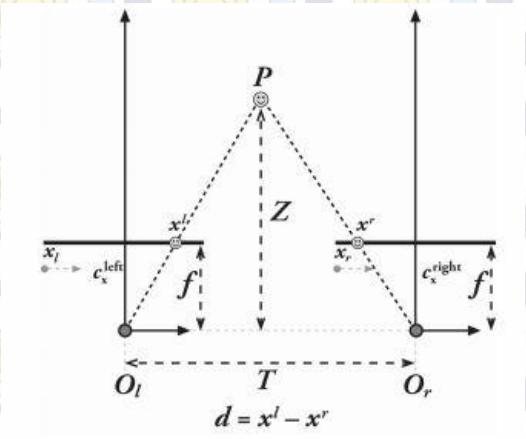
2.2 Teori Dasar

Beberapa teori dasar yang mendukung dalam pelaksanaan penelitian ini ada 10. Teori dasar yang digunakan secara detail dijabarkan pada poin-poin selanjutnya.

2.2.1 Kamera Stereo

Stereo kamera adalah salah satu jenis kamera yang menggunakan dua buah lensa dengan sensor gambar yang terpisah [13]. Kedua buah kamera ini terpisah oleh jarak tertentu (*intra-ocular distance*) sehingga membentuk efek triangulasi yang dapat mendeteksi jaraknya berdasarkan jarak objek pada gambar yang diambil.

Prinsip kerja dari stereo kamera dalam mendeteksi jarak objek adalah dengan melihat perbedaan posisi objek yang didapatkan antara kedua kamera. Perbedaan posisi objek pada kamera merepresentasikan jarak objek dari kamera. Ketika mengambil gambar yang semakin dekat jaraknya dengan kamera, maka perbedaan posisi objek di kedua kamera akan semakin jauh dikarenakan sudut pandang yang berbeda. Seperti yang ditampilkan pada gambar 2.1, ketika benda semakin jauh, maka perbedaan posisi pada gambar akan semakin dekat



Gambar 2.4 Ilustrasi stereo kamera untuk mendeteksi jarak [13]

perbedaannya ($|x^l - x^r|$). Dari Gambar 2.1 untuk mencari jarak (Z) maka didapatkan Persamaan:

$$Z = \frac{f \cdot T}{x^l - x^r} \tag{2.1}$$

Dimana Z adalah jarak objek terhadap sensor kamera, f adalah panjang fokus kamera, T adalah jarak kedua kamera, x^l adalah posisi objek pada gambar kiri, dan x^r adalah posisi objek pada gambar kanan [13]. Jarak antara 2 gambar biasanya direpresentasikan dalam satuan piksel.

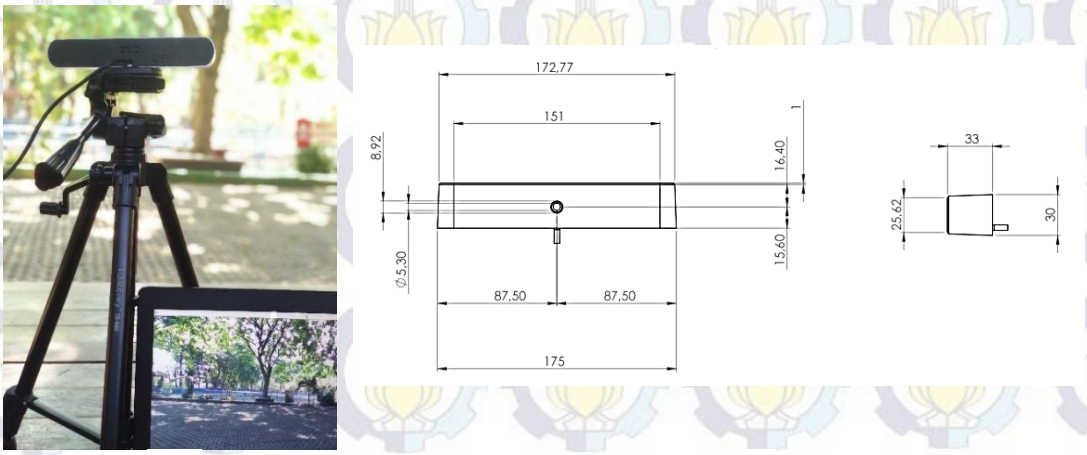
2.2.2 Kamera ZED

Kamera ZED adalah sebuah perangkat kamera stereo yang diproduksi oleh *Stereolabs*. Kamera ini berukuran relatif kecil jika dibandingkan kamera stereo lainnya seperti Bumblebee XB3, atau bahkan kamera Kinect yang juga mengukur kedalaman. Di halaman web resminya [14], kamera ini diklaim mampu mengambil gambar stereo dengan resolusi masing-masing kamera 2K. Kamera ini juga dapat merekam video dengan kecepatan hingga 100 FPS dalam resolusi WVGA. Beberapa aspek spesifikasi penting yang dimiliki kamera ZED ditampilkan pada Tabel 2.1 di bawah. Masing-masing kamera ini sendiri mampu mengambil gambar dengan resolusi hingga 2,2K atau 4416 x 1242 piksel, dan dapat merekam hingga 100 *frame* per detik. Dibantu dengan sensor bawaannya yang memiliki perbandingan ukuran 16:9, maka kamera ini dapat menangkap gambar dengan luas. Sudut pandang penglihatannya adalah 90° secara vertikal, 60° secara horizontal, dan 110° secara diagonal [14].

Tabel 2.1 Spesifikasi dan Fitur Kamera ZED

Ukuran dan berat	Dimensi: 175×30×33 mm Berat: 159 gr
Spesifikasi gambar yang diperoleh dari masing-masing kamera	HD2K: 2208 × 1242 (15 FPS) HD1080: 1920 × 1080 (30, 15 FPS) HD720: 1280 × 720 (60, 30, 15 FPS) WVGA: 672 × 376 (100, 60, 30, 15 FPS)
Format gambar kedalaman (*Depth*)	Jarak deteksi optimal: 1-20 m Format: 32 bits Jarak antar kamera: 120 mm
Spesifikasi lensa	Field of View: 110 ◦ f/2.0 aperture
Spesifikasi sensor	Ukuran: 1/3 Format: 16:9 Ukuran Piksel: 2-u pixels
Koneksi	USB 3.0 (5 V / 380 mA) 0◦C to +45◦C
Kebutuhan SDK	Windows or Linux Dual-core 2.3 GHz 4 GB RAM NVIDIA GPU

Kemampuan yang tinggi dari kamera ZED setara dengan kebutuhan kemampuan komputer yang memadai juga untuk dapat digunakan secara



(a) (b)

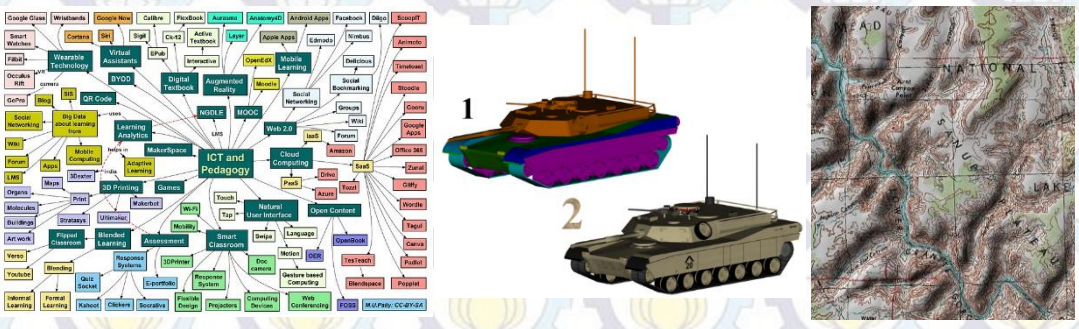
Gambar 2.5 (a) Bentuk fisik kamera ZED dan (b) dimensinya dalam milimeter

maksimum. Dari halaman web Stereolabs dikatakan bahwa koneksi kamera ke komputer harus melalui USB 3.0 untuk kelancaran transfer data dari kamera. Sedangkan spesifikasi komputer yang diperlukan untuk menjalankan *Software Development Kit* (SDK) ZED juga tinggi, prosesor yang digunakan harus memiliki kecepatan di atas 2,3 GHz, RAM dengan kapasitas 4 GB ke atas, dan harus memiliki NVIDIA GPU dengan kemampuan menghitung lebih dari 3,0 [14]. Kemampuan menghitung dari NVIDIA GPU dapat dilihat di halaman web resmi NVIDIA [15].

2.2.3 Pemetaan

Pemetaan dapat berarti banyak hal, di antaranya kartografi, yang berarti pembuatan peta, representasi simbol fitur dari berbagai bagian di permukaan bumi, atau di bidang astronomi dan imajinasi. Dalam matematika pemetaan adalah sinonim dari fungsi matematika. Dalam bidang grafik di komputer, pemetaan adalah metode untuk mendefinisikan permukaan objek, warna dan tekstur dalam bentuk objek 3 dimensi dengan detail yang tinggi. Dalam bidang robotika pemetaan termasuk dalam bidang yang berhubungan pengindraan visual komputer dan kartografi. Biasanya dalam robot tujuannya untuk mengaplikasikan ke dalam bentuk robot *autonomous* yang dapat melokalisasi dirinya sendiri terhadap peta.

Jika digabungkan pengertian dari beberapa penjelasan pada paragraf sebelumnya, maka pemetaan dapat berarti representasi simbol yang diaplikasikan dalam bidang pengindraan visual komputer dan dapat melokalisasi dirinya sendiri terhadap suatu bidang peta. Pemetaan tidak seluruhnya berbentuk peta pada



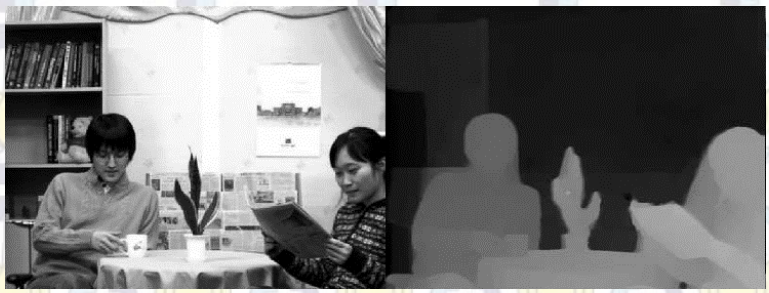
Gambar 2.6 Pemetaan pikiran (kiri), pemetaan permukaan objek 3 dimensi dengan tekstur dan tidak (tengah), dan peta topografi dengan efek bayangan sinar matahari (kanan) [30]

umumnya yang terlihat di atlas. Pemikiran juga dapat dipetakan dalam bentuk tulisan dan gambar yang dapat dimengerti orang lain seperti pada gambar 2.6.

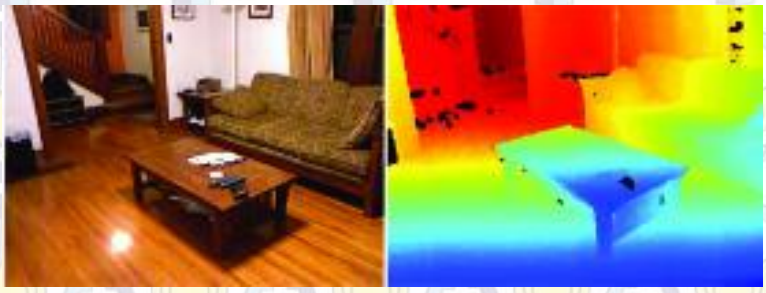
2.2.4 *Depth Image*

Depth image dalam arti harfiah adalah gambar kedalaman, dalam versi lain juga disebut sebagai *range image*. Secara makna artinya gambar yang menunjukkan jarak benda yang ditangkap terhadap kamera. Ada banyak metode yang dapat digunakan untuk mengambil *depth image*, yaitu dengan stereo kamera [16], laser [17], LIDAR [18], cahaya struktural, *Time-of-Flight* (ToF) *camera* [19] seperti pada Kinect [20], interferometri, dan masih banyak lainnya.

Pada gambar biasa, setiap piksel-nya berisi nilai warna dasar yaitu merah, hijau dan biru (RGB). Warna dasar ini ketika digabungkan dapat membentuk berbagai macam warna lainnya. Akan tetapi pada *depth image*, setiap piksel yang ditampilkan merupakan representasi dari jarak titik pada piksel tersebut terhadap kamera. Berbagai jenis tampilan dari *depth image* ada yang dalam bentuk warna, ada juga yang monokrom. Gambar 2.6 (a) merupakan *depth image* dengan warna putih yang paling dekat, kemudian makin gelap warnanya, makin jauh jaraknya



(a)



(b)

Gambar 2.7 Gambar warna biasa dan gambar depth-nya dalam bentuk (a) hitam putih dan (b) warna

[21]. Sedangkan Gambar 2.6 (b) jarak direpresentasikan dalam warna, biru merupakan jarak terdekat, dan merah merupakan jarak terjauh.

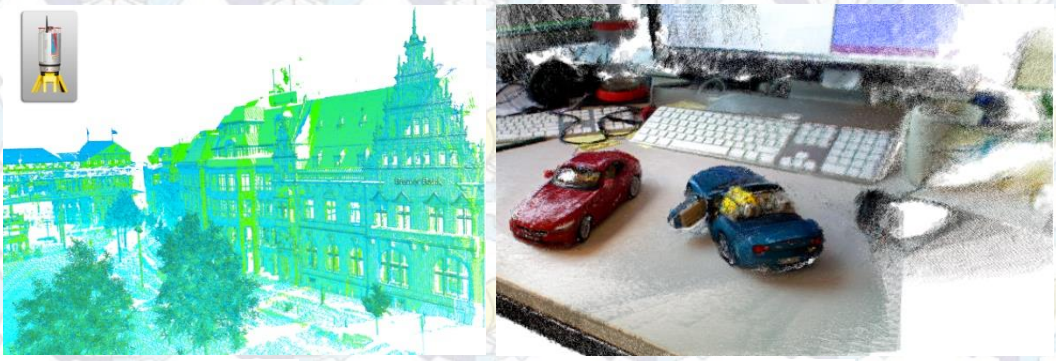
2.2.5 Point cloud

Point cloud merupakan sekumpulan titik data yang terletak di suatu bidang 3 dimensi. Biasanya *point cloud* diperoleh dari *scanner* 3D, yang mengukur banyak titik pada permukaan bidang yang dideteksinya. Sebagai keluaran dari proses *scan* 3D, *point cloud* biasanya digunakan dalam banyak bidang, termasuk membuat model CAD 3D untuk manufaktur di pabrik, untuk metrologi dan kualitas dalam inspeksi, untuk memvisualisasikan bentuk 3 dimensi, animasi, dan *rendering* juga untuk registrasi gambar 3D [22].

Untuk membuat sebuah *point cloud*, informasi utama yang diperlukan yaitu koordinat titik pada setiap sumbu x, y, dan z. Kumpulan informasi titik-titik yang di kumpulkan nantinya akan membentuk suatu objek yang dapat dikenali. Gambar 2.8 (a) di bawah merupakan bentuk kumpulan titik-titik *point cloud* yang membentuk suatu bangunan. Sedangkan informasi tambahan lainnya adalah warna. Warna pada *point cloud* adalah opsional, bukan informasi yang penting tetapi dapat mempermudah melihat bentuk *point cloud* pada layar. Gambar 2.8 (b) di bawah ini adalah contoh bentuk *point cloud* yang berwarna.

2.2.6 Navigasi

Navigasi berasal dari bahasa Yunani yang terdiri dari kata *navis* yang artinya perahu atau kapal, dan *agake* yang artinya mengarahkan. Secara harfiah artinya mengarahkan sebuah kapal dalam pelayaran. Akan tetapi dari waktu ke



(a) (b)

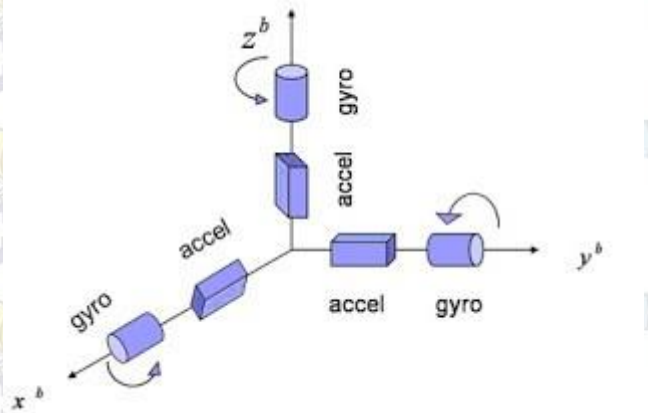
Gambar 2.8 Bentuk tampilan point cloud (a) tanpa warna dan (b) dengan warna

waktu kata navigasi mengalami perluasan makna, sehingga kata navigasi digunakan juga di daratan dan di udara. Di *Oxford Dictionary*, kata navigasi berarti proses atau aktivitas memastikan posisi seseorang, serta perencanaan mengikuti sebuah jalur baik di medan sebenarnya atau di peta [23]. Maka kata navigasi dapat digunakan untuk tunanetra dalam membantu menentukan posisinya secara akurat, dan memandu jalannya.

2.2.7 *Inertial Measurement Unit* (IMU)

Inertial measurement unit atau sering disebut juga *motion sensor* (sensor gerak), adalah sebuah alat elektronik yang dapat mengukur pergerakan suatu objek berdasarkan percepatannya. Di dalam sensor IMU, umumnya terdapat 3-axis akselerometer dan 3-axis giroskop. Akselerometer sendiri merupakan sebuah instrumen yang mengukur akselerasi dalam satu axis. Dengan menurunkan sekali, maka akan ditemukan kecepatannya. Dengan menurunkan sekali lagi, maka akan ditemukan posisinya terhadap posisi awal. Akan tetapi akselerometer tidak dapat mendeteksi posisi awal [24]. Dengan menggunakan 3 akselerometer, dan arahnya berbeda yaitu sejajar sumbu x, y dan z, maka pergerakan objek akan dapat dideteksi secara 3 dimensi.

Giroskop merupakan perangkat 1 dimensi, yang dapat memberikan keluaran berupa sinyal yang proporsional dengan pergerakan sudut yang ditempuh oleh giroskop tersebut [24]. Jika akselerometer mendeteksi pergerakan pada bidang x, y dan z secara linear, maka giroskop mendeteksi pergerakan terhadap sumbu x (*roll*, y (*pitch*) dan z (*yaw*) secara radial. Untuk gambaran yang lebih jelas, lihat



Gambar 2.9 Sensor IMU, giroskop 3 sumbu dan akselerometer 3 sumbu

pada Gambar 2.3. Beberapa sensor IMU juga memiliki 3-axis magnetometer, yang mana dapat mendeteksi medan magnet terhadap sumbu x, y dan z.

2.2.8 Lokalisasi Inersia

Lokalisasi navigasi inersia menggunakan akselerometer dan giroskop untuk mendeteksi gerakan. Akselerometer mengukur percepatan dalam ruang 3 dimensi. Perpindahan dihitung dari integral ganda percepatan. Giroskop yang dikombinasikan dengan akselerometer digunakan untuk menghitung arah hadap (*heading*) [25]. Prinsip ini disebut dengan *Dead Reckoning Technique* [26]. *Dead Reckoning* didasarkan pada penggabungan percepatan dan arah hadap selama satu langkah waktu untuk menentukan seberapa jauh dan ke mana arah benda tersebut dipindahkan dari posisi terakhir yang diketahui.

Algoritma estimasi posisi biasanya berbasis Kalman filter untuk merepresentasikan penyelesaian untuk permasalahan filter data diskrit linear. Kalman filter merupakan algoritma penghitungan sederhana, dengan menggunakan proses estimasi berbasis sistem kontrol umpan balik tertutup. Yang pertama mengestimasi kondisi proses pada sebuah titik pada waktu tertentu, kemudian mendapatkan umpan balik pengukurannya. Umpan balik pengukuran ini digunakan untuk menyesuaikan model parameter untuk estimasi berikutnya [25].

2.2.9 Matriks Transformasi 3 Dimensi

Dalam matriks transformasi 3 dimensi, ada beberapa jenis transformasi geometri yang dapat dilakukan pada titik-titik *point cloud*. Yang sering digunakan adalah translasi, skala, dan rotasi. Masing-masing transformasi tersebut memiliki matriks transformasi yang mewakili untuk proses perhitungan posisi tiap titik setelah transformasi. Matriks transformasi $[\boldsymbol{M}]$ pada titik $[x, y, z]$ akan menghasilkan titik-titik baru $[x_T, y_T, z_T]$ menggunakan perkalian matriks pada Persamaan 2.2 di bawah.

$$\begin{bmatrix} x_T \\ y_T \\ z_T \\ 0 \end{bmatrix} = \boldsymbol{M} \begin{bmatrix} x \\ y \\ z \\ 0 \end{bmatrix} \tag{2.2}$$

Matriks transformasi untuk translasi terhadap sumbu x dalam notasi (t_x), sumbu y dalam notasi (t_y), dan sumbu z dalam notasi (t_z) ditampilkan pada notasi $\boldsymbol{M_T}$. Kemudian matriks skala dengan skala terhadap sumbu x, y, dan z secara berurutan adalah (S_x, S_y, S_z) ditampilkan pada notasi $\boldsymbol{M_S}$. Sedangkan matriks untuk rotasi ada 3, yaitu rotasi terhadap sumbu x $(\boldsymbol{M_{Rx}})$, rotasi terhadap sumbu y $(\boldsymbol{M_{Ry}})$, dan rotasi terhadap sumbu z $(\boldsymbol{M_{Rz}})$ dengan perputaran sudut θ. Secara detail persamaan rotasi, translasi, dan skala ditunjukkan pada Persamaan 2.3 - 2.7.

$$\boldsymbol{M_T} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ t_x & t_y & t_z & 1 \end{bmatrix} \quad (2.3)$$

$$\boldsymbol{M_S} = \begin{bmatrix} S_x & 0 & 0 & 0 \\ 0 & S_y & 0 & 0 \\ 0 & 0 & S_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (2.4)$$

$$\boldsymbol{M_{Rx}} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(\theta) & -\sin(\theta) & 0 \\ 0 & \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (2.5)$$

$$\boldsymbol{M_{Ry}} = \begin{bmatrix} \cos(\theta) & 0 & \sin(\theta) & 0 \\ 0 & 1 & 0 & 0 \\ -\sin(\theta) & 0 & \cos(\theta) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (2.6)$$

$$\boldsymbol{M_{Rz}} = \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 & 0 \\ \sin(\theta) & \cos(\theta) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (2.7)$$

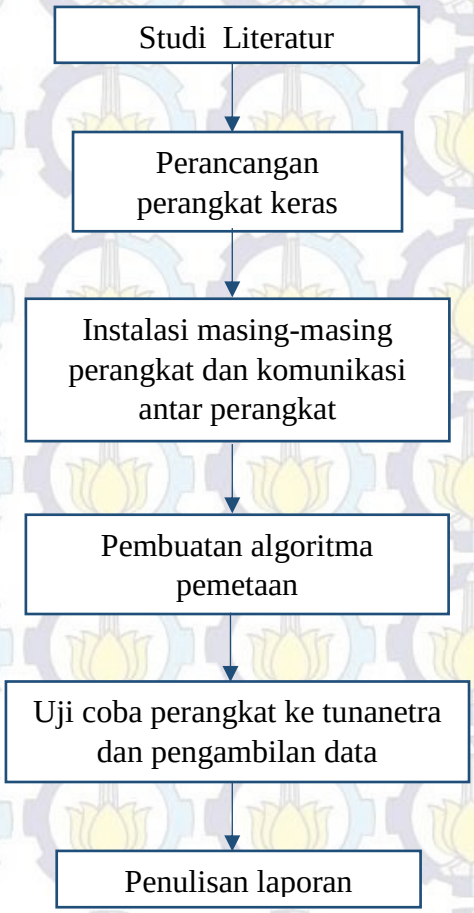
2.2.10 Komputasi Real-Time

Real-time secara bahasa berarti waktu yang sesuai dengan waktu sebenarnya. Dalam kasus komputasi, dapat berarti bahwa proses komputasi yang dilakukan berjalan dalam waktu yang seketika itu juga selesai. Komputasi *real-time* dikatakan harus merespons dalam waktu mili detik. Sistem real-time yang digunakan untuk manusia, setidaknya harus dapat merespons dengan waktu yang sama atau lebih cepat dari respons manusia. Sedangkan sebuah riset mengatakan bahwa respons manusia adalah 0.25 detik terhadap visual, 0.17 detik terhadap audio, dan 0.15 detik terhadap sentuhan [27].

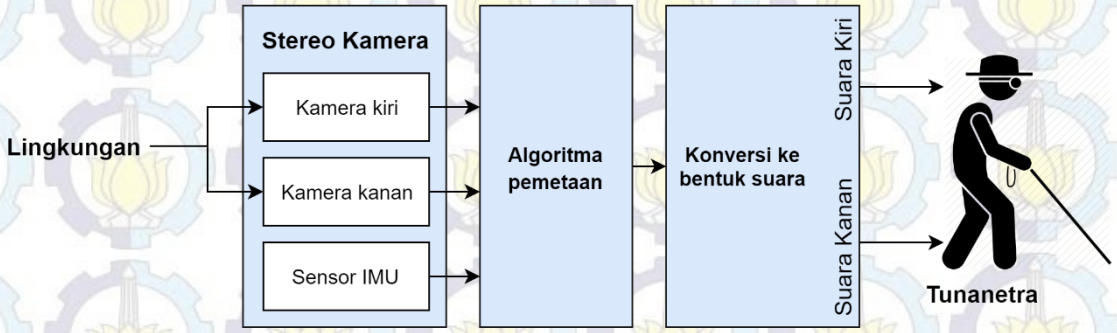
Halaman ini sengaja dikosongkan

BAB 3
METODE PENELITIAN

Pada bagian ini akan diuraikan tentang metodologi penelitian yang digunakan. Tahapan penelitian yang digunakan secara umum ditunjukkan pada Gambar 3.1. Sebelum melakukan penelitian, disini penulis melakukan studi literatur terlebih dahulu, menentukan perangkat keras apa saja yang akan dipakai, dan perangkat lunak yang mampu mendukung penelitian ini. Kamera stereo yang sudah ditentukan adalah kamera ZED dari Stereolabs. Berdasarkan spesifikasi yang tertera di halaman web resminya, kamera ini memerlukan spesifikasi komputer yang tinggi serta perangkat lunak dan bahasa pemrograman tertentu. Sehingga komputer atau laptop yang digunakan harus memiliki spesifikasi yang lebih tinggi



Gambar 3.1 Diagram alir tahapan penelitian



Gambar 3.2 Diagram blok dari keseluruhan sistem

dari yang ditentukan. Pemilihan penggunaan perangkat lunak juga ditentukan terbatas dari yang didukung oleh kamera ini.

Setiap perangkat keras dihubungkan dengan laptop yang digunakan. Perangkat keras yang digunakan pada sistem ini adalah kamera stereo dan *headset*, seperti yang ditampilkan pada diagram blok Gambar 3.2.

3.1 Kamera ZED

Bagian pertama dari keseluruhan sistem pemetaan halangan ini didapatkan dari gambar yang diambil oleh kamera stereo. Kamera stereo yang kami gunakan pada penelitian ini adalah kamera ZED. Kamera ini merupakan kamera khusus yang dibuat oleh StereoLabs, dan berbeda dengan kamera biasa lainnya. Kamera ini memiliki dukungan *software* dari Robot Operating System (ROS), ROS2, OpenCV, Matlab, TensorFlow, serta *plugin* untuk Unity dan Unreal Engine. Jika kamera ini digunakan oleh Developer atau pengembang perangkat lunak, bahasa pemrograman yang digunakan yang sudah ada secara resmi adalah bahasa C++ dan Python. Pada penelitian ini penulis menggunakan bahasa Python sebagai bahasa pemrograman pendukung. Untuk memulai menggunakan kamera ZED di Python, ada beberapa hal yang perlu di-*download* dan di-*install* pada sistem operasi komputer [28], yaitu:

- ZED SDK
- ZED Python API
- Python 3.5+ (64-bit)
- Cython 0.26+
- Numpy 1.13+
- C++ compiler

Cara instalasi ZED SDK dan Python dapat dilakukan dengan cara instalasi biasa seperti program Windows pada umumnya. Cython dan Numpy dapat di-*install* menggunakan PIP pada Python setelah aplikasi Python ter-*install* dengan benar. Sedangkan untuk ZED Python API dapat di-*install* setelah yang lain selesai dengan mengunduhnya di halaman web `https://github.com/stereolabs/zed-python-api`. Tata cara instalasi lengkap dapat dilihat di halaman web resmi Stereolabs [28].

Setelah instalasi selesai, untuk memastikan bahwa semua aplikasi telah ter-*install* dengan benar dapat mencoba contoh program yang tersedia di halaman web `https://github.com/stereolabs/zed-python-api/tree/master/examples`. Beberapa contoh sudah disediakan untuk uji coba seluruh fitur dari kamera, seperti pengambilan gambar kiri, kanan, *Depth Image*, dan pengambilan data asli dari sensor IMU.

3.2 Proses Pemetaan

Keseluruhan proses pemetaan ini dilakukan di PC, dengan susunan konstruksi *depth image*, penyederhanaan gambar, pengambilan posisi dari nilai terdekat setiap titik, dan konversi posisi titik ke dalam bentuk suara. Setiap prosesnya akan dijelaskan secara detail pada bagian selanjutnya.

3.2.1 Spesifikasi Komputer

Komputer atau PC yang digunakan dalam penelitian ini adalah laptop X456UF dengan spesifikasi sebagai berikut:

- Prosesor : Intel Core i5-6200U @2.3 GHz (4 CPU)
- Memori (RAM) : 12288 MB
- Kartu grafis : Intel HD Graphics 520 dan NVIDIA GeForce 930M
- Sistem Operasi : Windows 10 Pro 64-bit (10.0, Build 17134)
- *Software* : Python 3.6.5 64-bit

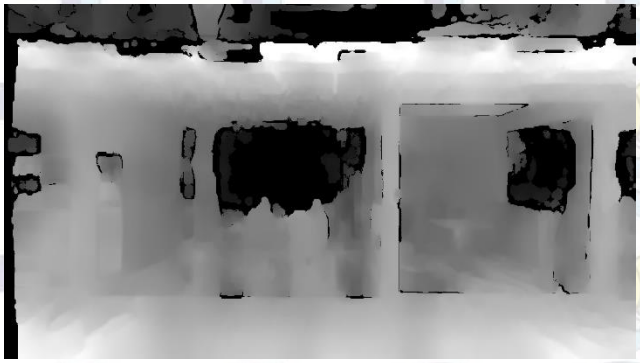
3.2.2 Konstruksi *Depth Image*

Depth image, atau yang disebut sebagai gambar kedalaman adalah gambar yang merepresentasikan jarak setiap titiknya terhadap kamera. Berbeda dengan

gambar biasa yang merepresentasikan setiap warna yang ditangkap. *Depth image* diperoleh dari dua gambar biasa yang ditangkap oleh dua buah kamera yang identik, akan tetapi memiliki jarak tertentu secara horizontal. Ukuran gambar pada gambar kedalaman sama dengan ukuran gambar yang diambil oleh kamera stereo. Dalam penelitian ini kami menggunakan resolusi 1080p (1920 x 1080 piksel). Pada kamera ZED, perhitungan perkiraan kedalaman gambar diperoleh dari perbedaan posisi dari kedua buah gambar. Sedangkan konstanta yang diperlukan untuk perhitungan sudah dikalibrasikan menggunakan alat khusus di pabrik pembuatan. Sehingga pengguna kamera ini hanya perlu mengunduh data kalibrasi berdasarkan nomor serial setiap kamera.

Salah satu contoh dari konstruksi gambar kedalaman yang didapatkan dari kamera stereo ditampilkan pada Gambar 3.3 di atas. Gambar 3.3 (a) adalah dua buah gambar kiri dan kanan yang disusun berjajar, kemudian Gambar 3.3 (b) adalah gambar kedalaman yang dikonstruksi dari kedua gambar di sebelah kiri. Pada Gambar 3.3 (a), gambar yang ditampilkan merupakan gambar biasa seperti yang ditangkap oleh kamera pada umumnya. Gambar kedalaman pada Gambar 3.3 (b)

(a)



(b)

Gambar 3.3 Gambar stereo (atas) dan rekonstruksi *depth image*-nya (bawah)

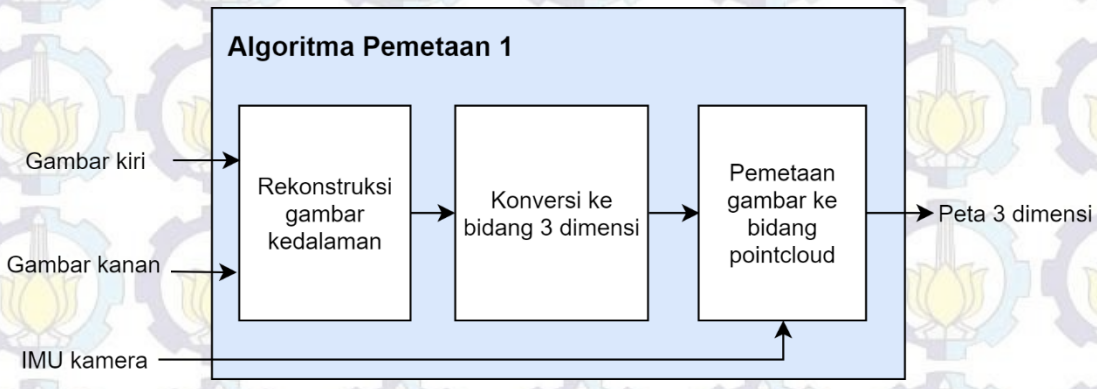
adalah gambar hitam-putih atau bernilai 0-255 yang merepresentasikan jarak di setiap piksel yang ditangkap ke kamera. Warna putih (255) melambangkan titik tersebut dekat dengan kamera, sedangkan makin hitam (0) warnanya, makin jauh jaraknya. Akan tetapi tidak semua titik dapat diprediksi jaraknya. Ada 3 jenis titik yang bernilai bukan angka [29], yaitu:

1. *NAN* yang merepresentasikan *OCCLUSION_VALUE*, artinya nilai jarak pada titik itu tidak dapat dihitung karena gambar bertabrakan atau berada di luar batas.
2. *–INFINITE* yang merepresentasikan *TOO_CLOSE*, artinya titik tersebut tidak dapat ditentukan jaraknya karena terlalu dekat dengan kamera.
3. *INFINITE* yang merepresentasikan *TOO_FAR*, artinya titik tersebut tidak dapat ditentukan jaraknya karena terlalu jauh dengan kamera.

Ketiga definisi tersebut ditampilkan pada gambar dengan nilai 0 atau sangat hitam.

3.2.3 Algoritma Pemetaan 1

Algoritma pertama yang digunakan adalah dengan memetakan ruangan dalam bentuk 3 dimensi. Kemudian dicari jalan yang tidak terhalang objek untuk menavigasikan tunanetra ke arah tersebut. Pada bagian ini, gambar stereo yang didapatkan kemudian dipetakan dalam bentuk 3 dimensi. Dengan menggunakan kamera ZED, ada 4 bentuk data yang dapat diperoleh. Gambar yang didapatkan ada 3, yaitu gambar warna kanan, gambar warna kiri, dan gambar kedalaman. Kemudian ada informasi posisi (p_x, p_y, p_z) yang didapatkan yaitu dalam bentuk



Gambar 3.4 Algoritma pemetaan 1, rekonstruksi gambar stereo dan sensor IMU ke bentuk peta 3D

koordinat x, y, dan z pada bidang kartesian 3 dimensi dengan satuan meter. Gambar kedalaman merupakan nilai relatif jarak yang ada. Sedangkan jarak pastinya dapat diperoleh dengan menggunakan triangulasi. Persamaan untuk mendapatkan jarak ditampilkan pada Persamaan 3.1,

$$D = \sqrt{p_x^2 + p_y^2 + p_z^2} \tag{3.1}$$

dimana D adalah jarak titik dengan kamera, p_x adalah koordinat titik pada sumbu x, p_y adalah koordinat titik pada sumbu y, dan p_z adalah koordinat titik pada sumbu z. Pada *point cloud* ini posisi kamera berada di koordinat (0,0,0).

Keseluruhan algoritma pada proses rekonstruksi 3 dimensi dari kamera ZED adalah sebagai berikut.

```
def pemetaan3D(jml_frame):
    input:  - jml_frame = Jumlah frame yang akan diambil
    output: - PCD = Kumpulan titik-titik point cloud.

    ZEDcam = ZED.inisialisasi()
    # Inisialisasi point cloud kosong
    PCD = Open3D.PointCloud()
    for n in range(jml_frame):
        # Ambil gambar warna dengan kamera biasa
        ZED.retrieveImage(image, ZED.VIEW_LEFT)
        # Ukuran gambar dalam Tinggi(T), Lebar(L), Warna (D)
        T, L, D = image.shape
        # Me-reshape gambar warna menjadi bentuk 1 dimensi
        informasiWarna = reshape(image, [T*L,D])

        # Ambil posisi tiap piksel dalam koordinat 3D
        ZED.retrieveMeasure(depth_map, ZED. MEASURE_DEPTH)
        Frame = sqrt(depth_map[0] * depth_map[0] +
                     depth_map[1] * depth_map[1] +
                     depth_map[2] * depth_map[2])
        T, L, D = Frame.shape
        Frame = reshape(Frame, [T*L,D])
        # Ambil posisi kamera relatif terhadap posisi awal
        ZED.zed_pose.pose_data(M)

        Frame_transformasi = Frame * M
        # Tambahkan pointcloud ke variabel PCD
        PCD.append(Frame_transformasi)
        PCD.colors = informasiWarna

    Tutup kamera ZED
    return PCD
```

Proses pemetaan membutuhkan beberapa foto dari gambar kedalaman yang direkonstruksi menjadi bentuk *point cloud*, kemudian digabungkan dengan

point cloud yang lain. Untuk menggabungkan 2 buah rekonstruksi ada beberapa metode yang dapat digunakan. Salah satu metode yang penulis pakai adalah lokalisasi dengan memanfaatkan sensor IMU (*Inertial Measurement Unit*). Sensor IMU ini untuk mengetahui posisi dan arah kamera relatif terhadap posisi awal menggunakan estimasi Lokalisasi. Sehingga dengan mengetahui posisinya, maka dapat dihitung matriks transformasi untuk mentransformasi *pointcloud* yang selanjutnya.

Pada *library* bawaan dari kamera ZED, sudah ada fitur untuk mengambil gambar kanan, kiri, dan *depth* dari kamera. Selain gambar, dengan *library* ini juga dapat mengambil nilai pergerakan sensor IMU, ataupun matriks transformasinya yang sudah dikonversi secara internal untuk mengetahui posisi relatif dari saat pertama kali menyala. Pengambilan gambar warna dapat dilakukan dengan perintah:

```
zed.retrieveImage(image, MODE).
```

Pada variabel `MODE` gambar yang dapat diambil ada 3 macam, yaitu `"VIEW_LEFT"` yang artinya mengambil gambar kiri, `"VIEW_RIGHT"` mengambil gambar kanan, dan `"VIEW_BOTH"` mengambil gambar kiri dan kanan bersamaan kemudian dijadikan satu. Mode yang digunakan adalah `"VIEW_LEFT"`, kemudian gambar tersebut disimpan ke dalam variabel `image`. Kemudian pengambilan posisi terhadap bidang 3 dimensi (x, y, z) dapat dilakukan dengan perintah:

```
zed.retrieveMeasure(depth_map, MEASURE_DEPTH)
```

perintah tersebut adalah untuk mengambil posisi setiap piksel dalam bidang 3 dimensi. Hasil dari fungsi di atas dimasukkan ke dalam variabel `depth`, yang mana di dalam variabel tersebut berisi matriks dengan dimensi $(T \times L \times 4)$ dengan T dan L adalah dimensi gambar berupa tinggi dan lebar. Sedangkan 4 angka di dalam setiap piksel yang ditangkap berisi informasi posisi pada bidang kartesian 3 dimensi (x, y, z, α), dengan x, y dan z adalah setiap sumbu pada bidang 3 dimensi, dan α adalah arah. Dalam hal ini karena yang diambil merupakan titik, jadi nilai $\alpha = 0$.

Untuk menentukan jarak dari posisi titik (x, y, z) pada bidang 3 dimensi terhadap kamera dengan asumsi posisi kamera (x_0, y_0, z_0) berada pada titik yang diketahui, maka diperlukan penghitungan dengan mentransformasi posisi kamera

ke titik (0,0,0), kemudian dihitung jarak tiap titik terhadap kamera menggunakan Persamaan 3.1.

Kemudian untuk matriks transformasi yang didapatkan berdasarkan pergerakan dan perpindahan yang dideteksi oleh sensor IMU terhadap titik awal dapat diperoleh dengan menggunakan perintah:

```
zed_pose.pose_data(M).
```

Di dalam matriks transformasi (M) ini sudah berisi nilai translasi terhadap sumbu x, y dan z (t_x, t_y, t_z) dan rotasinya terhadap sumbu x, y dan z dengan sudut (α, β, γ) dalam format matriks transformasi 3 dimensi.

Dalam menyusun pemetaan 3D, ada beberapa gambar kedalaman yang akan direkonstruksi diambil secara bersamaan dengan matriks transformasi. Gambar kedalaman yang sudah dikonversi ke dalam bentuk *point cloud* kemudian dikalikan dengan matriks transformasi (M) agar posisinya sesuai dengan referensi posisi awal. Masing-masing *point cloud* yang sudah ditransformasi kemudian disusun dan dijadikan dalam satu variabel.

Pada *library* Open3D di *Python*, untuk menampilkan konstruksi 3D variabel yang dibutuhkan adalah posisi tiap titik (p_x, p_y, p_z) menjadi format *point cloud*. Sedangkan untuk pemberian warna adalah opsi tambahan jika diperlukan. Jika diinginkan pemberian warna, maka dimensi gambar diubah menjadi bentuk 1 dimensi yang berisi informasi warna 3 variabel (*Red*, *Green*, *Blue*) dalam skala 0 sampai 1. Ukuran dimensi warna harus sama dengan ukuran dimensi *point cloud*. Keluaran dari algoritma ini adalah peta dalam bentuk titik-titik *point cloud* yang merepresentasikan posisi masing-masing titik dalam bentuk 3 dimensi.

3.2.4 Algoritma Pemetaan 2

Algoritma lain yang digunakan adalah dengan mengolah *depth image* atau *depth map* untuk secara langsung menavigasikan tuna netra dengan suara. Proses ini secara teori berjalan lebih cepat karena hanya memproses bidang 2 dimensi, yaitu *depth image*, yang kemudian hasilnya langsung dikonversi ke bentuk suara. Pada algoritma ini tidak ada bagian 3 dimensi sama sekali untuk mempercepat komputasi oleh komputer. Keseluruhan pemrosesan ini dipercepat dengan menghitung langsung dari *depth image* untuk dipetakan. Susunan pengolahan yang

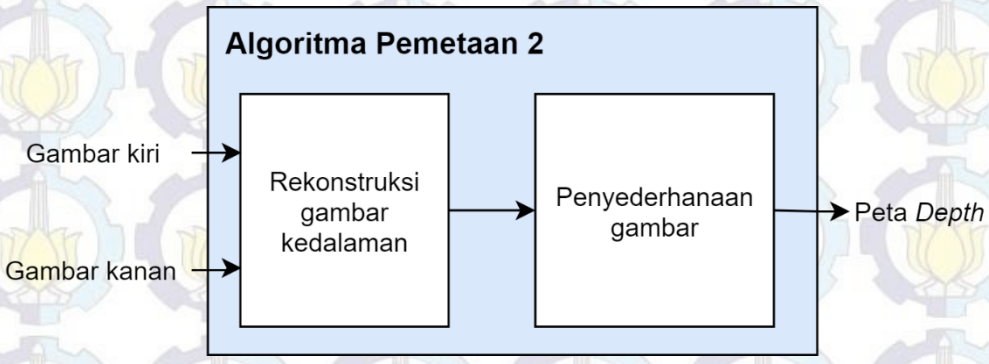
dilakukan ditampilkan pada Gambar 3.5, dan algoritmanya dalam *pseudo code* adalah sebagai berikut:

```
def pemetaanDepth():
    import ZED
    import pygame
    input:  - ZEDcam = kamera yang digunakan

    output: - depth_map = peta kedalaman (depth map)

    ZEDcam = ZED.inisialisasi()
    while(True):
        # Ambil gambar depth
        ZED.retrieveMeasure(depth, MEASURE_DEPTH)
        # Ukuran gambar dalam Tinggi(T), Lebar(L), Warna (D)
        depth_image =
        T, L, D = image.shape
        rT = T/8
        rL = L/8
        for i in range(8):
            for j in range(8):
                depth_map[i,j] = min(depth[i*rL:(i+1)*rL,
                                           j*rL:(j+1)*rL])

    return depth_map
```

Masukan dari kamera adalah gambar stereo kiri dan kanan yang sudah direkonstruksi menjadi *depth image*. Pengambilan gambar *depth image* ini dapat dilakukan dengan perintah:

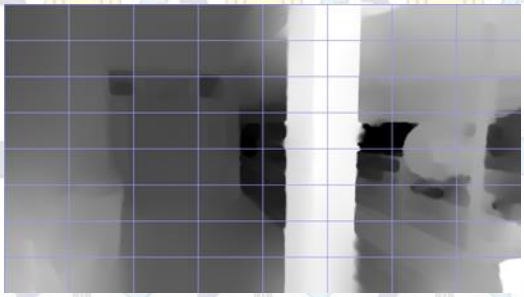
```
zed.retrieveMeasure(depth, MEASURE_DEPTH).
```

Perintah tersebut adalah untuk mengambil posisi setiap piksel dalam bidang 3 dimensi. Hasil dari fungsi di atas dimasukkan ke dalam variabel `depth`, yang mana di dalam variabel tersebut berisi matriks dengan dimensi $(T \times L \times 4)$ dengan T dan L adalah dimensi gambar berupa tinggi dan lebar. Sedangkan 4 angka di dalam



Gambar 3.5 Algoritma 2, menggunakan pemrosesan gambar kedalaman

(a)



(b)

(c)

Gambar 3.6 Proses pengolahan *depth map* pada algoritma pemetaan ke-2

setiap piksel yang ditangkap berisi informasi posisi pada bidang kartesian 3 dimensi dalam bentuk (x, y, z, α), dengan x, y dan z adalah setiap sumbu pada bidang 3 dimensi, dan α berisi nilai $\alpha = 0$.

Depth image yang sudah diperoleh seperti Gambar 3.6 (b) kemudian disederhanakan menjadi matriks dengan resolusi 8x8 seperti pada Gambar 3.6 (c). Matriks tersebut diperoleh dari *depth image* yang dibagi menjadi 8 bagian secara horizontal, dan 8 bagian secara vertikal. Maka *depth image* tersebut akan menjadi 64 bagian. Kemudian setiap bagian diambil nilai jarak terdekatnya untuk kemudian dimasukkan ke dalam matriks 8x8. Dalam hal ini penulis menentukan ukuran matriks 8x8 sebagai salah satu bentuk yang cukup sederhana, tetapi masih dapat terlihat polanya dan masih dapat dipahami. Penyederhanaan gambar ini juga mempercepat pemrosesan selanjutnya karena bagian matriks yang perlu dihitung jauh lebih kecil dibandingkan dengan resolusi gambar kedalaman yang didapatkan sebesar 1920 x 1080 piksel.

3.3 Konversi ke Bentuk Suara

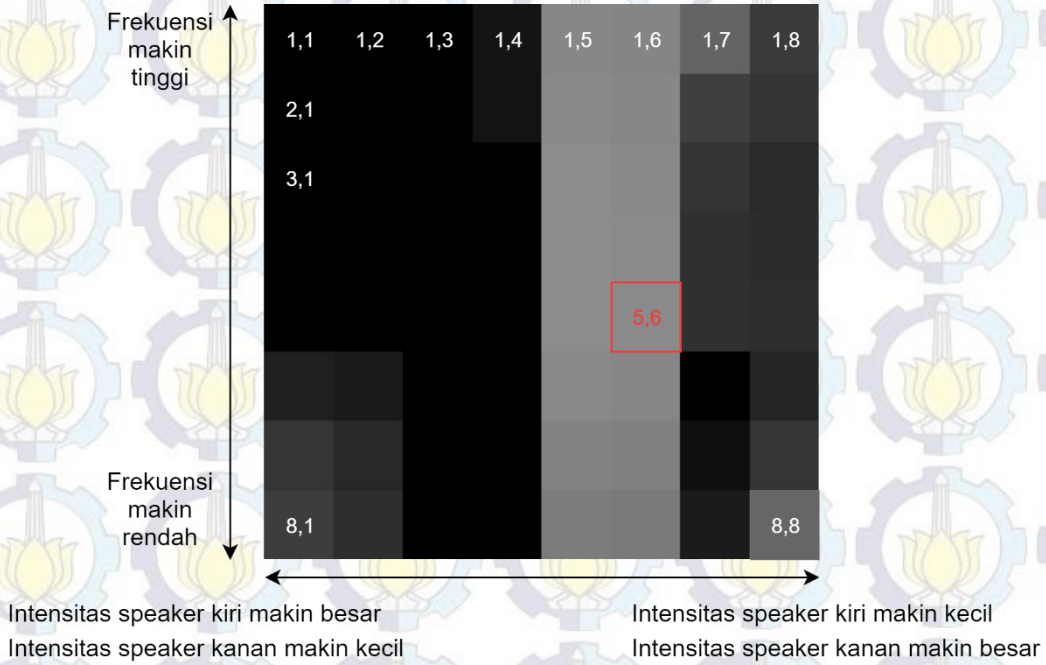
Dari peta yang keluar pada algoritma pemetaan akan dihitung jarak objek terdekatnya terhadap kamera. Pada bidang 3 dimensi, perhitungan jarak dapat

diperoleh menggunakan Persamaan 3.1 jika posisi kamera saat ini dan posisi tiap-tiap objek diketahui. Pada algoritma pemetaan yang kedua, *depth map* yang dihasilkan sudah merupakan representasi jarak terhadap kamera. Sehingga untuk menentukan posisi terdekatnya dapat diketahui dengan menggunakan fungsi berikut:

```
[x,y] = argmax(depth_map).
```

Nilai keluaran posisi terdekat adalah nilai posisi koordinat. Koordinat ini direpresentasikan sebagai (x, y) dimana x adalah posisi secara vertikal, dengan 1 adalah posisi paling atas, dan 8 adalah posisi paling bawah. Sedangkan y adalah posisi horizontal dengan 1 adalah posisi paling kiri, dan 8 adalah posisi paling kanan. Setiap koordinat memiliki suara yang berbeda-beda tergantung pada posisinya.

Pada penelitian ini nada suara yang digunakan merupakan sinyal sinusoid dengan frekuensi dan amplitudo yang berbeda-beda. Seperti yang ditampilkan pada Gambar 3.7, frekuensi suara menunjukkan posisi pada bidang vertikal, sedangkan amplitudo merupakan jarak objek terhadap kamera. Untuk membedakan posisi objek secara horizontal, suara yang dikeluarkan diatur amplitudonya dan dibuat



Gambar 3.7 Koordinat pada *depth map* yang disederhanakan

Tabel 3.1 Konversi Posisi ke Bentuk Suara

Nomor baris atau kolom	**Volume *speaker* kiri (skala 0-1)**	**Volume *speaker* kanan (skala 0-1)**	**Frekuensi suara (Hz)**	**Kekerasan (%)**
1	0.9	0.1	860	4
2	0.77	0.23	720	6
3	0.63	0.37	600	9
4	0.5	0.5	500	13
5	0.5	0.5	400	20
6	0.37	0.63	300	30
7	0.23	0.77	200	50
8	0.1	0.9	100	100

berbeda antara kiri dan kanan. Ketika objek berada di sebelah kanan, maka rasio suara di *speaker* kanan akan jauh lebih besar dari *speaker* kiri, begitu juga sebaliknya. Ketika objek semakin ke tengah, maka suara kiri dan kanan akan semakin setara amplitudonya. Nilai frekuensi, rasio, dan amplitudo suara secara detail ditampilkan pada Tabel 3.1.

Posisi paling kiri merupakan kolom nomor 1. Posisi berdasarkan letak horizontal dibedakan dengan intensitas suara yang berbeda di *speaker* kanan dan kiri. Pada Tabel 3.1 di atas ditampilkan bahwa ketika objek berada di kolom 1, maka rasio kiri-kanan adalah 0.1. Artinya adalah volume *speaker* di sebelah kanan 0.1 dari 1, dan di sebelah kiri 0.9 dari 1. Ketika posisi horizontal di kolom nomor 8, maka nilai rasionya adalah 0.9, artinya volume *speaker* di sebelah kanan adalah 0.9, dan volume di sebelah kiri 0.1. Ketika objek berada di tengah atau sekitar kolom 4 atau 5, maka volume di kiri dan kanan sama besarnya.

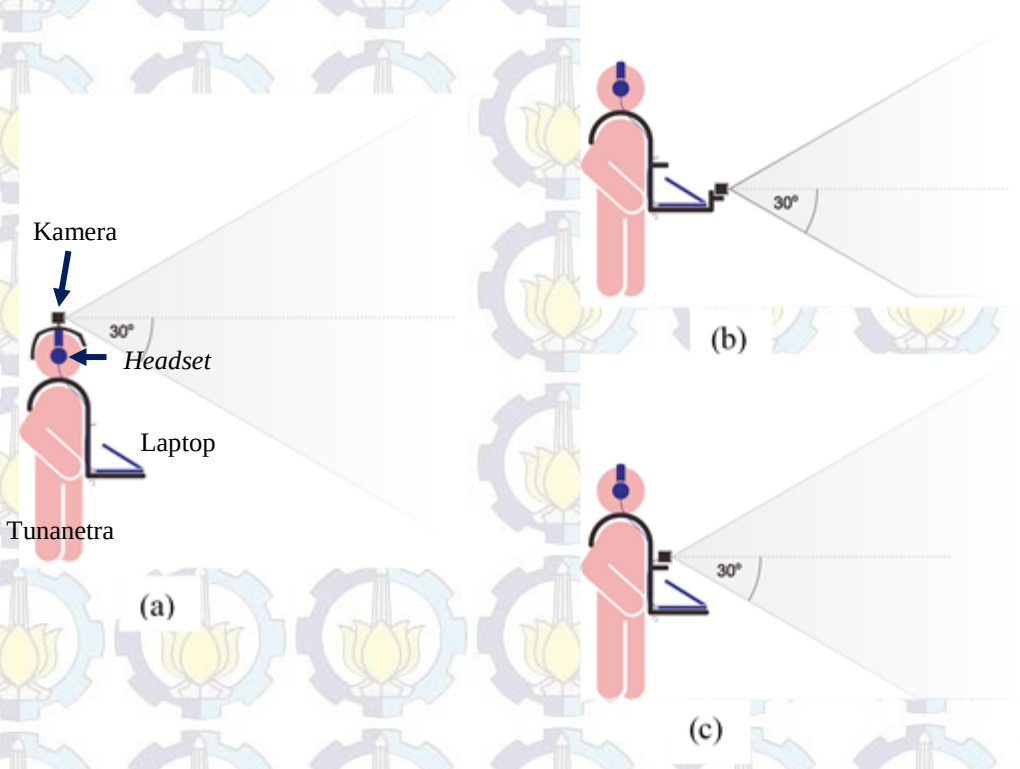
Gambar 3.7 menunjukkan salah satu contoh *frame* yang diambil. Pada gambar tersebut nilai jarak terdekatnya berada pada koordinat $(x, y) = (5, 6)$. Dengan posisi x = 5, maka frekuensi suara yang dikeluarkan sesuai tabel adalah sinusoid 400 Hz. Kemudian pada posisi horizontal, koordinat y menunjukkan angka 6, sehingga sesuai pada tabel maka amplitudo pada *speaker* kanan dikalikan sebesar

0.63, dan pada *speaker* kiri amplitudonya dikalikan 0.37. sinyal suara kiri dan kanan yang sudah diperoleh ini kemudian dikalikan dengan angka pada kolom kekerasan pada Tabel 3.1. Angka kekerasan ini nilainya semakin tinggi jika objek semakin dekat, dan semakin kecil bahkan menghilang jika semakin jauh. Jarak maksimal yang ditentukan untuk bunyi adalah 4 meter. Jika jarak yang didapatkan adalah 1,65 meter maka dengan normalisasi jarak, suara yang keluar adalah pada kategori 6, yaitu 30%. Maka masing-masing suara pada kiri dan kanan dikalikan 30%.

Keluaran sinyal suara dari proses konversi dikirimkan ke tunanetra menggunakan *speaker* menjadi bentuk suara. *Speaker* yang digunakan harus *speaker* stereo yang mampu mengeluarkan suara kiri dan kanan dengan berbeda. Pada tesis ini kami menggunakan *Headset* JBL T450 sebagai *speaker*-nya.

3.4 Konfigurasi Peletakan Kamera

Pada penelitian ini ada 3 konfigurasi peletakan kamera yang digunakan. Yang pertama kamera diletakkan di atas kepala setelah *headset* dengan menghadap ke depan. Sesuai dari spesifikasi dari kamera yaitu memiliki sudut pandang ke bawah sebesar 30° dari garis horizontal, posisi kamera diilustrasikan pada Gambar



Gambar 3.8 Peletakan posisi perangkat keras terhadap pengguna

3.8 (a). Konfigurasi kedua adalah kamera diletakkan di depan badan tunanetra, akan tetapi diberikan jeda sejauh laptop seperti pada Gambar 3.8 (b). Konfigurasi yang ketiga kamera diletakkan tepat di depan dada pengguna, dan di atas laptop. Agar pandangan kamera dapat melihat dengan lebih luas dan tidak terhalang oleh laptop. Konfigurasi yang ke tiga adalah peletakan kamera di depan dada, tanpa jeda laptop seperti ilustrasi Gambar 3.8 (c). Peletakan yang ketiga ini diharapkan kamera mampu mendeteksi dengan sudut pandang yang lebih luas, dan mendeteksi halangan yang berada sedikit di samping pengguna.

BAB 4
HASIL DAN PEMBAHASAN

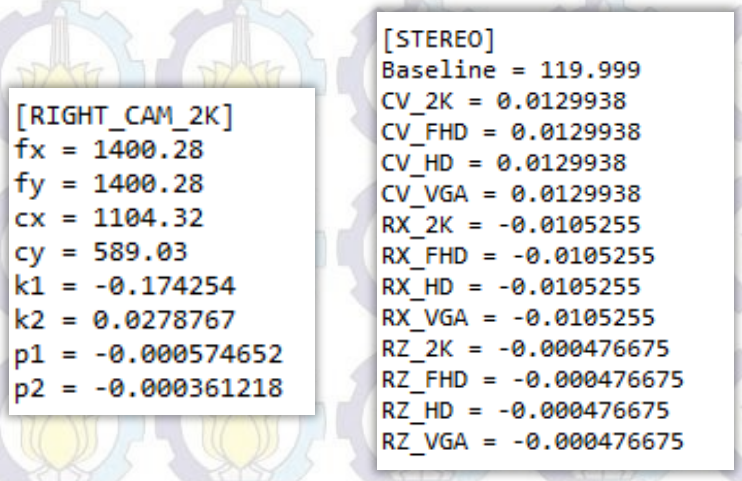
4.1 Pengaturan Perangkat Keras

Langkah awal uji coba pada penelitian ini adalah dengan mengatur perangkat keras terlebih dahulu, berupa kalibrasi kamera, kemudian pengaturan perangkat lunak.

4.1.1 Kalibrasi

Di halaman *support* resmi Stereolabs dikatakan bahwa setiap kamera ZED yang mereka kirim sudah dikalibrasikan di pabrik menggunakan perangkat khusus, maka dari itu setiap penggunanya tidak memerlukan kalibrasi lagi. Data kalibrasi setiap kamera disimpan di halaman web Stereolabs, dan akan diunduh secara otomatis saat kamera ZED pertama kali terkoneksi dengan aplikasi ZED SDK.

File kalibrasi ZED mengandung parameter intrinsik untuk masing-masing sensor kanan dan kiri pada setiap resolusinya, dan parameter ekstrinsik yang merepresentasikan hubungan antara sensor kamera kanan dan kamera kiri. *File* kalibrasi ini berbeda pada setiap kamera, yang ditandai berdasarkan nomor serialnya. Sebagian dari isi *file* kalibrasi pada kamera yang digunakan pada penelitian ini ditampilkan pada Gambar 4.1. Pada parameter intrinsik berisi panjang



(a) (b)

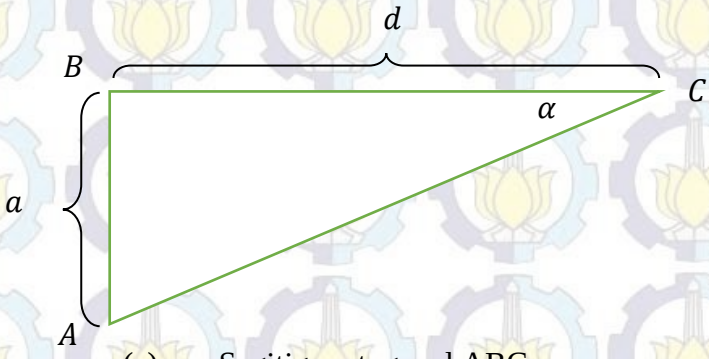
Gambar 4.1 Isi file kalibrasi (a) intrinsik, dan (b) ekstrinsik

fokal lensa dalam piksel (fx dan fy), koordinat tengah optik dalam piksel (cx dan cy), dan parameter distorsi (k1 dan k2). Sedangkan pada parameter ekstrinsik berisi *Baseline* atau jarak antara kedua kamera dalam milimeter, konvergensi optik pada tiap resolusi (CV_*** atau RY_***), dan sumbu rotasi yang mendeskripsikan transformasi antar kamera pada tiap sumbu (RX_*** dan RZ_***). Rotasinya direpresentasikan dalam notasi *Rodriguez* [29].

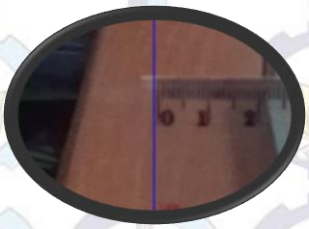
4.1.2 Bidang pandang

Di halaman web resmi Stereolebs dikatakan bahwa bidang pandang maksimum dari kamera ZED adalah 90° secara horizontal, 60° secara vertikal, dan 110° secara diagonal. Akan tetapi sudut pandang kamera setiap resolusi pada kenyataannya ada kemungkinan perbedaan dengan yang diklaim di halaman resminya. Untuk itu uji coba sudut pandang ini diperlukan agar mengetahui batasan dari kamera yang digunakan.

Di percobaan ini, kami menghitung sudut pandang menggunakan teori trigonometri. Kamera diletakkan pada bidang yang datar, kemudian sebuah



(a) Segitiga ortogonal ABC



(c) Titik tengah gambar

(b) Gambar yang diambil dengan penggaris 40 cm di depan kamera.

(d) Ujung kanan dari penggaris

Gambar 4.2 Penghitungan bidang pandang dengan penggaris

penggaris panjang diletakkan tepat di depan kamera dengan jarak (d) 40 cm. Gambar 4.2 (b) adalah pandangan yang diambil dari kamera. Garis biru horizontal adalah garis bantu yang menandakan penggaris tersebut tegak lurus dengan arah hadap kamera. Sedangkan garis biru vertikal adalah garis bantu yang menentukan titik tengah dari gambar, atau diasumsikan titik tengah lensa. Titik tengah ini dijadikan acuan untuk meletakkan penggaris pada titik 0 cm, dan menentukan angka penggaris terluar yang dapat terlihat. Angka tersebut menentukan panjang a dalam centimeter.

Jika panjang a dan panjang d diketahui, maka sudut α dapat dihitung dengan Persamaan 4.1.

$$\alpha = \text{atan}\left(\frac{a}{d}\right) \tag{4.1}$$

Bidang pandang horizontal (FOV) dari kamera ZED adalah sudut antara ujung kanan dan ujung kiri. Sedangkan sudut α hanya sudut antara ujung kanan dengan tengah kamera. Sehingga untuk menghitung bidang pandang horizontal dapat dilakukan dengan Persamaan 4.2.

$$FOV = 2 * \alpha \tag{4.2}$$

Akan tetapi setiap bingkai resolusi memiliki bidang pandang yang berbeda-beda berdasarkan eksperimen yang sudah dilakukan. Perhitungan bidang pandang tiap resolusi lebih jelasnya ditampilkan pada tabel 4.1. Perhitungan bidang pandang ini dilakukan pada lensa kiri dan kanan, yang mana pada lensa kanan ada pergeseran sejauh 12 cm dikarenakan posisi kamera kanan dengan kamera kiri berjarak *baseline* 12 cm. Ujung kanan dari lensa kanan disebutkan dalam notasi

Tabel 4.1 Posisi Titik A dan B Tiap Resolusi dan Masing-masing Kamera

	Titik terjauh di setiap resolusi		
	HD2K (2208 x 1242)	**HD1080 (1920 x 1080)**	**HD720 (1280 x 720)**
$\boldsymbol{A_{kiri}}$	31 cm	26,7 cm	37 cm
$\boldsymbol{A_{kanan}}$	43 cm	38,2 cm	49 cm
$\boldsymbol{B_{kiri}}$	0 cm	0 cm	0 cm
$\boldsymbol{B_{kanan}}$	12 cm	12 cm	12 cm

Tabel 4.2 Sudut Pandang Tiap Resolusi

	Resolusi		
	HD2K (2208 x 1242)	**HD1080 (1920 x 1080)**	**HD720 (1280 x 720)**
$\boldsymbol{\alpha_{kiri}}$	37.77°	33.22°	42.76°
$\boldsymbol{\alpha_{kanan}}$	37.77°	33.22°	42.76°
$\boldsymbol{FOV_{kiri}}$	75.55°	66.44°	85.53°
$\boldsymbol{FOV_{kanan}}$	75.55°	66.44°	85.53°

A_{kanan}, sedangkan titik tengah dari lensa kanan disebutkan dalam notasi B_{kanan}. Sedangkan untuk kamera kiri, ujung kanannya adalah A_{kiri}, dan titik tengahnya adalah B_{kiri}.

Ketika jarak titik *A* ke *B* atau panjang a diketahui, maka sudut α dapat diperoleh berdasarkan Persamaan 4.1. Begitu juga dengan sudut pandang (FOV) yang dapat diketahui menggunakan Persamaan 4.2. Hasil perhitungan sudut α dan FOV ditampilkan pada Tabel 4.2.

Dengan diperolehnya sudut pandang setiap resolusi pada Tabel 4.2 di atas, maka dapat disimpulkan bahwa resolusi HD720 merupakan resolusi dengan sudut pandang paling lebar, dan paling mendekati spesifikasi yang ditampilkan di halaman web Stereolabs.

4.1.3 Peletakan Perangkat Keras

Ada 3 buah perangkat keras yang digunakan pada sistem ini. Yaitu sebuah kamera stereo, sebuah laptop, dan sebuah *headset*. Sudah ada 3 model posisi yang dicobakan pada pengguna tunanetra. Posisi pertama adalah *headset* diletakkan seperti biasa, yaitu di atas kepala dengan letak *speaker* tepat di sebelah telinga, kemudian kamera stereo diletakkan di atas *headset*, dan laptop dengan keadaan dipegang menggunakan tangan di depan badan. Pada posisi ini, penulis menyimpulkan bahwa orang tunanetra yang menggunakannya mengalami berbagai kesulitan. Kesulitan pertama adalah membawa laptop dengan tangan. Mereka kesulitan untuk membawa laptop dengan tegak dan aman, karena mereka tidak dapat melihat posisi laptop tersebut dan lebih fokus kepada jalan yang akan

dilaluinya. Kesulitan selanjutnya adalah ketika kamera diletakkan di atas kepala. Berbeda dengan orang yang memiliki penglihatan normal, orang tunanetra tidak terbiasa melihat ke arah depan. Mereka lebih biasa berjalan dengan memiringkan kepalanya agar lebih memfokuskan pada pendengaran mereka. Sehingga peletakan kamera di atas kepala terhalang oleh gerakan kepala mereka yang terkadang mulai miring.

Posisi kedua adalah dengan menempatkan penahan laptop yang dipasang di depan badan pengguna, kemudian kamera diletakkan di depan laptop seperti pada Gambar 4.3 di sebelah kanan dan tengah. Dengan posisi ini, para tunanetra yang menggunakannya tidak kesulitan lagi memegangi laptop sehingga dapat bergerak lebih bebas. Lalu posisi kamera juga lebih rendah dari yang sebelumnya, dan tidak tergantung pada posisi kepala. Sehingga permasalahan terkait pergerakan kepala yang tidak tegak tidak jadi masalah di sini. Akan tetapi posisi kamera ini menjadi sedikit lebih maju terhadap penggunanya. Sehingga jangkauan kamera ini menjadi lebih sempit, dan objek-objek di sekitar pengguna menjadi tidak tertangkap oleh kamera.

Posisi ketiga adalah posisi dengan kamera tepat di depan dada pengguna. Posisi ini mengurangi masalah yang ada pada posisi kedua dalam hal jangkauan kamera. Kamera yang dimundurkan sedikit ini membuat sudut pandang menjadi

Gambar 4.3 Posisi peletakan perangkat keras terhadap pengguna tunanetra

lebih luas sehingga halangan yang lebih dekat dengan penggunanya lebih terdeteksi.

4.2 Percobaan Algoritma

Dalam pelaksanaan penelitian ini, ada dua algoritma yang dicobakan untuk mencapai tujuan dari penelitian ini. Tujuan utama dari penelitian ini adalah menavigasikan tuna netra untuk berjalan melalui suatu lingkungan yang tidak dikenalnya dengan baik. Sehingga dilakukan riset untuk menentukan algoritma yang cocok dalam mencapai tujuan tersebut.

4.2.1 Uji Coba Algoritma Pemetaan 3 Dimensi

Pada uji coba algoritma ini, pemetaan dilakukan dengan mengambil beberapa *frame* dari gambar kedalaman yang direkonstruksi menjadi bentuk *point cloud*. Pengambilan masing-masing *frame* dilakukan dengan kamera menghadap ke arah yang berbeda atau posisi kamera yang berbeda satu sama lain. Sehingga *point cloud* yang didapatkan berbeda juga.

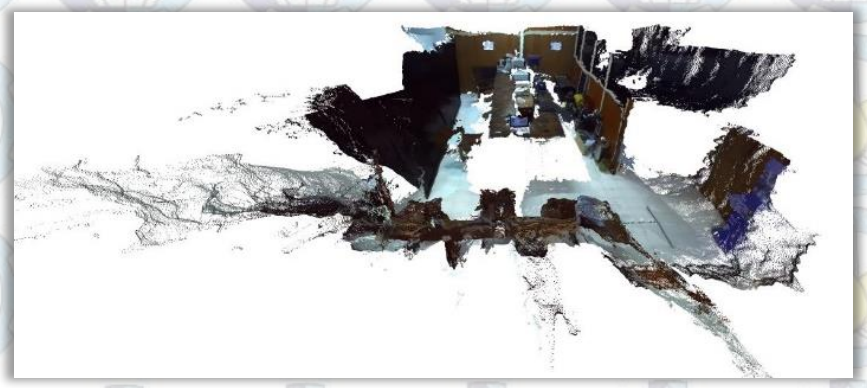
Gambar 4.4 (a) merupakan pemetaan dari 8 *frame* yang diambil di ruang B402 di kampus ITS pada tanggal 9 Maret 2019, kemudian Gambar 4.4 (b) adalah pemetaan dari ruang B403 di kampus ITS pada tanggal 9 Maret 2019, masing-masing menggunakan resolusi kamera 720p. Pada Gambar 4.4 (c), merupakan pemetaan ruang B402 di kampus ITS tanggal 15 Maret 2019 dengan resolusi 2K.

Tabel 4.3 Waktu yang Diperlukan Algoritma Pemetaan 3 Dimensi untuk Menghitung Satu Frame

***Frame* ke**	**Waktu tiap resolusi (detik)**			
	HD2K (2208 x 1242)	**HD1080 (1920 x 1080)**	**HD720 (1280 x 720)**	**VGA (672 x 376)**
1	6.898	5.226	2.317	0.783
2	6.961	5.641	2.41	0.753
3	6.642	5.329	2.371	0.677
4	6.568	5.507	2.958	0.693
5	6.721	5.543	2.799	0.792
Rata-rata	**6.758**	**5.4492**	**2.571**	**0.7396**

Gambar pemetaan dengan resolusi kamera 2K pada Gambar 4.4 (c) memiliki hasil gambar yang lebih tajam, dan posisi titik yang lebih menyerupai aslinya. Sedangkan gambar pemetaan dengan menggunakan resolusi 720p pada dinding dan bidang pembatas yang seharusnya lurus, terlihat lebih melengkung dan tidak menyerupai aslinya.

(a)



(b)

(c)

Gambar 4.4 Hasil pemetaan 3 dimensi menggunakan kamera stereo dengan resolusi (a) VGA, (b) 720p, dan (c) 2K

Akan tetapi algoritma ini tidak cocok untuk pengolahan data *real-time*. Karena proses ini memakan waktu yang banyak. Sesuai pada data waktu pemrosesan per-*frame* tiap resolusi yang ditampilkan di Tabel 4.3 menunjukkan bahwa untuk mengambil dan memproses 1 *frame* hingga proses konversi ke *point cloud* selesai dibutuhkan waktu 0.7 – 6.9 detik. Jika dihubungkan dengan teori *real-time* yang disebutkan pada BAB 2, maka hitungan waktu ini tidak cukup cepat untuk dikategorikan sebagai sistem *real-time*, karena sistem *real-time* membutuhkan waktu respons yang setidaknya lebih cepat dari respons manusia, yaitu kurang dari 0.25 detik. Maka dari itu kelanjutan dari algoritma ini dihentikan di proses pemetaan, karena hanya sampai proses ini saja sudah memakan waktu yang lebih besar dari waktu yang ditentukan.

4.2.2 Uji Coba Algoritma Pemetaan Gambar Kedalaman

Pada algoritma ini, gambar kedalaman atau *depth image* diproses langsung tanpa mengubahnya ke bentuk 3 dimensi terlebih dahulu. Hal pertama yang dicoba adalah dengan membuat gambar kedalaman dari gambar stereo. Gambar kedalaman ini diperoleh dengan menggunakan perintah pada Python di bawah ini.

```
zed.retrieve_image(image, sl.PyVIEW.PyVIEW_DEPTH)
```

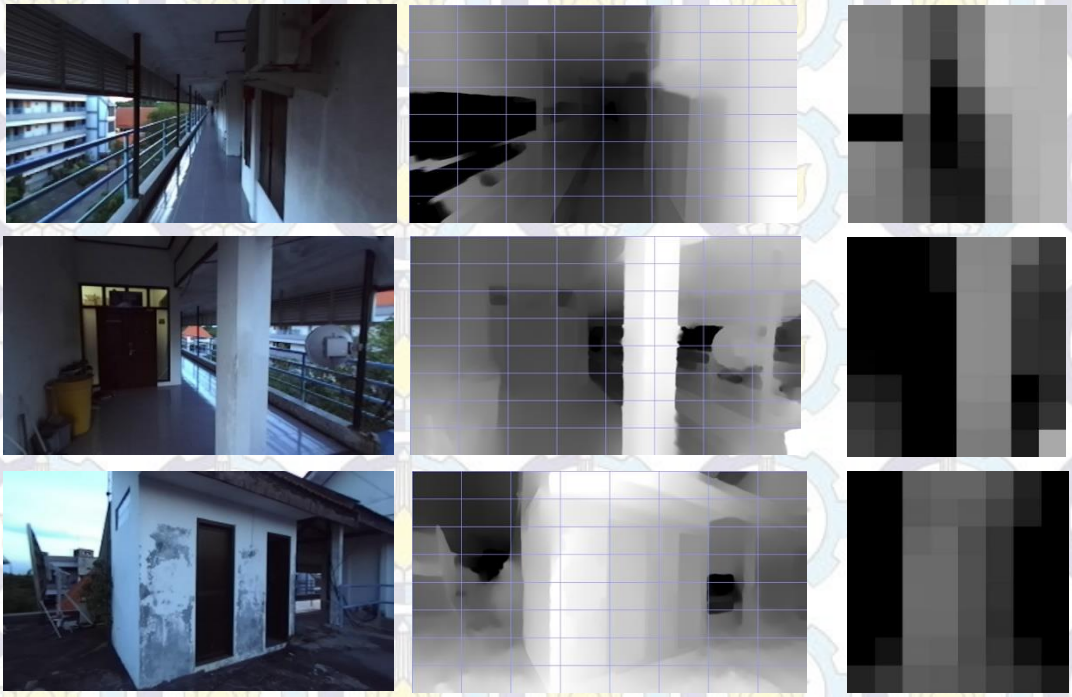
Berikut beberapa contoh gambar konstruksi gambar kedalaman yang didapat dari gambar stereo ditampilkan pada Gambar 4.5.

Tabel 4.4 Waktu yang Diperlukan Algoritma Gambar Kedalaman untuk Menghitung Satu Frame

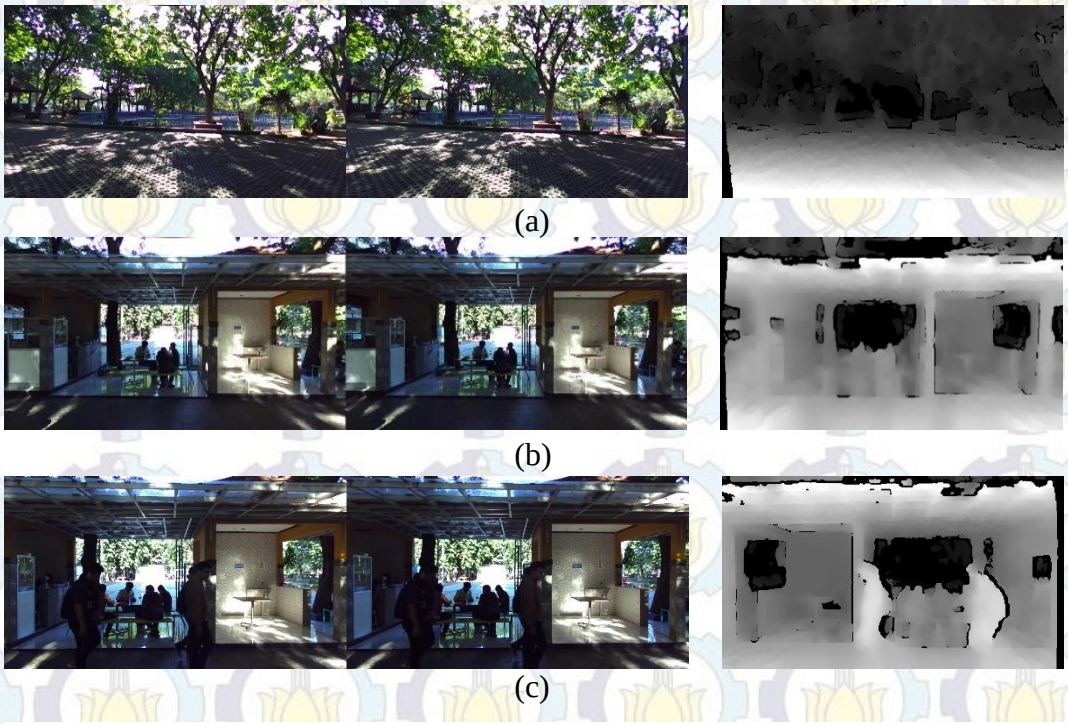
Frame ke	Waktu tiap resolusi (detik)			
	HD2K (2208 x 1242)	**HD1080 (1920 x 1080)**	**HD720 (1280 x 720)**	**VGA (672 x 376)**
1	0.68	0.579	0.434	0.110
2	0.612	0.557	0.433	0.104
3	0.585	0.591	0.431	0.101
4	0.582	0.591	0.424	0.103
5	0.593	0.573	0.426	0.104
Rata-rata	**0.6104**	**0.5782**	**0.4296**	**0.1044**

Setelah didapatkan bentuk gambar kedalaman, gambar tersebut disederhanakan menjadi matriks 8x8 seperti pada Gambar 4.5. Gambar tersebut dibagi menjadi 8 bagian secara horizontal dan 8 bagian secara vertikal, kemudian diambil nilai maksimum setiap bagiannya. Nilai maksimum setiap bagian tersebut yang menjadi nilai *grid* kecil yang berukuran 8x8. Dari gambar *grid* tersebut, hampir semua nilai *NAN*, *INFINITE*, dan *–INFINITE* hilang dikarenakan dalam 1 kotak hampir tidak mungkin seluruhnya bernilai tersebut. Sehingga nilai yang ditampilkan pada *grid* hampir keseluruhan merupakan nilai angka.

Hasil uji coba menggunakan algoritma ini menunjukkan perhitungan yang lebih cepat dari algoritma sebelumnya. Pada resolusi yang paling tinggi waktu yang diperlukan untuk menghitung satu *frame* hanya sekitar 0,6 detik. Sedangkan pada resolusi terendah dapat diperoleh dengan waktu sekitar 0,1 detik. Dengan waktu 0,1 maka sistem ini dapat dikategorikan sebagai sistem *real-time* karena lebih cepat dari respons manusia yang disebutkan pada BAB 2, yaitu 0.15 – 0.25 detik.



Gambar 4.5 Gambar kedalaman dan hasil konversi menjadi sederhana



(a)

(b)

(c)

Gambar 4.6 Konstruksi gambar kedalaman dari gambar stereo

4.3 Pemasangan Perangkat pada Tunanetra

Algoritma dari sistem yang dapat berjalan secara real-time diuji coba kepada tunanetra. Perancangan perangkat keras yang disebutkan pada BAB 3 dicobakan satu per satu. Pada awal perancangan adalah dengan meletakkan headset tepat di atas kepala seperti pemakaian headset pada umumnya. Kemudian kamera ZED diletakkan tepat di atas headset, dan komputer dipegang dengan tangan seperti pada Gambar 4.7. Seluruh komponen dihubungkan dengan kabel, dan program dijalankan. Ketika uji coba dijalankan ada permasalahan yang muncul, yaitu pemasangan kamera di kepala membuat posisi kamera tidak konstan menghadap ke depan. Penyebabnya adalah kepala yang dapat fleksibel bergerak bebas. Permasalahan tersebut semakin besar ketika digunakan oleh tunanetra, karena mereka tidak dapat menentukan bagaimana posisi kepala yang tegak disebabkan mereka tidak dapat melihat referensi kepala tegak dari orang lain. Pendeteksian objek pun bermasalah, karena objek yang pendek seperti pagar sering tidak terdeteksi karena sudut pandang dari kamera itu tidak sampai cukup rendah setinggi pagar tersebut. Permasalahan lain adalah mereka sedikit keberatan dengan membawa laptop di tangannya, karena tidak dapat bergerak secara bebas. Sehingga

Gambar 4.7 Peletakan kamera di atas pengguna

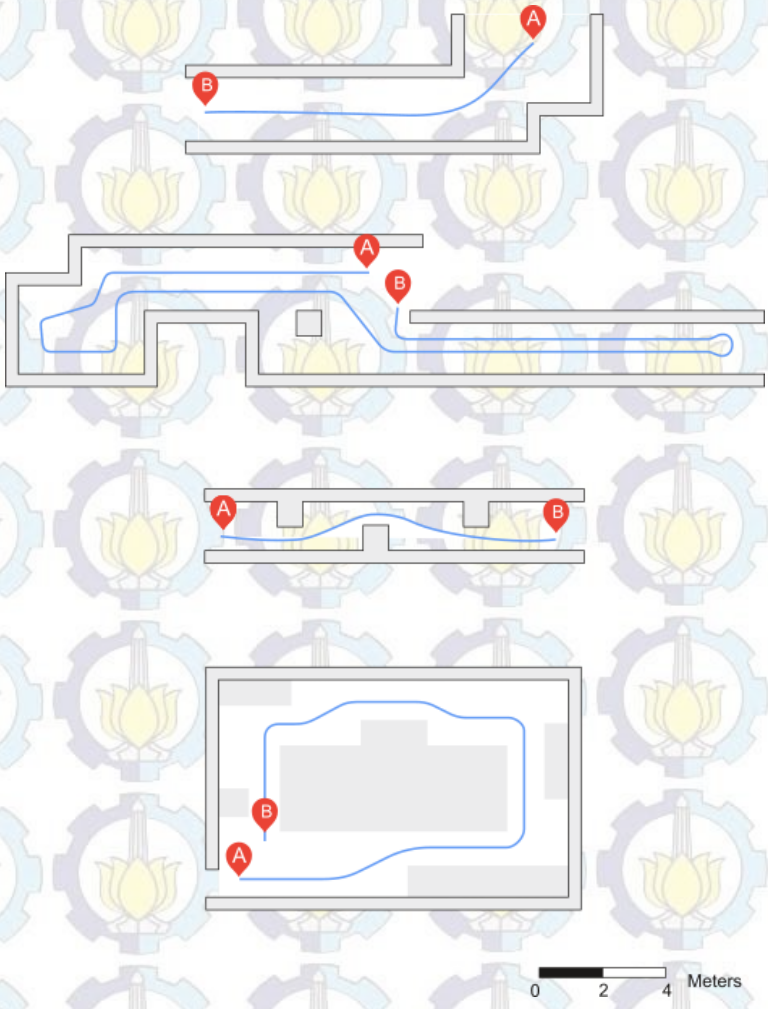
peletakan kamera dan laptop akan dipindahkan ke tempat yang lebih memudahkan pengguna.

Selanjutnya dengan mengubah peletakan laptop dengan penyangga di depan badan pengguna, dan peletakan kamera tepat di depan laptop seperti Gambar 4.8. Posisi ini lebih stabil karena posisi kamera statis, tidak bergerak ketika ada pergerakan di kepala. Dengan ini juga jarak dan sudut pandang kamera ke bawah dan ke atas sama besarnya sehingga pengguna dapat menghindari halangan yang rendah dengan baik.

Gambar 4.8 Peletakan kamera di depan pengguna dengan jeda laptop

4.4 Percobaan pada Jalur yang Ditentukan

Sistem yang sudah selesai secara keseluruhan diuji coba pada tunanetra. Para tunanetra dipakaikan sistem pemetaan halangan kami yang dipasang. Keberhasilan uji coba ini diukur dengan cara orang tunanetra tersebut diarahkan untuk berjalan pada rintangan, kemudian dihitung jumlah tabrakan selama di rintangan tersebut. Ada 4 bentuk rintangan yang digunakan untuk uji coba seperti dalam Gambar 4.9, dengan kesulitan yang berbeda-beda. Rintangan level 1 adalah yang termudah, dan rintangan level 4 yang tersulit. Masing-masing rintangan memiliki kemungkinan tabrakan yang bervariasi. Jumlah kemungkinan tabrakan ini dihitung ketika ada belokan. Dalam satu belokan atau satu halangan objek diperkirakan jumlah tabrakan yang memungkinkan adalah 1 hingga 3 tabrakan.



Gambar 4.9 Rintangan yang digunakan, level 1 (atas) sampai level 4 (bawah).

Hasil percobaan pada 5 orang tunanetra dan jumlah tabrakannya ditampilkan pada Tabel 4.5.

Dari hasil uji coba yang ditampilkan pada Tabel 4.5, didapatkan hasil bahwa masing-masing pengguna yang memakai sistem ini masih ada yang menabrak sesuatu dengan jumlah yang berbeda tiap level. Dari hasil tersebut, semakin tinggi levelnya maka semakin banyak tabrakan yang mungkin terjadi.

Tabel 4.5 Uji Coba ke Pengguna Tuna Netra, dan Jumlah Tabrakan pada Tiap Level

No.	Pengguna	Jalur rintangan	Jumlah tabrakan yang mungkin	Jumlah tabrakan yang terjadi
1.	Pengguna 1	Level 1	4	0
		Level 2	10	0
		Level 3	8	1
		Level 4	16	3
2.	Pengguna 2	Level 1	4	0
		Level 2	10	0
		Level 3	8	0
		Level 4	16	2
3.	Pengguna 3	Level 1	4	0
		Level 2	10	0
		Level 3	8	3
		Level 4	16	3
4.	Pengguna 4	Level 1	4	0
		Level 2	10	1
		Level 3	8	4
		Level 4	16	8
5.	Pengguna 5	Level 1	4	0
		Level 2	10	1
		Level 3	8	1
		Level 4	16	5
Total			**190**	**32**

Kemudian jumlah tabrakan yang terjadi setiap orang berbeda-beda, ada yang dapat menghindari lebih dari 80% halangan, tetapi ada juga yang dalam level tertentu hanya dapat menghindari 50% halangan. Dari data ini maka dapat disimpulkan mereka memiliki pemahaman dan kesesuaian yang berbeda-beda terhadap sistem ini. Akan tetapi secara keseluruhan mereka dapat menghindari halangan dengan baik, yaitu dengan rata-rata tabrakan yang dapat dihindari sebanyak 32 dari total 190 yang mungkin terjadi. Artinya mereka dapat berjalan secara akurat sebesar 83,16%.

BAB 5
PENUTUP

5.1 Kesimpulan

Dari percobaan yang sudah dilakukan pada tesis ini, maka dapat disimpulkan bahwa kamera stereo ZED memiliki sudut pandang horizontal yang berbeda-beda tiap resolusi. Sudut pandangnya adalah 75,55° pada resolusi 2K, 66,44° pada 1080p, dan 85,53° pada 720p. Kamera ZED juga dapat memetakan dari gambar stereo menjadi bentuk 3 dimensi, akan tetapi dengan waktu yang bervariasi tiap resolusi, yaitu 0,7 hingga 6,9 detik setiap *frame*. Proses pemetaan ini tidak dapat dikategorikan sebagai sistem yang *real-time*, karena untuk mengolah satu *frame* dibutuhkan waktu lebih dari 0,25 detik. Sedangkan uji coba algoritma dengan memproses *depth image* dapat diselesaikan lebih cepat, yaitu 0,1 hingga 0,6 detik yang bervariasi tiap resolusi, maka dapat dikatakan sebagai *real-time*.

Pemasangan perangkat pada tunanetra ketika kamera ditaruh di atas kepala mengalami masalah, yaitu pergerakan kepala mereka yang tidak teratur sehingga mengganggu proses deteksi kamera. Pemasangan kamera yang diletakkan di depan pengguna tunanetra menunjukkan hasil yang lebih baik. Algoritma yang memproses *depth image* pada saat diuji coba ke tunanetra menunjukkan bahwa mereka dapat menghindari rata-rata 83.16% halangan di depannya. Akan tetapi pemahaman mereka terhadap perangkat ini tidak semuanya baik, ada yang kurang memahami terbukti dengan mereka tidak dapat menghindari halangan dengan baik yaitu di bawah 75%. Hal ini menunjukkan bahwa tingkat pemahaman mereka terhadap sistem ini berbeda-beda, akan tetapi 4 dari 5 orang yang diuji coba dapat menerimanya dengan baik dibuktikan dengan jumlah tabrakan yang sedikit.

5.2 Saran

Penelitian ini masih jauh dari kata sempurna, sehingga perlu adanya pengembangan lebih lanjut agar lebih memudahkan orang tunanetra yang menggunakannya. Dengan meningkatkan akurasi pendeteksian, menginformasikan suara yang lebih mudah dipahami, menambahkan fitur yang dapat mendeteksi lubang, yang diharapkan nantinya orang tunanetra akan semakin lebih percaya diri

dalam berjalan. Serta dengan ditambahkannya pergerakan sistem yang lebih halus, dan penyederhanaan laptop yang digantikan Mini PC atau *Single Board Computer* yang mampu melaksanakan sistem ini secara keseluruhan akan semakin memudahkan dan tidak memberatkan penggunanya.

DAFTAR PUSTAKA

[1] Vision Australia, "Difficulties with vision loss," Vision Australia. Blindness and low vision services, [Online]. Available: https://www.visionaustralia.org/information/newly-diagnosed/difficulties-with-vision-loss. [Diakses 5 Mei 2018].

[2] Population Reference Bureau, "2017 World Population Datasheet," Agustus 2017. [Online]. Available: https://assets.prb.org/pdf17/2017_World_Population.pdf. [Diakses 11 Mei 2018].

[3] Centers for Disease Control and Prevention, "Blindness and Vision Impairment," 16 September 2017. [Online]. Available: https://www.cdc.gov/healthcommunication/ToolsTemplates/EntertainmentEd/Tips/Blindness.html. [Diakses 6 Juni 2019].

[4] A. T. Noman, M. A. Mahmud, H. Rashid dan S. M. Saifur Rahman, "Design and Implementation of Microcontroller Based Assistive Robot for Person with Blind Autism and Visual Impairment," dalam *International Conference of Computer and Information Technology (ICCIT)*, Chittagong, Bangladesh, 2017.

[5] M. H. Mahmud, R. Saha dan S. Islam, "Smart Walking Stick - An Electronic Approach Assist Visually Disabled Person," *International Journal of Scientific & Engineering Research*, vol. 4, no. 10, 2013.

[6] J. Borenstein dan I. Ulrich, "The GuideCane - A Computerized Travel Aid for the Active Guidance of Blind Pedestrians," dalam *International Conference on Robotics and Automation, Albuquerque*, New Mexico, 1997.

[7] C. Stoll, R. Palluel-Germain, V. Fristot, D. Pellerin, D. Alleysson dan C. Graff, "Navigating from a Depth Image Converted into Sound," *Hindawi*, vol. 2015, no. 10.1155/2015/543492, p. 9, 2015.

[8] StereoLabs, "ZED Camera Now Available for Order," 21 Mei 2015. [Online]. Available: https://www.stereolabs.com/blog/zed-camera-now-available-for-order/. [Diakses 28 Mei 2019].

[9] L. E. Ortiz, L. M. G. Gonçalves dan E. V. Cabrera, "Depth Data Error Modeling of the ZED 3D Vision Sensor from Stereolabs," *Electronic Letters on Computer Vision and Image Analysis*, vol. 17, no. 1, pp. 1-15, 2018.

[10] M. S. H. Achmad, W. S. Findari, N. Q. Ann dan M. R. Daud, "Stereo Camera – based 3D Object Reconstruction Utilizing Semi-Global Matching Algorithm," dalam *2nd International Conference on Science and Technology-Computer* (ICST), Yogyakarta, Indonesia, 2016.

[11] T. Gutpa dan H. Li, "Indoor Mapping for Smart Cities - an Affordable Approach," dalam *International Conference on Indoor Positioning and Indoor Navigation*, Sapporo, Japan, 2017.

[12] E. Kirsten, L. C. Inocencio, M. R. Veronez, L. G. da Silveira, F. Bordin dan F. P. Marsin, "3D Data Aquisition using Stereo Camera," dalam *IGARSS*, Valencia, 2018.

[13] Wikipedia, "Stereo camera," 2018. [Online]. [Diakses 8 Mei 2019].

[14] Stereolabs, "Meet the ZED Stereo Camera," Stereolabs, 2018. [Online]. Available: https://www.stereolabs.com/zed/. [Diakses 20 Mei 2018].

[15] NVIDIA Corporation, "CUDA GPUs | NVIDIA Developer," NVIDIA Corporation, 2018. [Online]. Available: https://developer.nvidia.com/cuda-gpus. [Diakses 20 Mei 2018].

[16] H. Shin dan K. Sohn, "Real-time Depth Range Estimation and Its Application to Mobile Stereo Camera," dalam *Consumer Communications and Networking Conference*, Las Vegas, USA, 2012.

[17] G. F. Marshall, Handbook of Optical and aser Scanning, Marcel Dekker, 2004.

[18] M. Dassot, T. Constant dan M. Fournier, "The use of terrestrial LiDAR technology in forest science: application fields, benefits and challenges," *Annals of forest science*, vol. 5, no. 959-974, p. 68, 2011.

[19] Wikipedia, "Kinect," [Online]. Available: https://en.wikipedia.org/wiki/ Kinect. [Diakses 10 July 2019].

[20] D. S. O. Wasenm¨uller, "Comparison of Kinect v1 and v2 Depth Images in Terms of Accuracy and Precision," *Asian Conference on Computer Vision* (ACCV 2016), vol. 10117, pp. 34-45, 2017.

[21] TEMICS, "Digital Image Processing, Modelling and Communication," *INRIA, Informatics mathematics*, Rennes, 2011.

[22] D. Holz, A. E. Ichim, F. Tombari, R. B. Rusu dan S. Behnke, "Registration with the Point Cloud Library: A Modular Framework for Aligning in 3-D," *IEEE Robotics Automation*, vol. 22, no. 4, pp. 110-124, 2015.

[23] K. Barber, "Navigation," The Canadian Oxford dictionary, 2019.

[24] A. D. King, "Intertial Navigation - Forty Years of Evolution," *GEC Review*, vol. 13, no. 3, 1998.

[25] M. Ibrahim dan O. Moselhi, "IMU-Based Indoor Localization for Construction Application," dalam *32nd International Symposium on Automation and Robotics in Construction* (ISARC), Oulu, Finland, 2015.

[26] C. Randell, C. Djiallis dan H. Muller, "Personal Position Measurement using Dead Reckoning," dalam *7th International Symposium on Wearable Computers*, New York, 2003.

[27] Backyard Brains, "Experiment: How Fast Your Brain Reacts To Stimuli," [Online]. Available: https://backyardbrains.com/experiments/reactiontime. [Diakses 15 Juli 2019].

[28] Stereolabs, "Getting Started - Python Development," Stereolabs, 2018. [Online]. Available: https://www.stereolabs.com/docs/getting-started/python-development/#getting-started. [Diakses 15 Juni 2019].

[29] Stereo Labs, "Depth Sensing - Advanced Settings," 2018. [Online]. Available: https://www.stereolabs.com/docs/depth-sensing/advanced-settings/. [Diakses 13 May 2019].

[30] E. Imhof, Cartographic Relief Presentation, Esri Pr. ISBN 978-1-58948-026-1., 2007.

Halaman ini sengaja dikosongkan

LAMPIRAN GAMBAR

Lampiran Gambar 1. Stereo kamera ZED

Lampiran Gambar 2. Stereo dan gambar kedalamannya (*depth image*):

Lampiran Gambar 3. Bentuk perangkat penahan laptop dan kamera

Lampiran Gambar 4. Uji coba ke orang yang ditutup matanya

Lampiran Gambar 5. Uji coba ke tunanetra

Halaman ini sengaja dikosongkan

LAMPIRAN PROGRAM

Lampiran program 1. Algoritma pemetaan

```
import cv2
import numpy as np
import pyzed.camera as zcam
import pyzed.defines as sl
import pyzed.types as tp
import pyzed.core as core
import open3d as d3
import time

class Mapping3D(object):
    # Create a PyZEDCamera object
    zed = zcam.PyZEDCamera()

    def __init__(self):
        # Buat objek PyInitParameters dan konfigurasi parameters
        self.params = zcam.PyInitParameters()
        self.params.depth_mode = sl.PyDEPTH_MODE.PyDEPTH_MODE_ULTRA
        self.params.coordinate_units = sl.PyUNIT.PyUNIT_METER
        self.params.camera_resolution = sl.PyRESOLUTION.PyRESOLUTION_VGA
        self.params.coordinate_system = sl.PyCOORDINATE_SYSTEM.
                                PyCOORDINATE_SYSTEM_RIGHT_HANDED_Y_UP
        self.params.coordinate_units = sl.PyUNIT.PyUNIT_METER

        # Open the camera
        err = self.zed.open(self.params)
        if err != tp.PyERROR_CODE.PySUCCESS:
            exit(1)

        # Create and set PyRuntimeParameters after opening the camera
        self.runtime_parameters = zcam.PyRuntimeParameters()
        self.runtime_parameters.sensing_mode = sl.PySENSING_MODE.
                                               PySENSING_MODE_STANDARD
        self.ambil = sl.PyVIEW.PyVIEW_DEPTH

        # Enable positional tracking with default parameter
        py_transform = core.PyTransform()
        self.tracking_parameters =
                            zcam.PyTrackingParameters(init_pos=py_transform)
        err = self.zed.enable_tracking(self.tracking_parameters)
        if err != tp.PyERROR_CODE.PySUCCESS:
            exit(1)

        self.mat = core.PyMat()
        self.point_cloud = core.PyMat()

        self.zed_pose = zcam.PyPose()

    def capture(self, mode='DEPTH'):
        self.errcode = self.zed.grab(self.runtime_parameters)
        if self.errcode!=tp.PyERROR_CODE.PySUCCESS:
            print(self.errcode)
        if mode == 'DEPTH' and self.errcode==tp.PyERROR_CODE.PySUCCESS:
            self.zed.retrieve_image(self.mat, sl.PyVIEW.PyVIEW_DEPTH)
            self.gambar = cv2.cvtColor(self.mat.get_data(),
                                       cv2.COLOR_BGRA2BGR)
```

```
            self.gambar = self.gambar.astype(np.uint8)
            return self.gambar
        elif mode == 'COLOR' and self.errcode==tp.PyERROR_CODE.PySUCCESS:
            self.zed.retrieve_image(self.mat, sl.PyVIEW.PyVIEW_LEFT)
            self.gambar = cv2.cvtColor(self.mat.get_data(),
                                       cv2.COLOR_BGRA2BGR)
            self.gambar = self.gambar.astype(np.uint8)
            return self.gambar
        else:
            print('gagal')
            return 0

    def measure(self):
        if self.errcode==tp.PyERROR_CODE.PySUCCESS:
            self.zed.retrieve_measure(self.point_cloud,
                                      sl.PyMEASURE.PyMEASURE_XYZ)
            return (self.point_cloud.get_data())
        else:
            print('gagal')

    def getMatTransform(self):
        self.zed.get_position(self.zed_pose, sl.PyREFERENCE_FRAME.
                              PyREFERENCE_FRAME_WORLD)
        matrx = core.PyTransform()
        self.zed_pose.pose_data(matrx)
        return matrx

mapping = Mapping3D()
mapping.zed.set_depth_max_range_value(7)

pcd_points_array = np.array([])
pcd_colors_array = np.array([])
transformasi = []
pcds = d3.PointCloud()
cnt = 0

while(True):
    wkt = time.time()
    color = mapping.capture('COLOR')
    depth = mapping.capture('DEPTH')
    cv2.imshow('Stereo Left', cv2.resize(depth, (720,480)))

    xyz = mapping.measure()

    key = cv2.waitKey(1)
    if key >= 27:
        cv2.destroyAllWindows()
        cnt += 1

        rot = core.PyTransform()
        mapping.zed.get_position(mapping.zed_pose, sl.PyREFERENCE_FRAME.
                                 PyREFERENCE_FRAME_WORLD)
        mapping.zed_pose.pose_data(rot)

        W,H,L = np.shape(xyz)
        rgb = cv2.cvtColor(color, cv2.COLOR_BGR2RGBA)
        rgb = np.array(rgb[np.logical_not(np.isnan(xyz))])
        xyz = np.array(xyz[np.logical_not(np.isnan(xyz))])
        rgb = np.array(rgb[np.logical_not(np.isinf(xyz))])
        xyz = np.array(xyz[np.logical_not(np.isinf(xyz))])
        rgb = np.reshape(rgb,[len(rgb)//L, L])
        xyz = np.reshape(xyz,[len(xyz)//L, L])
        rgb = rgb[:,0:-1].astype('float32')/255
```

```
114             xyz = xyz[:,0:-1]
115
116             pcds.clear()
117             pcds.points = d3.Vector3dVector(xyz)
118             pcds.colors = d3.Vector3dVector(rgb)
119             pcds.transform(rot[:])
120
121             pcd_points_array = np.append(pcd_points_array,
122                                          np.asarray(pcds.points))
123             pcd_colors_array = np.append(pcd_colors_array,
124                                          np.asarray(pcds.colors))
125             print(np.round(time.time() - wkt, 3), "detik")
126             if key == 27:
127                 break
128             if cnt > 20:
129                 break
130
131 mapping.zed.close()
132
133 L = len(pcd_points_array)
134 pcd_points_array = pcd_points_array.reshape([L//3,3])
135 pcd_colors_array = pcd_colors_array.reshape([L//3,3])
136
137 pcds.clear()
138 pcds.points = d3.Vector3dVector(pcd_points_array)
139 pcds.colors = d3.Vector3dVector(pcd_colors_array)
140
141 nama = 'Mapping\\Pointcloud\\Around_'+str(round(time.time()))
142        + '_'+str(cnt)+'-titik.pcd'
143
144 if d3.write_point_cloud(nama, pcds):
145     print ("Pointcloud disimpan di",nama)
146
147 d3.draw_geometries([pcds])
```

Lampiran program 2. Algoritma pemrosesan *depth image*

```
import cv2
import numpy as np
import pyzed.camera as zcam
import pyzed.defines as sl
import pyzed.types as tp
import pyzed.core as core
import open3d as d3
import audiolib
from time import time

class Mapping3D(object):
    # Create a PyZEDCamera object
    zed = zcam.PyZEDCamera()

    def __init__(self):
        # Buat objek PyInitParameters dan konfigurasi parameters
        self.params = zcam.PyInitParameters()
        self.params.depth_mode = sl.PyDEPTH_MODE.PyDEPTH_MODE_ULTRA
        self.params.coordinate_units = sl.PyUNIT.PyUNIT_MILLIMETER
        self.params.camera_resolution = sl.PyRESOLUTION.PyRESOLUTION_VGA

        # Open the camera
        err = self.zed.open(self.params)
        if err != tp.PyERROR_CODE.PySUCCESS:
            exit(1)

        # Create and set PyRuntimeParameters after opening the camera
        self.runtime_parameters = zcam.PyRuntimeParameters()
        self.runtime_parameters.sensing_mode =
                                    sl.PySENSING_MODE.PySENSING_MODE_FILL
        self.ambil = sl.PyVIEW.PyVIEW_DEPTH

        self.mat = core.PyMat()
        self.point_cloud = core.PyMat()

    def capture(self, mode='DEPTH'):
        self.errcode = self.zed.grab(self.runtime_parameters)
        if self.errcode!=tp.PyERROR_CODE.PySUCCESS:
            print(self.errcode)
        if mode == 'DEPTH' and self.errcode==tp.PyERROR_CODE.PySUCCESS:
            self.zed.retrieve_image(self.mat, sl.PyVIEW.PyVIEW_DEPTH)
            self.gambar = cv2.cvtColor(self.mat.get_data(),
                                       cv2.COLOR_BGRA2BGR)
            self.gambar = self.gambar.astype(np.uint8)
            return self.gambar
        elif mode == 'COLOR' and self.errcode==tp.PyERROR_CODE.PySUCCESS:
            self.zed.retrieve_image(self.mat, sl.PyVIEW.PyVIEW_LEFT)
            self.gambar = cv2.cvtColor(self.mat.get_data(),
                                       cv2.COLOR_BGRA2BGR)
            self.gambar = self.gambar.astype(np.uint8)
            return self.gambar
        else:
            print('gagal')
            return 0

    def measure(self):
        self.errcode = self.zed.grab(self.runtime_parameters)
        if self.errcode==tp.PyERROR_CODE.PySUCCESS:
            self.zed.retrieve_measure(self.point_cloud,
                                      sl.PyMEASURE.PyMEASURE_XYZ)
```

```
            return (self.point_cloud.get_data())
        else:
            print('gagal')


mapping = Mapping3D()
maxmeter = 8
mapping.zed.set_depth_max_range_value(maxmeter)
jml_grid = [8,8]
res_tmpl = (500,500)
dist_const = [300, 3000]
fps = 0; wkt = 0
wimg = mapping.capture()
min_dist = []
key = ''

def mapp(x, in_min, in_max, out_min, out_max):
    return (x-in_min) * (out_max-out_min) / (in_max-in_min) + out_min

def click(event, x, y, flags, param):
    global min_dist
    if event == cv2.EVENT_LBUTTONDOWN:
        jr = min_dist.reshape(jml_grid)
        print("x,y:",x,y)
        koord = [x*8//500,y*8//500]
        print("Koord:",koord)
        print(min_dist[koord[1]*8 + koord[0]])

def playAudio(koordinat):
    kirikanan = np.argmin(koordinat) % 8
    atasbawah = np.argmin(koordinat) // 8
    distance = np.min(koordinat)
    distance = 1-np.clip(mapp(distance, 600, 3000, 0, 1), 0, 1)

    audiolib.play(kirikanan, atasbawah, distance)

cv2.namedWindow("Grid")
cv2.setMouseCallback("Grid", click)
n = 0
fps = []
wkt = time()

while(True):
    n+=1
    fps.append(time()-wkt)
    wkt = time()

    meas = mapping.measure()
    dist = np.sqrt(meas[:,:,0]*meas[:,:,0] + meas[:,:,1]*meas[:,:,1] +
                   meas[:,:,2]*meas[:,:,2])

    h,w,d = meas.shape

    le,ti = w//jml_grid[0], h//jml_grid[1]

    grid_dist = []
    min_dist = []

    for i in range(jml_grid[1]):
        for j in range(jml_grid[0]):
            grid_dist.append(dist[i*ti:(1+i)*ti, j*le:(1+j)*le])
            min_dist.append(np.nanmin(grid_dist[-1]))

```

```
    min_dist = [40000.0 if f==np.inf else f for f in min_dist]
    min_dist = np.array(min_dist)
    dist = np.clip(255 - (min_dist-0)*(255) / (4000) + 0,
                   0, 255) # 0-4 meter
    dist = dist.reshape(jml_grid).astype(np.uint8)

    ''' Saving '''
    if n%9 == 0:
        np.savetxt("Dataset 2\\" + str(wkt) + "_distance.txt",min_dist)
        cv2.imwrite("Dataset 2\\" + str(wkt) + "_image.jpg", dist)

    ''' Showing '''
    dist_tampil = cv2.resize(dist, res_tmpl, interpolation = False)
    cv2.imshow("Grid",dist_tampil)
    cv2.imshow("depth", mapping.capture())

    playAudio(min_dist)

    key = cv2.waitKey(1)

    if key == ord('q'):
        cv2.destroyAllWindows()
        mapping.zed.close()
        print("Waktu:",np.average(fps))
        break

    elif key == ord('c'):
        namadepth = "Dataset\\"+str(int(time()))+"-DEPTH.jpg"
        namacolor = "Dataset\\"+str(int(time()))+"-COLOR.jpg"
        namagrid = "Dataset\\"+str(int(time()))+"-GRID.jpg"
        cv2.imwrite(namadepth, mapping.capture())
        cv2.imwrite(namacolor, mapping.capture(mode='COLOR'))
        cv2.imwrite(namagrid, dist_tampil)
        print("image saved to",namadepth)
```

Lampiran program 3. Program pengolahan suara

```
import pygame
import time


g = [0, 0.1, 0.2, 0.5, 0.5, 0.8, 0.9, 1.0]
wkt_terakhir = time.time()
pygame.mixer.init()

sounds = []


for i in range(1,9):
    file = "Sound data\\beep\\audacity\\L7"+str(9-i)+".wav"
    sound = pygame.mixer.Sound(file)
    sounds.append(sound)


channel0 = pygame.mixer.Channel(0)


def play(hori, verti, jarak):
    global wkt_terakhir
    if time.time() - wkt_terakhir > 0.8:
        wkt_terakhir = time.time()
        volL, volR = 1.0 - g[hori], g[hori]
        volL = volL * jarak * jarak * jarak
        volR = volR * jarak * jarak * jarak
        channel0.set_volume(volL, volR)
        channel0.play(sounds[verti])

```

BIOGRAFI PENULIS

Ichsan Pratama Adi dilahirkan di Klaten pada tanggal 02 Mei 1994. Sebagai anak pertama dari tiga bersaudara, penulis mengawali kegiatan pendidikan formal di TK Yayasan Pendidikan Jayawijaya di Kuala Kencana, Mimika, Papua, kemudian SD Yayasan Pendidikan Jayawijaya, yang kemudian dilanjutkan di MTs Pondok Pesantren Modern Islam Assalaam di Surakarta, SMA Negeri 1 Klaten dan pada tahun 2012 penulis diterima sebagai Mahasiswa S1 di jurusan Teknik Elektro ITS. Selama menjalani perkuliahan Sarjana, penulis aktif dalam kepanitiaan baik itu di lingkungan jurusan maupun di lingkungan institut. Selain itu, penulis juga turut berpartisipasi sebagai asisten praktikum di bidang studi elektronika. Lulus bulan Maret tahun 2017 membawa gelar Sarjana Teknik dari ITS, kemudian melanjutkan pendidikan Magister di Teknik Elektro ITS. Selama menempuh pendidikan Magister, penulis aktif mengikuti berbagai seminar, dan menulis *paper* untuk publikasi.

Email: ichsan.application@gmail.com