**TUGAS AKHIR – TE 141599**

# SEGMENTASI MODEL PRIMITIF TIGA DIMENSI BERBASIS *POINT CLOUD* MENGGUNAKAN *RANDOM SAMPLE CONSENSUS*

Aditya Rifa Utama
NRP 2211100066

Dosen Pembimbing
Dr. Eko Mulyanto Yuniarno, ST., MT.
Christyowidiasmoro, ST., MT.

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

Institut
Teknologi
Sepuluh Nopember

**TUGAS AKHIR – TE 141599**

# SEGMENTASI MODEL PRIMITIF TIGA DIMENSI BERBASIS *POINT CLOUD* MENGGUNAKAN *RANDOM SAMPLE CONSENSUS*

Aditya Rifa Utama
NRP 2211100066

Dosen Pembimbing
Dr. Eko Mulyanto Yuniarno, ST., MT.
Christyowidiasmoro, ST., MT.

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


**FINAL PROJECT – TE 141599**

# *THREE-DIMENSIONAL PRIMITIVE MODEL SEGMENTATION BASED ON POINT CLOUD USING RANDOM SAMPLE CONSENSUS*

Aditya Rifa Utama
NRP 2211100066

Advisors
Dr. Eko Mulyanto Yuniarno, ST., MT.
Christyowidiasmoro, ST., MT.

Department of Electrical Engineering
Faculty of Industrial Technology
Institut Teknologi Sepuluh Nopember
Surabaya 2015

**SEGMENTASI MODEL PRIMITIF TIGA DIMENSI BERBASIS *POINT CLOUD* MENGGUNAKAN *RANDOM SAMPLE CONSENSUS***

**TUGAS AKHIR**

**Diajukan untuk Memenuhi Sebagian Persyaratan**
**Untuk Memeperoleh Gelar sarjana Teknik**
**Pada**
**Bidang Studi Teknik Komputer dan Telematika**
**Jurusan Teknik Elektro**
**Institut Teknologi Sepuluh Nopember**

**Menyetujui:**

| Dosen Pembimbing I | Dosen Pembimbing II |
|---|---|
| Dr. Eko Mulyanto Y., ST., MT. | Christyowidiasmoro, ST., MT. |
| NIP. 196806011995121009 | NIP. 198301272009121004 |

**SURABAYA**
**JULI, 2015**

# ABSTRAK

Nama Mahasiswa : Aditya Rifa Utama
Judul Tugas Akhir : Segmentasi Model Primitif Tiga Dimensi Berbasis *Point Cloud* menggunakan *Random Sample Consensus*
Pembimbing : 1. Dr. Eko Mulyanto Yuniarno, ST., MT.
2. Christyowidiasmoro, ST., MT.

Permukaan dari suatu objek nyata dapat direpresentasikan kedalam data *point cloud* dengan setiap *point* memiliki komponen *x*, *y*, dan *z*. Data *point cloud* dimanfaatkan dalam mengembangkan teknologi *augmented reality* yang lebih interaktif. Dalam penelitian *augmented reality* sebelumnya, data *point cloud* yang diproses belum dipisahkan antar model primitif. Pemisahan antar model diperlukan dalam proses identifikasi objek atau pengolahan *point cloud* lebih lanjut. Tujuan dari tugas akhir ini adalah melakukan segmentasi model primitif tiga dimensi dari data *point cloud*, hasil tangkapan kamera Kinect, dengan menggunakan metode *Random Sample Consensus*. Model primitif yang disegmentasi adalah model *plane*, *sphere*, dan *cylinder*. Dari hasil pengujian, *point cloud* dapat dipisahkan menjadi data *inliers* yaitu data yang sesuai model berdasarkan nilai *threshold*, dan data *outliers* atau selain data *inliers*. Pada segmentasi tunggal bidang *plane* dengan masukan *multiframe* data *point cloud* didapatkan *frame persecond* sebesar 0,5 hingga 1,1 pada visualisasi sinkronous, sedangkan pada visualisasi asinkronous dihasilkan 0,7 hingga 1,9 *fps*. Sedangkan pada multisegmentasi dihasilkan nilai *fps* yang lebih kecil.

Kata kunci : Model primitif, *Point cloud*, *Random Sample Consensus*, Segmentasi

# *ABSTRACT*

*Name* : Aditya Rifa Utama
*Title* : *Three-Dimensional Primitive Model Segmentation Based On Point Cloud Using Random Sample Consensus*
*Advisors* : 1. Dr. Eko Mulyanto Yuniarno, ST., MT.
2. Christyowidiasmoro, ST., MT.

*The surface of real object can be represented in point cloud data with each point containing components x, y, z. Point cloud data is used to make augmented reality more interactive. In latest research of augmented reality, point cloud has not yet been separated from each model to another when it was processed. Separating each object is required in order to identify an object or more complex process. The purpose of this final project is to do three-dimensional primitive model segmentation from point cloud, in which the point cloud comes from Kinect, using Random Sample Consensus. Plane, sphere and cylinder are three-dimensional primitive model that is segmented. From the results, point cloud data can be separated into inliers data or represented data of primitive model based on threshold and outliers. In single-segmentation of plane with multi-frame point cloud input, synchronous visualization generates 0.5-1.1 frame per second. Whereas in asynchronous visualization generates 0.7-1.9 fps. On multi-segmentation process, the fps amounts less than single-segmentation.*

*Keywords: Primitive model, Point cloud, Random Sample Consensus, Segmentation*

# KATA PENGANTAR

Puji dan Syukur kehadirat Tuhan YME atas segala limpahan berkah, serta rahmat-Nya, penulis dapat menyelesaikan tugas akhir ini dengan judul : ***Segmentasi Model Primitif Tiga Dimensi Berbasis Point Cloud Menggunakan Random Sample Consensus***.

Tugas akhir ini disusun dalam rangka pemenuhan bidang riset di Jurusan Teknik Elektro ITS, Bidang Studi Teknik Komputer dan Telematika, serta digunakan sebagai persyaratan menyelesaikan pendidikan S-1. Tugas akhir ini dapat terselesaikan tidak lepas dari bantuan berbagai pihak. Oleh karena itu, penulis mengucapkan terima kasih kepada:

1. Keluarga, Ibu, Bapak dan Saudara yang telah memberikan dorongan spiritual dan material serta seluruh kerabat dan kolega penulis yang banyak membantu proses dalam menyelesaikan buku penelitian ini.
2. Bapak Dr. Tri Arief Sardjono, S.T., M.T. selaku Ketua Jurusan Teknik Elektro, Fakultas Teknologi Industri, Institut Teknologi Sepuluh Nopember
3. Secara khusus penulis mengucapkan terima kasih yang sebesar-besarnya kepada Bapak Dr. Eko Mulyanto Yuniarno, ST., MT. dan Bapak Christyowidiasmoro, ST., MT. atas bimbingan selama mengerjakan tugas akhir.
4. Bapak-ibu dosen pengajar Bidang Studi Teknik Komputer dan Telematika, atas pengajaran, bimbingan, serta perhatian yang diberikan kepada penulis selama ini.
5. Rekan tim Niltava Labs atas fasilitas dan dukungan selama pengerjaan tugas akhir ini.
6. Seluruh teman-teman angkatan e-51 serta teman-teman B201crew Laboratorium Bidang Studi Teknik Komputer dan Telematika.

Kesempurnaan hanya milik Tuhan, untuk itu penulis memohon segenap kritik dan saran yang membangun serta menghatur maaf atas segala kekurangan yang ada dalam penulisan buku ini.

Surabaya, Juli 2015
Penulis

# DAFTAR ISI

# DAFTAR TABEL

# DAFTAR GAMBAR

# BIOGRAFI PENULIS


Aditya Rifa Utama lahir di Kediri pada tanggal 21 Desember 1993. Ia menyelesaikan jenjang Sekolah Dasar di SDN Tengger Lor pada tahun 2005, kemudian melanjutkan pendidikan SMP di SMPN 1 Kunjang dan lulus pada tahun 2008. Pada tahun 2011, penulis menyelesaikan pendidikan SMA di SMAN 2 Pare Kediri. Setelah lulus dari jenjang SMA, penulis melanjutkan pendidikan di Institut Teknologi Sepuluh Nopember dengan mengambil Jurusan Teknik Elektro. Pada semester kelima, penulis mengambil konsentrasi bidang studi Teknik Komputer dan Telematika dan aktif sebagai asisten laboratorium B201. Penulis Selama menempuh pendidikan kuliah, penulis bergabung dalam Himpunan Mahasiswa Teknik Elektro dan mengikuti beberapa kompetisi karya ilmiah seperti Program Kreatifitas Mahasiswa dan Olimpiade Sains Nasional Pertamina.

# BAB 1
# PENDAHULUAN

## 1.1 Latar Belakang

Permukaan dari suatu objek nyata dapat direpresentasikan kedalam data *point cloud* tiga dimensi dengan setiap *point* memiliki komponen *x*, *y*, dan *z*. Data *point cloud* tersebut diperoleh dari hasil tangkapan kamera kedalaman, seperti kamera Kinect. Data *point cloud* dapat dimanfaatkan dalam membuat model 3D CAD (*Computer-aided Design*), animasi dan *augmented reality*. Pada penelitian sebelumnya dengan judul *Raycasting In Three-Dimentional Augmented Reality*[1], objek *virtual* pada *augmented reality* dapat ditampilkan ke lingkungan nyata dengan mengolah data *point cloud*, namun lokasi objek *virtual* tersebut ditentukan secara manual berdasarkan posisi kamera. Data *point cloud* yang digunakan pada penelitian tersebut belum dapat dipisahkan antar model primitif. Pada pengerjaan tugas akhir ini diharapkan mampu melakukan segmentasi model primitif pada data *point cloud* sehingga dapat memisahkan antar model dan menampilkan hasil segmentasi pada layar untuk mengetahui model yang telah terpisah. Proses segmentasi yang telah diimplementasikan dapat dimanfaatkan sebagai tahap *pre-processing* dalam pengolahan data *point cloud* atau referensi dalam pengembangan teknologi *augmented reality* yang lebih interaktif.

## 1.2 Permasalahan

Pada penelitian teknologi *augmented reality*, data *point cloud* dari hasil tangkapan kamera kedalaman belum dapat dipisahkan antar model primitif, sehingga tidak bisa dibedakan antara model satu dengan model yang lain.

## 1.3 Tujuan

Tujuan dari pengerjaan tugas akhir ini adalah melakukan segmentasi model primitif pada data *point cloud* tiga dimensi dari hasil tangkapan kamera kedalaman.

## 1.4 Batasan Masalah

Dalam pengerjaan tugas akhir ini, diberikan beberapa batasan masalah, diantaranya sebagai berikut:

1. Lingkungan nyata berupa lingkungan yang sudah diatur sebelumnya.
2. Model primitif yang disegmentasi adalah *plane*, *sphere* dan *cylinder*.
3. Kamera yang digunakan adalah kamera Kinect.
4. Target platform produk adalah komputer dengan sistem operasi Windows.

## 1.5 Sistematika Penulisan

Laporan penelitian tugas akhir ini tersusun dalam sistematika dan terstruktur sehingga lebih mudah dipahami dan dipelajari oleh pembaca maupun seseorang yang hendak melanjutkan penelitian ini. Alur sistematika penulisan laporan penelitian ini yaitu :

1. Bab I Pendahuluan
   Bab ini berisi uraian tentang latar belakang, permasalahan, tujuan, metodologi, sistematika laporan dan relevansi.
2. Bab II Tinjauan Pustaka
   Pada bab ini berisi tentang uraian secara sistematis teori-teori yang berhubungan dengan permasalahan yang dibahas pada tugas akhir ini. Teori-teori ini digunakan sebagai dasar dalam tugas akhir, yaitu pemodelan objek primitif, metode *Random Sample Consensus*, *point cloud*, dan teori-teori penunjang lainya.
3. Bab III Desain dan Implementasi Sistem
   Bab ini berisi tentang penjelasan terkait sistem yang dibuat. Guna mendukung itu digunakanlah diagram alir, *flowchart*, dan *pseudocode* agar sistem mudah dipahami dan diimplementasikan.
4. Bab IV Pengujian dan Analisa
   Bab ini menjelaskan tentang pengujian yang dilakukan terhadap sistem dalam penelitian ini dan menganalisa sistem. Spesifikasi perangkat keras dan perangkat lunak yang digunakan juga disebutkan dalam bab ini. Tujuannya adalah sebagai variabel kontrol dari pengujian yang dilakukan.

5. Bab V Penutup
   Bab ini merupakan penutup yang berisi kesimpulan yang diambil dari penelitian dan pengujian yang telah dilakukan. Saran dan kritik yang membangun untuk pengembangan lebih lanjut juga dituliskan pada bab ini.

## 1.6 Relevansi

Penelitian ini memiliki relevansi dengan penelitian-penelitian tentang *augmented reality* sebelumnya dan pengembangan metode segmentasi data *point cloud* yang telah dilakukan oleh pengembang pustaka Point Cloud Library. Penelitian ini juga berkaitan dengan pengembangan aplikatif kamera Kinect pada *Unity3d.*

# BAB 2
# TINJAUAN PUSTAKA

Untuk mendukung penelitian dalam tugas akhir ini, dibutuhkan beberapa teori penunjang sebagai bahan acuan dan referensi. Dengan demikian penelitian ini lebih terarah.

## 2.1 *Point Cloud*

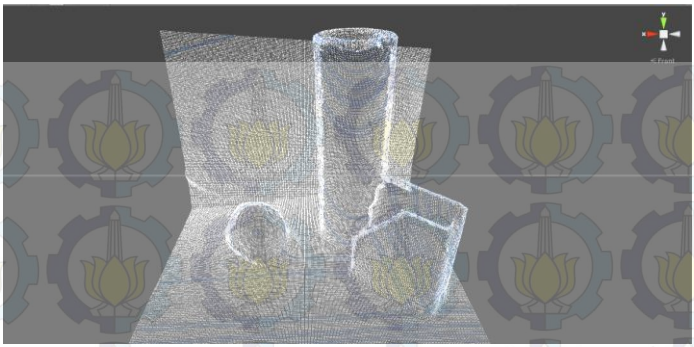
*Point cloud* adalah struktur data yang merepresentasikan kumpulan dari *multi-dimentional points* dan sering digunakan untuk mendeskripsikan data pada objek tiga dimensi. Pada *point cloud* tiga dimensi dari sebuah objek, setiap *point* mewakili nilai koordinat geometri *x, y,* dan *z* dari permukaan objek tersebut.

$$p_1 = \{x_1, y_1, z_1\} \rightarrow P = \{p_1, p_2, p_3, \dots, p_n\} \quad (2.1)$$

Selain data XYZ, setiap *point* dapat memuat informasi tambahan yang lain, seperti spektrum warna, besar intensitas, jarak, dan besar vektor normal. Setiap dimensi data dari *point cloud* disimpan dalam berbagai format untuk mengurangi proses komputasi dan ruang penyimpanan, seperti data XYZ yang disimpan dalam *format Polygon*.

*Point cloud* sangat bermanfaat dalam berbagai kebutuhan. Selain digunakan untuk merepresentasikan geometri dari suatu objek, *point cloud* dapat melengkapi dan menggantikan gambar ketika data gambar memiliki dimensi yang tinggi [2], membuat model 3D CAD (*Computer-aided Design*), metrologi atau inspeksi kualitas, aplikasi kustomisasi visualisasi, animasi dan pengembangan *augmented reality*.

Ada berbagai macam sensor yang dapat menghasilkan informasi tiga dimensi dari suatu objek, seperti laser atau sensor Lidar, kamera *stereo, Time of Flight Cameras*. Setiap sensor memiliki metode tersendiri dalam mendapatkan data *point cloud*. Metode *triagulasi* merupakan metode yang sering digunakan untuk mendapatkan data *point cloud* dengan mendapatkan persamaan dari hasil citra pada dua buah kamera *stereo* dan arah pantulan sinar laser, seperti yang diterapkan pada kamera Kinect. Gambar 2.1 merupakan contoh visualisasi data *point cloud* dari hasil tangkapan kamera Kinect.



**Gambar 2.1** Visualisasi *point cloud* tiga dimensi

Pada umumnya, data *point cloud* tidak dapat langsung digunakan oleh sebagian besar aplikasi 3D, oleh karena itu biasanya dikonversi ke *polygon* atau model segitiga *mesh*, model *NURBS surface*, atau model *CAD* dengan melalui proses yang umum disebut sebagai rekonstruksi permukaan. Ada banyak teknik untuk mengubah *point cloud* ke permukaan tiga dimensi, seperti *Alpha Shapes* dan *Ball Pivoting* [1].

Salah satu aplikasi di mana *point cloud* secara langsung dapat digunakan adalah pada metrologi industri atau inspeksi. Pada hasil produksi, *point cloud* dapat disesuaikan dengan model *CAD* (atau bahkan *point cloud* lainnya), dan dibandingkan untuk mengetahui perbedaannya. Perbedaan ini dapat ditampilkan sebagai *color maps* yang memberikan indikator visual dari deviasi antara bagian produksi dan model *CAD*. Dimensi geometris dan toleransi juga dapat diperoleh langsung dari *point cloud* [3].

## 2.2 Point Cloud Library

Point Cloud Library merupakan *open source framework* untuk pengolahan data *point cloud*. Point Cloud Library berada dibawah lisensi dari BSD (*Barkeley Software Distribution*) dan didukung lebih dari 450 *developer* atau kontibutor [4]. Point Cloud Library berisikan berbagai algoritma yang berkaitan dengan pengolahan citra dua dimensi dan tiga dimensi, seperti *filtering*, estimasi fitur, rekonstruksi permukaan objek, registrasi, *model fitting*, dan segmentasi objek.

Point Cloud Library dapat dijalankan dalam berbagai sistem operasi yang berbeda atau *cross-platform* yakni pada Linux, MacOs, Windows, Android/iOS. Point Cloud Library telah diintegrasikan secara penuh dengan ROS (*Robot Operating System*) dan telah digunakan dalam berbagai proyek dalam komunitas robot. Selain itu, Point Cloud Library mampu mendukung OpenMP dan Intel Threading Building Blocks (TBB) *library* dalam *multi-core parallelization* [5]. Dasar alur proses pada *point cloud library* sebagai berikut,

1. Membuat modul pengolahan, misalkan *filter*, estimasi fitur, atau segmentasi,
2. Menggunakan *setInputCloud* untuk mendapatkan masukan data *point cloud* pada modul pengolahan,
3. Mengatur beberapa parameter sesuai modul pengolahan,
4. Memanggil fungsi *compute, filter, segment* atau fungsi lain yang sejenis untuk mendapatkan hasil.

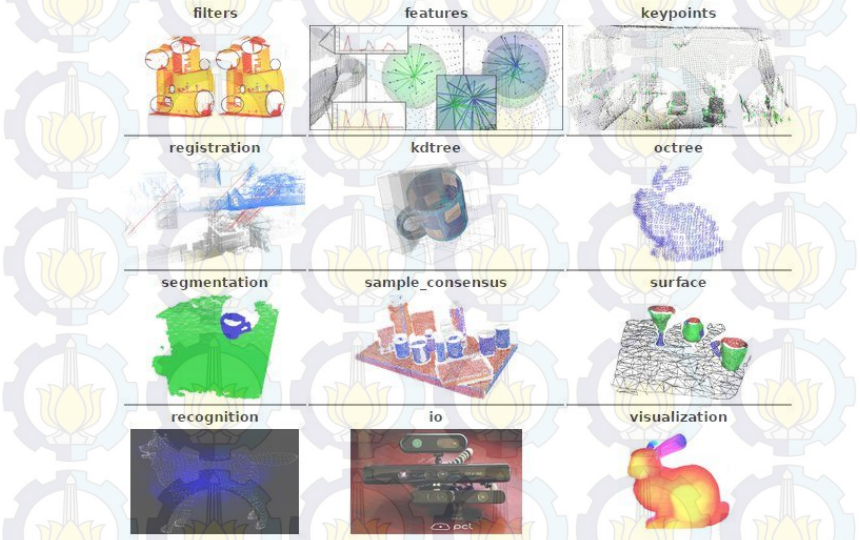
Point Cloud Library dibagi menjadi beberapa seri pustaka yang lebih kecil untuk mempermudah pengembangan dan dapat digunakan secara terpisah. Fitur-fitur dari Point Cloud Library dapat dilihat pada gambar 2.2 dimana dibagi menjadi beberapa fitur, yakni,

a. *Libpcl_filters*, pustaka yang mengimplementasikan metode *filter* data seperti *downsampling*, seleksi *outlier*, ekstrasi indek, dan proyeksi.
b. *Libpcl_features*, pustaka yang mengimplementasikan pengolahan fitur 3D seperti vektor normal dan kelengkungan dari permukaan, estimasi titik batas, dan *moment invariants.*
*c*. *Libpcl_io*, pustaka yang mengimplementasikan operasi I/O seperti menulis dan membaca *file* PCD (*Point Cloud Data*).
d. *Libpcl_segmentation*, pustaka yang mengimplementasikan metode ekstrasi *cluster* dan *model fitting* dengan metode *sample consensus* untuk berbagai model parametrik.
e. *Libpcl_surface*, pustaka yang mengimplementasikan teknik rekonstruksi permukaan, *meshing*, dan *moving least squares*.
f. *Libpcl_registration*, pustaka yang mengimplementasikan metode registrasi data *point cloud*, seperti ICP.
g. *Libpcl_keypoints*, pustaka yang mengimplementasikan metode ekstrasi perbedaan *keypoint* yang dapat digunakan sebagai langkah *preprocessing*.

h. *Libpcl_range_images*, pustaka yang mengimplementasikan pembuatan gambar dengan data *point cloud*.

Point Cloud Library dilengkapi dengan pustaka visualisasi sendiri berbasis VTK (The Visualization Toolkit). VTK menawarkan dukungan *multi-platform* untuk proses *rendering* 3D data *point cloud* dan permukaan termasuk dukungan untuk visualisasi tensor, tekstur, dan metode *volumetric*. Pada versi 0.2, pustaka *visualization* meliputi,

a. Metode untuk rendering dan pengaturan properti visual seperti warna, ukuran dan *opacity* pada setiap *point*.
b. Motode untuk menggambar bentuk 3D dasar di layar seperti *cylinder*, bola, garis, dan poligon baik dari dataset *point* atau persamaan parametrik.
c. Modul visualisasi *histogram* untuk plot 2D.
d. *Multitude* dari geometri dan warna pada koordinat kartesian tiga dimensi.
e. Modul visualisasi untuk *range image*.




**Gambar 2.2** Fitur-fitur pada Point Cloud Library [6]

## 2.3 *Random Sample Consensus*

Salah satu metode dalam proses segmentasi *point cloud* pada Point Cloud Library dengan menggunakan metode Ransac (*Random Sample Consensus*). Ransac merupakan algoritma acak yang digunakan untuk melakukan pendekatan dalam pemodelan objek. Ransac dapat menafsirkan dan merapikan data yang memiliki presentasi *gross error* yang signifikan. Metode ini cocok untuk diaplikasikan dalam analisis citra secara otomatis. Metode *Random Sample Consensus* sering digunakan untuk memecahkan masalah penetapan lokasi atau LDP (*Location Determination Problem*), dimana tujuannya adalah untuk menentukan *point* dalam ruang yang memproyeksikan gambar kedalam satu set *landmark* dengan lokasi yang diketahui [7].

Asumsi dasar dari metode *Random Sample Consensus* adalah bahwa data terdiri dari *inliers*, yaitu data yang distribusinya dapat dijelaskan oleh beberapa set dari parameter model, dan *outliers*, yaitu data yang tidak sesuai dengan model. *Outliers* dapat muncul dikarenakan nilai *noise* yang ekstrim, kesalahan pengukuran, atau hipotesa yang salah tentang interpretasi data.

Algoritma dari Ransac adalah teknik *learning* untuk memperkirakan parameter dari model dengan cara mengambil data *sampling* dari data yang diamati secara acak. Skema *voting* digunakan untuk mendapatkan hasil terbaik karena adanya *inliers* dan *outliers*. Elemen data pada *dataset* yang digunakan untuk memilih satu atau beberapa model. Implementasi dari skema *voting* tersebut didasarkan pada dua asumsi. Asumsi yang pertama adalah fitur *noise* tidak akan memilih secara konsisten untuk setiap model tunggal. Asumsi kedua adalah jumlah fitur *inliers* yang cukup untuk menentukan model terbaik. Algoritma Ransac pada dasarnya terdiri dari dua langkah yang iteratif diulang,

1. *Subset* sampel minimal memiliki data item yang dipilih secara acak dari data masukan. Parameter model didapatkan berdasarkan data *subset* sampel tersebut.
2. Algoritma mencari elemen pada seluruh data masukan yang konsisten dengan parameter model yang didapatkan pada langkah pertama. Sebuah elemen akan dianggap *outliers* jika tidak sesuai dengan parameter model diluar ambang batas kesalahan.

Dua langkah pada algoritma Ransac akan diulang sampai didapatkannya cukup data *inliers* atau sampai batas jumlah literasi yang ditentukan. Jumlah literasi dari metode Ransac dapat ditentukan dengan persamaan 2.2, dimana $k$ adalah jumlah literasi, $p$ adalah probabilitas algoritma menemukan hasil terbaik, $w$ adalah probabilitas didapatkannya *inliers* setiap *point* dipilih, $n$ adalah jumlah minimum data sampel untuk mendapatkan parameter model.

$$k = \frac{\log(1-p)}{\log(1-w/n)} \tag{2.2}$$

Kelebihan dari metode Ransac adalah kemampuan untuk melakukan *robust estimation* terhadap parameter model. Ransac dapat mengestimasikan parameter model dengan akurasi yang tinggi walaupun terdapat banyak *outliers* pada data masukan. Sedangkan kelemahannya adalah tidak adanya konsistensi waktu dalam menetukan parameter dari model.

## 2.4 Kamera Kinect

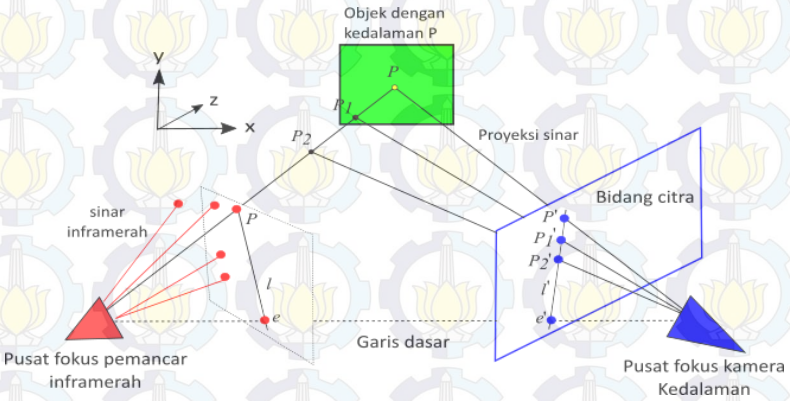
Kinect merupakan kamera *stereovision*, dimana untuk proses pengambilan citra menggunakan dua arah sudut pandang yang berbeda. Kinect memiliki dua buah kamera utama, yaitu kamera *depth* dan kamera RGB, dan sebuah pemancar *inframerah* [1]. Kamera *depth* digunakan untuk mengetahui jarak kedalaman objek dari kamera, yaitu dengan menangkap *inframe*rah yang dipancarkan oleh *IR Transmitter* dan dipantulkan kembali oleh objek. Sedangkan kamera RGB digunakan untuk mengetahui bentuk tekstur atau permukaan dari objek.

Kamera *depth* Kinect memiliki jarak optimal yang harus dipenuhi ketika melakukan proses penangkapan citra yaitu 1.2 meter sampai 3.5 meter. Apabila jarak yang digunakan melebihi atau kurang dari jarak tersebut, maka objek tidak akan diketahui jaraknya oleh kamera *depth* sehingga jarak kedalaman yang dihasilkan bernilai 0.

Pada kamera Kinect terdapat *microphone array* yang digunakan untuk merekam atau menginputkan suara. Kinect juga dilengkapi dengan *tilt* motor dimana dapat digunakan untuk mengatur sudut pandang kamera, sehingga area yang bisa ditangkap oleh kamera menjadi luas. Pengaturan sudut pandang kamera dengan jangkauan sudut kurang lebih 27 derajat dapat menggunakan program tertentu [8].

Kamera RGB memiliki resolusi yang lebih luas apabila dibandingkan dengan kamera *depth*, dimana hasil dari kamera tersebut memiliki resolusi maksimal 640x480 piksel dengan rentang 30 fps berupa informasi citra RGB. Sedangkan kamera *depth* memiliki resolusi maksimal sebesar 320x240 piksel dengan rentang 30 fps.

Pemancar *inframerah* berfungsi untuk memancarkan titik-titik *inframerah* dengan pola yang tidak teratur dan memiliki intensitas yang acak. Sinar *inframerah* yang dipancarkan tersebut akan mengenai objek yang berada didepan kamera dan dipantulkan kembali sehingga dapat dikenali secara cepat dan tepat oleh kamera *depth*. Pada kamera Kinect, posisi objek diketahui melalui dua proses. Pertama, menghitung kedalaman area dengan menganalisa partikel sinar *inframerah* yang tersebar pada area atau *structured light*. Kedua melalui pendekatan lokasi melalui metode *machine learning* dari hasil citra yang didapatkan [9]. Ilustrasi dari proses pancaran dari sinar inframerah dari pemancar menuju sensor kedalaman kamera Kinect dapat dilihat pada gambar 2.3.




**Gambar 2.3** Pancaran sinar *inframerah* kamera Kinect

Data *depth* bisa ditampilkan pada *depth image* dalam beberapa macam warna sesuai yang diinginkan. Warna tersebut untuk mengetahui tingkat jarak kedalaman suatu objek terhadap posisi dari kamera Kinect. Untuk jarak objek yang dekat dengan kamera, warna yang ditampilkan semakin gelap dan apabila semakin jauh jarak objek warna yang ditampilkan semakin terang hingga batas jarak pandang

kamera paling jauh. Kamera Kinect memiliki jarak pandang minimum dan jarak pandang maksimum, sehingga objek yang berada diluar batas jarak tersebut tidak memiliki data *depth*. Demikian juga untuk objek yang tidak bisa memantulkan pancaran dari *inframerah* yang mengakibatkan kamera *depth* tidak menerima data dari objek tersebut sehingga tidak ada data yang diterima[8].

## 2.5 Unity3D

Unity3D merupakan *Integrated Development Environment* (IDE) yang digunakan dalam tugas akhir ini. IDE ini didukung beberapa *plugin* atau *third-party software* yang memudahkan dalam proses perancangan suatu aplikasi.

### 2.5.1 Microsoft Kinect Unity SDK

Microsoft Kinect Unity SDK berisikan pustaka dasar yang mengakses fungsi-fungsi pada kamera Kinect. Pustaka ini menerjemahkan *file* DLL (*Dynamic-link library*) dari Microsoft Kinect kedalam fungsi-fungsi dasar yang dapat diterapkan pada Unity3D, seperti *stream depth data* dan *stream color data*.

### 2.5.2 Particle System

Pada Unity3D, *particles* merupakan gambar atau *mesh* kecil dan sederhana yang ditampilkan serta digerakkan dalam jumlah besar oleh *Particle System* [10]. Sedangkan *Particle System* sendiri merupakan teknik dalam komputer grafis dan *game physics* yang menggunakan banyak *sprites*, gambar dua dimensi kecil, atau objek lain untuk mensimulasikan beberapa jenis fenomena *fuzzy* yang sangat sulit untuk dilakukan dengan teknik rendering biasa. Fenomena tersebut bisa berupa proses reaksi kimia atau fenomena alam seperti api, ledakan, asap, air terjun, dan awan [11].

Pada dasarnya, posisi dan pergerakan dari *Particle System* dikontrol oleh *emmiter*. *Emmiter* bertindak sebagai sumber dari partikel dan lokasinya di ruang 3D menentukan lokasi partikel akan ditampilkan. Geometri dasar 3D seperti kubus dan bidang datar dapat digunakan sebagai *emmiter*. Pada *emmiter* dapat diatur beberapa parameter seperti tingkat *spawning* atau berapa banyak partikel yang ditampilkan setiap unit waktu, kecepatan awal dari partikel, range waktu partikel, dan warna partikel.

Tahap pembaruan dari *Particle System* yang dilakukan untuk setiap *frame* animasi dapat dipisahkan menjadi dua tahap yang berbeda [12]. Kedua tahap tersebut adalah,

1. *Simulation Stage*, selama tahap simulasi, jumlah partikel baru yang harus diciptakan dihitung berdasarkan tingkat *spawning* dan interval antar *update* serta setiap partikel ditempatkan pada pada posisi tertentu dalam ruang 3D berdasarkan posisi *emmiter* dan lokasi yang telah ditentukan.
2. *Rendering Stage*, setelah tahap *update* selesai, setiap partikel akan direnderer sebagai *pixel* tunggal dalam resolusi kecil.

### 2.5.3 Point Cloud Viewer

Point Cloud Viewer merupakan *plugin* dari Unity3D untuk menampilkan data *point cloud*. Keunggulan dari *plugin* ini adalah kemampuan untuk memvisualisasikan lebih dari 10 juta RGB *points* dengan data masukan berupa *file* bertipe data OFF [13].

Metode visualisasi dari Point Cloud Viewer adalah dengan membuat *gameobject* baru sesuai jumlah *point* group yang terbentuk. Masing-masing *gameobject* akan direnderer terpisah dan disimpan dalam *storage* sehingga dapat mempercepat tahap visualisasi pada *point cloud* yang pernah diproses.

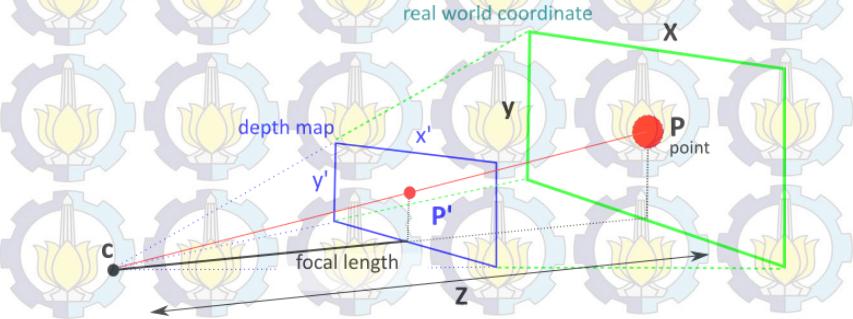
## 2.6 Konversi Peta Kedalaman

Peta kedalaman atau *depth map* adalah gambar dua dimesi yang berisikan informasi tentang jarak permukaan sebuah objek terhadap sudut pandang atau *view point* tertentu. Jarak ini biasanya berhubungan dengan data kedalaman yang berupa koordinat *z* dari sebuah koordinat kartesian. Dalam kasus ini, sumbu *z* yang dimaksud adalah sumbu *z* yang relatif terhadap view dari kamera, dan bukan merupakan sumbu *z* dari koordinat dunia. Untuk mendapatkan data *point cloud* tiga dimensi, maka peta kedalaman harus dikonversi dari data dua dimensi menjadi data tiga dimensi yang memiliki nilai terhadap sumbu *x*, *y*, dan *z* pada bidang kartesian. Skema proses konversi peta kedalaman ke koordinat dunia dapat dilihat pada gambar 2.4.

Dalam proses konversi *depth map* menjadi data *point cloud* tiga dimensi melibatkan parameter intrinsik kamera yang disebut sebagai $K$ dan juga melibatkan data *depth map* yang disebut sebagai $D$. Konversi ini dilakukan secara terpisah pada masing-masing titik di *depth map* yang

dilambangkan sebagai $u$. Berdasarkan kedua data tersebut, persamaan 2.3 digunakan untuk melakukan konversi *depth map* [14].

$$V_i(u) = D_i(u)K^{-1}[u, 1] \quad (2.3)$$




**Gambar 2.4** Skema proses konversi *depth map* ke *point cloud*

## 2.7 Polygon *File*

Polygon *file* format atau PLY adalah deskripsi objek sederhana yang dirancang sebagai format yang nyaman bagi para peneliti yang bekerja dengan model *polygonal* [15]. Versi awal dari format ini digunakan oleh peneliti di Standford University dan UNC Chapel Hill.

PLY menggambarkan sebuah objek sebagai kumpulan simpul, wajah atau permukaan, dan unsur-unsur lain, termasuk properti warna dan arah vektor normal yang dapat disisipkan ke dalam tiap element. Secara sederhana, *file* PLY berisikan daftar *x*, *y*, dan z setiap simpul dan daftar *faces* yang diurutkan berdasarkan indeks. Struktur dari *file* PLY terdiri dari tiga bagian utama. Bagian tersebut adalah,

a. *Header*, yang mencangkup deskripsi dari setiap jenis elemen, termasuk nama dari elemen, jumlah elemen dari objek, dan daftar properti yang berhubungan dengan elemen. *Header* menentukan berapa banyak simpul dan *polygon* dalam *file* serta menyatakan properti yang terkait pada setiap simpul atau titik, seperti elemen koodinat kartesian (*x, y, z*), vektor normal, dan warna. [16]. Pada *header* juga menjelaskan *format* data berupa *binary* atau ASCII.
b. *Vertex list*, merupakan bagian dari *file* PLY yang memuat nilai dari tiap element yang telah dijelaskan pada *header*. Pada

umumnya, *vertex list* berisikan koordinat *point* dengan memiliki tiga properti, yakni properti *x, y, z*.

c. *Face list*, merupakan bagian dari *file* PLY setelah *vertex list*. Deskripsi dari *face list* juga telah dideskripsikan pada *header* setelah deskripsi dari *vertex list*. *Face list* dapat memuat informasi berupa *mesh* dari data *point cloud*.

## 2.8 Bidang Geometri Primitif

Istilah geometri primitif dalam komputer grafik dan sistem CAD telah digunakan dalam berbagai aspek, dengan makna umum yakni bidang geometri paling sederhana yang dapat digambar atau disimpan. Geometri primitif pada umumunya terdiri dari titik, garis, lingkaran, *triangle*, kurva, dan poligon. Sedangkan pada bidang geometri primitif tiga dimensi terdiri dari *plane, sphere, cubes, cylinder, pyramid, dan toroid*. Bidang geometri primitif tiga dimensi merupakan bidang yang memiliki komponen *x, y,* dan *z* dalam koordinat kartesian. Masing-masing bidang geometri primitif memiliki persamaan matematika yang dapat dibentuk dari beberapa titik pada permukaan bidang tersebut.

### 2.8.1 Bidang *Plane*

Persamaan dasar dari bidang geometri *plane* [17] adalah

$$ax + by + cz + d = 0 \tag{2.4}$$

Untuk mendapatkan koefisien $a$, $b$, $c$, dan $d$ dapat dibutuhkan tiga sampel titik dari permukaan *plane*. Sedangkan pada ruang *Euclidean* dimensi berapapun, persamaan dari bidang *plane* dapat ditentukan dengan empat cara, yakni

a. Tiga titik non-linier, titik tersebut tidak berada dalam satu garis,
b. Sebuah garis dan titik yang tidak berada di garis tersebut,
c. Dua garis yang saling sejajar,
d. Dua garis yang berbeda namun berpotongan.

Metode *Cramer's rule*, merupakan salah satu metode untuk menentukan koefisien dari persamaan *plane* dengan informasi koordinat dari tiga titik pada bidang kartesian tiga dimensi. Misalkan titik-titik tersebut,

$$p_1 = (x_1, y_1, z_1),\ p_2 = (x_2, y_2, z_2)\ , p_3 = (x_3, y_3, z_3) \quad (2.5)$$

maka akan didapatkan nilai deskriminan melalui persamaan 2.6.

$$D = \begin{bmatrix} x_1 & y_1 & z_1 \\ x_2 & y_2 & z_2 \\ x_3 & y_3 & z_3 \end{bmatrix} \quad (2.6)$$

Koefisien $a$, $b$, $c$, dan $d$ pada persamaan 2.4 didapatkan dari mensubsitusikan koordinat titik-titik tersebut dalam persamaan 2.7 hingga 2.9.

$$a = \frac{-d}{D} \begin{bmatrix} 1 & y_1 & z_1 \\ 1 & y_2 & z_2 \\ 1 & y_3 & z_3 \end{bmatrix} \quad (2.7)$$

$$b = \frac{-d}{D} \begin{bmatrix} x_1 & 1 & z_1 \\ x_2 & 1 & z_2 \\ x_3 & 1 & z_3 \end{bmatrix} \quad (2.8)$$

$$c = \frac{-d}{D} \begin{bmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ x_3 & y_3 & 1 \end{bmatrix} \quad (2.9)$$

Persamaan tersebut parametrik pada nilai $d$, nilai dari $d$ sama dengan bilangan *non-zero* dan mensubsitusikan nilai $d$ akan menghasilkan satu set solusi. Jarak suatu titik ke bidang *plane* dapat dicari dengan mensubsitusi persamaan *plane* dan suatu titik ke dalam persamaan 2.10.

$$Ds = \frac{|ax_1 + by_1 + cz_1 + d|}{\sqrt{a^2 + b^2 + c^2}} \quad (2.10)$$

Jika hasil $Ds$ = 0 maka titik berada pada permukaan bidang plane tersebut.

### 2.8.2 Bidang *Sphere*

Bidang *sphere* adalah sebuah objek geometri yang bulat sempurna dalam ruang tiga-dimensi. Jika dianalogikan dengan lingkaran pada ruang dua dimensi, bidang *sphere* didefinisikan secara matematika sebagai himpunan titik-titik yang memiliki jarak $R$ yang

sama dari suatu titik dalam koordinat kartesian. Nilai *R* tersebut adalah jari-jari dari *sphere*, dan suatu titik tersebut adalah pusat dari *sphere*. Jarak lurus terbesar dari titik di permukaan *sphere* ke titik lain dan melewati pusat dari *sphere* merupakan dua kali dari nilai *R* atau disebut dengan diameter. Didalam geometri analitik, persamaan dari bidang *sphere* dengan koordinat titik pusat $(x_0, y_0, z_0)$ dan jari-jari *R* adalah

$$(x - x_0)^2 + (y - y_0)^2 + (z - z_0)^2 = R^2 \qquad (2.11)$$

dimana $x$, $y$, $z$ merupakan koordinat dari semua titik yang berada di permukaan bola tersebut pada bidang kartesian. Selain persamaan tersebut, semua titik pada permukaan *sphere* juga akan memenuhi persamaan 2.12.

$$\begin{gathered} x = x_0 + R\cos\theta \ \sin\varphi, \\ y = y_0 + R\sin\theta \ \sin\varphi, \\ z = z_0 + R\cos\varphi, \\ (0 \le \theta \le 2\pi \ dan \ 0 \le \varphi \le \pi) \end{gathered} \qquad (2.12)$$

Persamaan model *sphere* dapat dicari dari empat buah titik pada permukaan bidang *sphere* [18]. Empat titik tersebut tidak berada pada bidang *plane* yang sama. Misalkan titik-titik tersebut,

$$\begin{gathered} p_1 = (x_1, y_1, z_1), \\ p_2 = (x_2, y_2, z_2), \\ p_3 = (x_3, y_3, z_3), \\ p_4 = (x_4, y_4, z_4) \end{gathered} \qquad (2.13)$$

Maka dapat diselesaikan melalui persamaan determinan berikut,

$$\begin{vmatrix} x^2+y^2+z^2 & x & y & z & 1 \\ {x_1}^2+{y_1}^2+{z_1}^2 & x_1 & y_1 & z_1 & 1 \\ {x_2}^2+{y_2}^2+{z_2}^2 & x_2 & y_2 & z_2 & 1 \\ {x_3}^2+{y_3}^2+{z_3}^2 & x_3 & y_3 & z_3 & 1 \\ {x_4}^2+{y_4}^2+{z_4}^2 & x_4 & y_4 & z_4 & 1 \end{vmatrix} = 0 \qquad (2.14)$$

Persamaan 2.14 dapat disederhanakan dengan mengevaluasi *cofactors* untuk baris pertama pada persamaan determinan tersebut,

$$(x^2 + y^2 + z^2)M_{11} - xM_{12} + yM_{13} - zM_{14} + M_{15} = 0 \quad (2.15)$$

Dengan mensubsitusi persamaan 2.11, maka koordinat titik pusat $(x_0,y_0,z_0)$ dan jari-jari $R$ diperoleh dari persamaan 2.16 hingga 2.19.

$$x_0 = +0.5\,\frac{M_{12}}{M_{11}} \quad (2.16)$$

$$y_0 = -0.5\,\frac{M_{13}}{M_{11}} \quad (2.17)$$

$$z_0 = +0.5\,\frac{M_{14}}{M_{11}} \quad (2.18)$$

$$R^2 = {x_0}^2 + {y_0}^2 + {z_0}^2 - \frac{M_{15}}{M_{11}} \quad (2.19)$$

Tidak akan ditemukan solusi persamaan *sphere* apabila $M_{11}$ bernilai 0. Jika persamaan *sphere* yang ditemukan tidak termasuk himpunan titik pada permukaan *sphere* yang diinginkan, maka seluruh titik yang disubsitusikan berada dalam bidang *plane* yang sama atau tiga titik tersebut berada dalam satu garis lurus.

### 2.8.3 Bidang *Cylinder*

Bidang *cylinder* didefinisikan sebagai himpunan semua titik dalam ruang *euclidean* tiga dimensi atau $R^3$ yang terletak pada jarak tetap dari garis lurus yang merupakan sumbu dari bidang *cylinder*. Dalam geometri analitik, *cylinder* didefinisikan oleh jari-jari, tinggi dan arah bidang *cylinder*. Untuk dapat menghitung bidang *cylinder* dari *minimum set of points* (MSS), tiga buah titik harus tidak kolinier [19]. Dari ketiga titik tersebut akan diperoleh lingkaran setelah menyelesaikan persamaan bidang *plane*, persamaan 2.4. Besar jari-jari *r* dan titik tengah dari bidang *cylinder* $c = (x_0, y_0, z_0)$ dapat diperoleh dengan persamaan 2.20 dan 2.21.

$$d(p_1, c) = d(p_2, c) = d(p_3, c) = r \qquad (2.20)$$

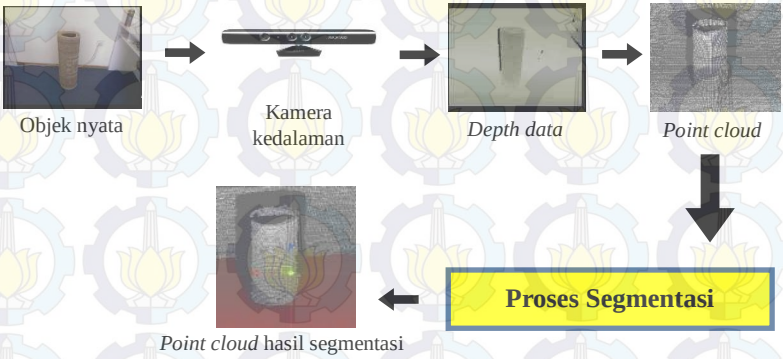
$$d(p_1, c) = \sqrt{(x_i - x_0)^2 + (y_i - y_0)^2 + (z_i - z_0)^2} \qquad (2.21)$$

sedangkan pada arah dari bidang *cylinder* diperoleh dari normal vektor dari bidang *plane* yang diestimasi sebelumnya.

# BAB 3
# DESAIN DAN IMPLEMENTASI SISTEM

## 3.1 Desain Sistem

Tugas akhir ini bertujuan untuk membuat aplikasi yang mampu melakukan segmentasi model primitif tiga dimensi berdasarkan data *point cloud* pada lingkungan nyata. Data *point cloud* diperoleh melalui perekaman secara langsung dengan kamera Kinect, hasil *recording* dari kamera Kinect dalam bentuk *file* Binary dan *file* PLY. Visualisasi dari data *point cloud* akan ditampilkan berupa *point* yang mewakili koordinat dari tiap data *point cloud* sesuai dengan koordinat dunia. Data *point cloud* yang merepresentasikan objek tiga dimensi sederhana yang tersegmentasi akan ditampilkan dengan warna yang berbeda dengan objek yang lain. Gambaran umum dari sistem dari tugas akhir ini diilustrasikan dalam gambar 3.1.




**Gambar 3.1** Gambaran umum desain sistem

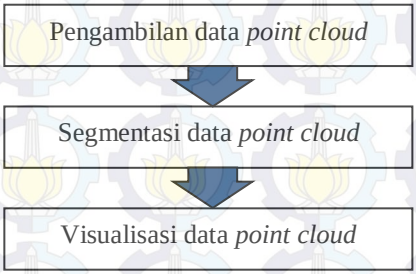
## 3.2 Alur Kerja Sistem

Pada tahap ini akan dijelasan mengenai rancangan sistem pada perangkat lunak. Rancangan sistem perangkat lunak menggunakan pustaka Point Cloud Library dan Microsoft Kinect Unity SDK serta didesain pada IDE Unity3D, IDE tersebut juga sebagai perender grafis

untuk visualisasi data *point cloud*. Tahapan proses dari tugas akhir ini sebagai berikut,

a. Pengambilan data *point cloud*
b. Segmentasi data *point cloud*
c. Visualisasi data *point cloud*

Untuk ketiga proses ini dapat diilustrasikan pada gambar 3.2.




**Gambar 3.2** Diagram alir alur kerja sistem

## 3.3 Pengambilan Data *Point Cloud*

Data *point cloud* diperoleh dengan berdasarkan peta kedalaman atau *depth map* dari hasil tangkapan kamera Kinect. Pada tugas akhir ini, terdapat dua cara dalam proses pengambilan data *point cloud*.

### 3.3.1 *Multiframe*

*Multiframe* pada proses pengambilan data *point cloud* adalah kumpulan peta kedalaman dua dimensi dalam jangka waktu tertentu. Cara pengambilan ini diterapkan pada saat *stream* dari kamera kedalaman secara langsung dan *replay* hasil rekaman dari *stream* sebelumnya. Hasil rekam dari proses *stream* pada kamera Kinect berupa *file* Binary yang berisikan kumpulan peta kedalaman selama proses merekam. Tahapan dalam cara pengambilan secara *multiframe* dibagi menjadi dua bagian,

#### *3.3.1.1 Menyimpan dan membaca file binary*

Proses menyimpan dan membaca *file* binary merupakan tahapan yang tidak selalu dilaksanakan dalam cara pengambilan data *point cloud* secara *multiframe*. Tahapan ini dapat dilakukan jika *user* ingin

memisahkan proses merekam peta kedalaman dari lingkungan nyata dengan proses segmentasi. Dari hasil *stream* pada kamera kinect maka didapatkan peta kedalaman dengan lebar 320 *pixel* dan tinggi 240 *pixel*. Setiap *cell* pada peta kedalaman tersebut memuat data *depth* dari suatu titik pada permukaan objek. Setiap *frame* dari peta kedalaman akan di-*serialize* dan disimpan dengan format *file* Binary pada *storage* saat proses merekam berakhir. *File* Binary akan dibaca berdasarkan *playback speed* tertentu untuk didapatkan kembali data peta kedalaman sebelum melalui proses konversi ke data *point cloud*.

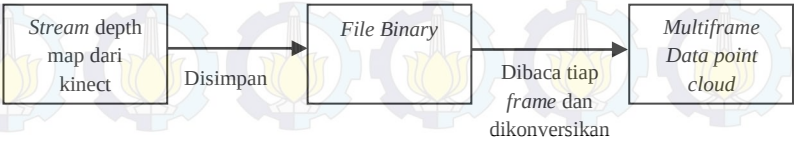
### *3.3.1.2 Mengkonversikan peta kedalaman*

Peta kedalaman yang didapatkan kamera Kinect sebenarnya adalah matriks dua dimensi yang berisi data kedalaman. Data ini akan dikonversi menjadi data tiga dimensi atau *point cloud* dengan menggunakan persamaan 2.3. Persamaan tersebut diaplikasikan dalam *pseudocode* pada gambar 3.3.

```
FUNCTION DepthToPointCloud (float x, float y, float depth) {
DEKLARASI
   Point as float[3] contain zero
BEGIN
   IF (depth > 0) DO
      Point[2] = depth/ 100
      Point[0] = (x - 0.5) * 0.003501 * Point[2] * 320
      Point[1] = (0.5 - y)* 0.003501 * Point[2] * 240
   END IF
   RETURN Point
END FUNCTION
```

**Gambar 3.3** *Pseudocode* konversi peta kedalaman

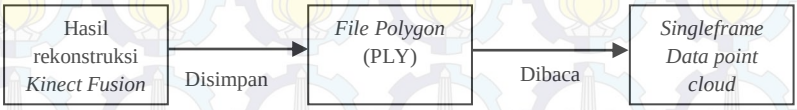
Jika tidak melalui tahapan menyimpan dan membaca *file* Binary, maka peta kedalaman dari *stream* kamera Kinect akan langsung dikonversikan ke data *point cloud* tiap *frame*. Ilustrasi dari proses pengambilan data *point cloud* secara *multiframe* dapat dilihat pada gambar 3.4.




**Gambar 3.4** Diagram alir pengambilan data *point cloud multiframe*

### 3.3.2 *Singleframe*

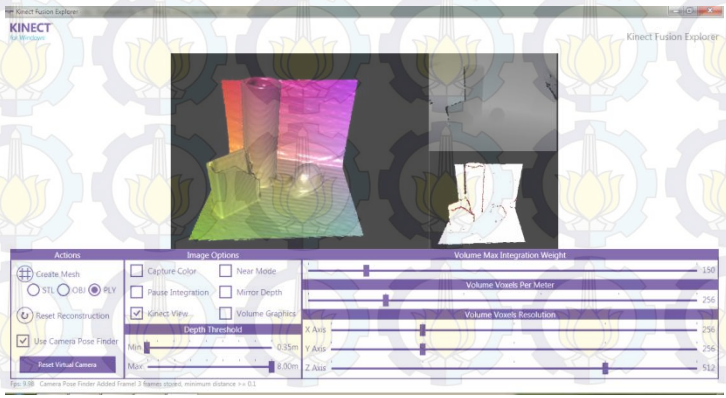
*Singleframe* dirancang dengan masukan data *point cloud* berupa *file* Polygon atau PLY dari *storage*. *File* Polygon dapat diperoleh dari proses rekonstruksi dengan bantuan pustaka Kinect Fusion. *File* Polygon dapat pula didapatkan dari mengunduh dari internet. Tahapan dalam proses pengambilan data *point cloud* secara *singleframe* pada tugas akhir ini diilustrasikan dalam diagram alir pada gambar 3.5.




**Gambar 3.5** Diagram alir pengambilan data *point cloud singleframe*

#### *3.3.2.1 Menyimpan Hasil Rekonstruksi* Kinect Fusion

Hasil *stream* lingkungan nyata dari kamera Kinect direkonstruksi dengan pustaka Kinect Fusion dan disimpan dalam format *file* Polygon. Tampilan dari pustaka Kinect Fusion yang digunakan sesuai gambar 3.6. Hasil rekonstruksi tersebut merupakan *singleframe* data *point cloud* tiga dimensi yang memuat elemen *x*, *y*, *z* dalam koordinat dunia sehingga tidak memerlukan konversi dari peta kedalaman menjadi data *point cloud*.



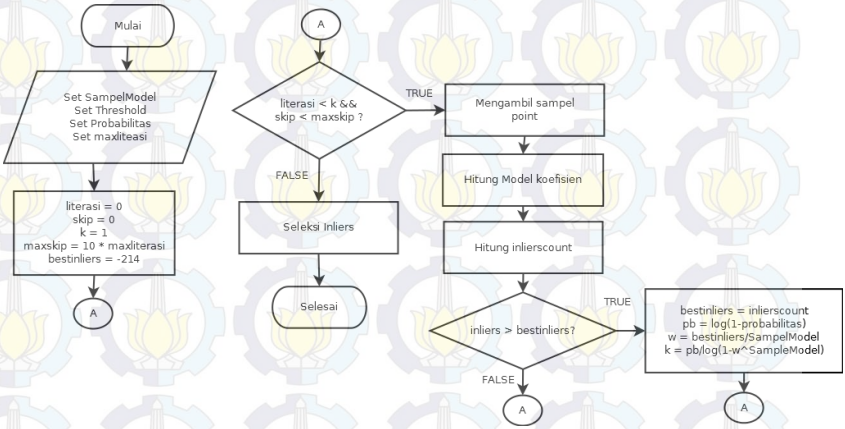

**Gambar 3.6** *Interface* dari Kinect Fusion

### *3.3.2.2 Membaca File Polygon*

Setiap *point* dari data *point cloud* diperoleh dari membaca perbaris pada bagian *vertex list*. Informasi jumlah point dalam satu *frame* diperoleh pada bagian *header file* beserta elemen-elemen dari setiap *vertex*.

## 3.4 Segmentasi Data *Point Cloud*

Segmentasi data *point cloud* dilakukan setiap *frame* baik pada masukan berupa *multiframe* maupun *singleframe*. Proses segmentasi menggunakan metode Random Sample Consensus dengan luaran data *inliers* atau list *point* dari data *point cloud* yang merupakan model tersegmentasi, dan data *outliers* atau list *point* yang tidak termasuk model tersegmentasi. Proses segmentasi dari data point cloud digambarkan dalam *flowchart* pada gambar 3.7.




**Gambar 3.7** *Flowchart* proses segmentasi data *point cloud*

Tahapan proses dalam proses segmentasi data *point cloud* adalah,

### 3.4.1 Mengatur Parameter Segmentasi

Terdapat empat buah parameter penting dalam setiap proses segmentasi pada seluruh model yang dirancang. Parameter tersebut adalah,

1. Jumlah sampel, merupakan jumlah minimal dari titik atau *point* pada permukaan model untuk menentukan persamaan dari model atau bidang tersebut. Jumlah minimum sampel yang dibutuhkan dari masing-masing model primitif adalah,
   a. Model *plane*, dibutuhkan tiga sampel *point*,
   b. Model *sphere*, dibutuhkan empat sampel *point*,
   c. Model *cylinder*, dibutuhkan dua sampel *point*,
   d. Model *cone*, dibutuhkan tiga sampel *point*,
   e. Model *line*, dibutuhkan dua sampel *point*,
   f. Model *circle2D*, dibutuhkan tiga sampel *point*,
   g. Model *circle3D*, dibutuhkan tiga sampel *point*
2. *Threshold* model, merupakan batas maksimum jarak dari suatu *point* terhadap hipotesa persamaan model. Besar nilai *threshold* berbanding lurus dengan probabilitas ditemukannya *inliers* pada setiap *point* dalam satu literasi. Kondisi ini menyebabkan jumlah *inliers* yang didapatkan semakin besar.
3. Probabilitas ditemukannya model, merupakan parameter yang berkaitan dengan jumlah literasi pada penentuan persamaan model terbaik. Semakin besar nilai probabilitas ditemukannya model mengakibatkan semakin sedikit jumlah literasi yang akan berjalan.
4. Literasi maksimum, merupakan parameter yang berkaitan dengan jumlah literasi total pada penentuan hipotesa persamaan model dari setiap pengambilan sampel *point cloud*.

### 3.4.2 Menentukan Sampel *Point Cloud*

Jumlah sampel *point cloud* yang diperlukan berdasarkan parameter jumlah sampel dari model. Pemilihan sampel *point cloud* dilakukan secara acak. Proses pengacakkan dari sistem operasi saat ini sangat baik, dimana untuk jumlah *point cloud* yang sama akan didapatkan pola hasil pengacakkan yang sama pula. Oleh karena itu dilakukan penyimpanan hasil pengacakan untuk menghindari pengambilan sampel yang sama dalam setiap literasi.

### 3.4.3 Menghitung Koefisien Model

Koefisien dari model didapatkan melalui subsitusi persamaan geometri dasar dari model yang ditentukan dengan data dari sampel

*point*. *Pseudocode* dari algoritma menghitung koefisien model bidang *plane* dijelaskan pada gambar 3.8.

```
FUNCTION KoefisienModelPlane
DECLARATION
   p1p0, p2p0 as float[3]
   Dot, Norm as float
   Input cloud as float[3,3]
   Output coefficients as float[4]
BEGIN
   p1p0 = cloud[1] - cloud[0]
   p2p0 = cloud[2] - cloud[0]
   IF ((p1p0, p2p0) is collinear?) THEN
      RETURN false
   ENDIF
   coefficients[0]=(p1p0[1]*p2p0[2])-(p1p0[2]*p2p0[1])
   coefficients[1]=(p1p0[2]*p2p0[0])-(p1p0[0]*p2p0[2])
   coefficients[2]=(p1p0[0]*p2p0[1])-(p1p0[1]*p2p0[0])
   coefficients[3]=0
   Dot = sum of sqr(coefficients)
   Norm = sqrt(Dot)
   coefficients = coefficients / norm
  RETURN true
END FUNCTION
```

**Gambar 3.8** *Pseudocode* fungsi untuk menghitung koefisien bidang *plane*

Koefisien yang diperoleh dalam *pseudocode* pada gambar 3.8 merupakan nilai $a$, $b$, $c$, dan $d$ dari persamaan 2.4. Variabel *cloud* merupakan sampel data dari proses sebelumnya, sedangkan variable *coefficients* merupakan koefisien model. Koefisien dari bidang *sphere* pada persamaan 2.9, yakni titik pusat ($x_0$, $y_0$, $z_0$) dan jari-jari $R$ diperoleh dari *pseudocode* pada gambar 3.9. Model *cylinder* memiliki tujuh koefisien, dimana tiga koefisien merupakan koordinat *x, y, z* suatu titik pada *axis* model *cylinder*, tiga koefisien menjelaskan arah dari *axis* model *cylinder*, dan satu koefisien merupakan jari-jari model *cylinder*. Dua buah sampel *point* yang digunakan akan dihitung vektor normalnya berdasarkan *point* tetangga. Metoda yang digunakan dalam pencarian *point* tetangga adalah *Kdtree*. Dari *point* tetangga akan didapatkan persamaan bidang *plane* untuk mendapatkan nilai vektor normal dari sampel *point* tersebut. Data koordinat *x, y, z* dan vektor normal dari sampel *point* akan digunakan untuk mendapatkan koefisien dari persamaan model *cylinder*.

```
FUNCTION KoefisienModelSphere
DECLARATION
   temp as float[4,4]
   m11, m12, m13, m14, m15 as float
   Input cloud as float[4,3]
   Output coeff as float[4]
BEGIN
   FOR i <- 0 to 4 DO
      temp[i][0] = cloud[i][0]
      temp[i][1] = cloud[i][1]
      temp[i][2] = cloud[i][2]
      temp[i][3] = 1
   ENDFOR
   m11 = Determinant(temp)
   IF (m11 == 0)
      RETURN false
   ENDIF
   FOR j <- 0 to 4 DO
      temp[i][0] = square of cloud[j][0] + square of//
      cloud[j][1] +  square of cloud[j][2]
   ENDFOR
   m12 = Determinant(temp)
   FOR k <- 0 to 4 DO
      temp[k][1] = temp[k][0]
      temp[k][0] = cloud[k][0]
   ENDFOR
   m13 = Determinant(temp)
   FOR l <- 0 to 4 DO
      temp[l][2] = temp[l][1]
      temp[l][1] = cloud[l][1]
   ENDFOR
   m14 = Determinant(temp)
   FOR m <- 0 to 4 DO
      temp[m][0] = temp[m][2]
      temp[m][1] = cloud[m][0]
      temp[m][2] = cloud[m][1]
      temp[m][3] = cloud[m][2]
   ENDFOR
   m15 = Determinant(temp)
   coeff[0] = 0.5f * m12 / m11
   coeff[1] = 0.5f * m13 / m11
   coeff[2] = 0.5f * m14 / m11
   coeff[3] = Sqrt(square of coeff[0]+square of coeff[1]//
   +square of coeff[2] - m15 / m11);
   RETURN true
END FUNCTION
```

**Gambar 3.9** *Pseudocode* fungsi untuk menghitung koefisien bidang *sphere*

### 3.4.4 Menentukan Koefisien Model Terbaik

Koefisien model terbaik ditentukan dari jumlah *inliers* terbanyak dari semua hipotesa persamaan yang didapatkan selama proses literasi berjalan. Jumlah *inliers* merupakan jumlah *point* yang memiliki jarak yang sesuai dengan threshold yang telah ditentukan dari koordinat *point* tersebut terhadap persamaan hipotesa model. Jumlah *inliers* yang didapatkan setiap pengujian hipotesa persamaan akan mempengaruhi nilai *k* sesuai persamaan 2.2. *Pseudocode* dari algoritma menghitung jumlah *inliers* dari koefisien model *plane* sesuai dengan persamaan 2.8 dijelaskan pada gambar 3.10.

```
FUNCTION CountInliersPlane
DECLARATION
   Point as float[3]
   Dot as double
   Input cloud as float[n,3]
   Input coefficients as float[4]
   Input threshold as double
   Output npoint as int
BEGIN
   For i<-0 to n DO
      Point = [cloud[i][0], cloud[i][1], cloud[i][2], 1]
      Dot = sum (Point*coefficients)
      IF (Dot < threshold)
        Increment npoint
      ENDIF
   ENDFOR
END FUNCTION
```

**Gambar 3.10** *Pseudocode* fungsi untuk menghitung *inliers* model *plane*

Sebelum dihitung jumlah *inliers*, validitas dari setiap hipotesa persamaan ditentukan berdasarkan jumlah koefisien model yang didapatkan. Pada model *sphere*, jumlah *inliers* diperoleh dengan menggunakan *pseudocode* pada gambar 3.11 .

### 3.4.5 Seleksi *Inliers* Berdasarkan Koefisien Model Terbaik

Setelah mendapatkan koefisien terbaik dari proses literasi, setiap *point* akan dihitung jaraknya kembali terhadap persamaan model terbaik. *Point* yang memiliki jarak yang sesuai *threshold* termasuk data *inliers*. Sedangkan *point* yang memiliki jarak tidak sesuai *threshold* termasuk data *outliers*. Algoritma dari seleksi *inliers* dari model *plane* memiliki

```
FUNCTION CountInliersSphere
DECLARATION
   Input cloud as float[n,3]
   Input coeff as float[4]
   Input double threshold
   Output npoint as int
BEGIN
   FOR i <- 0 to n DO
      IF (Abs(Sqrt(Sqr(cloud[i][0]-coeff[0])+ //
      Sqr(cloud[i][1]-coeff[1])+ //
      Sqr(cloud[i][2]-coeff[2]))-coeff[3] ) < threshold )
        npoint++
      ENDIF
   ENDFOR
END FUNCTION
```

**Gambar 3.11** *Pseudocode* fungsi untuk menghitung *inliers* model *sphere*

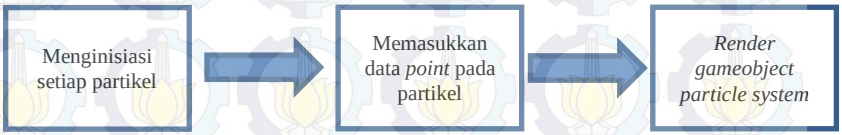
struktur yang sama dengan algoritma pada gambar 3.10. Setiap *point* yang memiliki jarak kurang dari *threshold* akan disimpan dalam larik sebagai data *inliers*. Pada model *sphere*, algoritma menghitung seleksi *inliers* memiliki struktur yang sama dengan algoritma menghitung jumlah *inlier* model *sphere*.

## 3.5 Visualisasi Data *Point Cloud*

Visualisasi dari data *point cloud* menggunakan *Particle System* dan Point Cloud Manager. Data *inliers* diberikan warna yang berbeda dengan data *point cloud* yang lain atau *outliers* untuk membedakan antar objek.

### 3.5.1 Visualisasi dengan *Particle System*

Diagram alur visualisasi dengan *Particle System* diilustrasikan pada gambar 3.12.




**Gambar 3.12** Diagram alir proses visualisasi dengan *Particle System*

Tahapan proses visualisasi dengan *Particle System* terdiri atas,

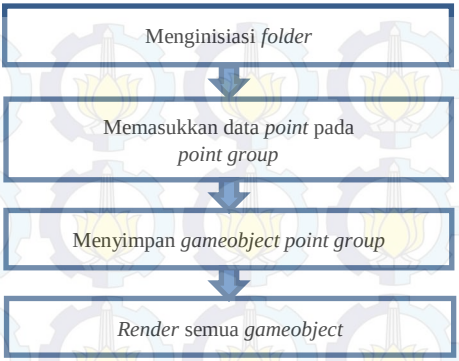
1. Menginisiasi setiap partikel
   Jumlah partikel yang diinisiasi sejumlah data *point* pada *point cloud.* Beberapa elemen dari partikel dipersiapkan, seperti ukuran, koordinat dan warna.
2. Memasukkan data *point* pada partikel
   Data koordinat dari setiap *point cloud* dimasukkan kedalam setiap point, sedangkan elemen warna dikorelasikan dengan data *inliers* untuk membedakan antara data *inliers* dan data *outliers.*
3. *Render gameobject Particle System*
   Proses *render* pada *Particle System* dilakukan secara keseluruhan setelah proses memasukkan data *point* cloud beserta properti tampilan sehingga dihasilkan satu *gameobject.*

### 3.5.2 Visualisasi dengan Point Cloud Viewer

Proses visualisasi dengan Point Cloud Viewer memiliki empat tahapan. Tahapan tersebut terdiri atas,

1. Menginisiasi *folder*
   Point Cloud Viewer mempunyai mekanisme untuk menyimpan *gameobject.* Setiap *gameobject* tersebut disimpan dalam *file asset*. Pada awal proses dilakukan pengecekkan *folder resource* sebagai tempat penyimpanan *asset* dan akan mempersiapkan *folder* jika tidak tersedia. Apabila ingin menampilkan visualisasi yang telah tersimpan, maka cukup dengan mengakses *gameobject* yang telah ada.
2. Memasukkan data *point cloud* pada *point group*
   Koordinat dan warna dari setiap *point* akan disimpan dalam larik. Larik dibagi menjadi sejumlah *point group*. Kapasitas dari setiap *point group* ditentukan pada awal proses visualisasi.
3. Menyimpan *gameobject point group*
   *Gameobject* dari setiap *point group* disimpan dalam *file asset* pada *folder* yang telah diinisiasi.
4. *Render* semua *gameobject*
   *Gameobject* yang telah disimpan digabungkan dan ditampilkan dalam satu *prefab.*

Keempat tahapan dalam visualisasi dengan Point Cloud Viewer dapat diilustrasikan kedalam diagram alur pada gambar 3.13.




**Gambar 3.13** Diagram alir proses visualisasi dengan Point Cloud Viewer

## 3.6 Perancangan Alur Visualisasi

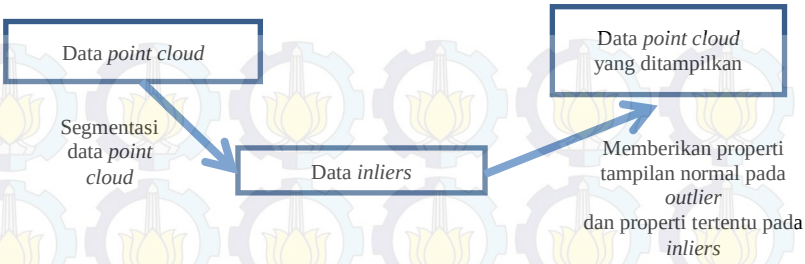
Proses segmentasi data *point cloud* yang membutuhkan waktu yang cukup lama, maka disusunlah rancangan alur visualisasi dari data *point cloud* agar proses visualisasi dapat berjalan dengan baik.

### 3.6.1 Visualisasi Secara Sinkronous

Pada visualisasi secara sinkronous, data *point cloud* yang ditampilkan ke layar akan diproses setelah tahap segmentasi berakhir. Alur data *point cloud* pada rancangan visualisasi ini adalah searah. Setelah proses segmentasi berakhir, data *inliers* akan dikorelasikan dengan data masukan *point cloud* untuk menentukan properti tampilan. Dalam tugas akhir ini properti yang dibedakan antara *inliers* dan *outliers* adalah warna. Ilustrasi dari perancangan visualisasi secara sinkronous dapat dilihat pada gambar 3.14.

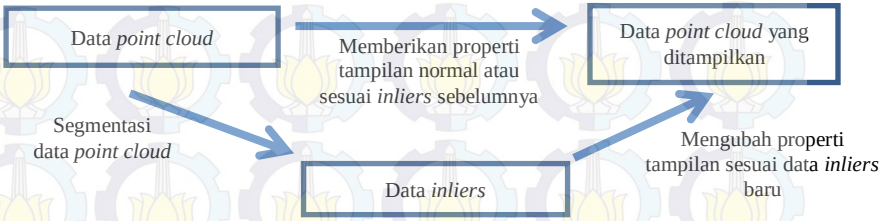
### 3.6.2 Visualisasi Secara Asinkronous

Pada visualisasi secara asinkronous, data *point cloud* yang ditampilkan pada layar memiliki dua alur. Yang pertama saat didapatkan data masukan *point cloud*. Pada kondisi ini, jika tidak tersedia data *inliers* maka data *point cloud* akan ditampilkan seluruhnya sebagai *outliers*. Jika data *inliers* telah tersedia dari proses segmentasi




**Gambar 3.14** Diagram alir proses visualisasi secara sinkronous

sebelumnya, maka data masukan tersebut akan dikorelasikan dengan data inliers tersebut. Kondisi yang lain adalah saat didapatkan data *inliers* baru, maka properti tampilan dari *point cloud* yang akan ditampilkan akan disesuaikan dengan data tersebut. Ilustrasi dari perancangan visualisasi secara asinkronous dapat dilihat pada gambar 3.15.



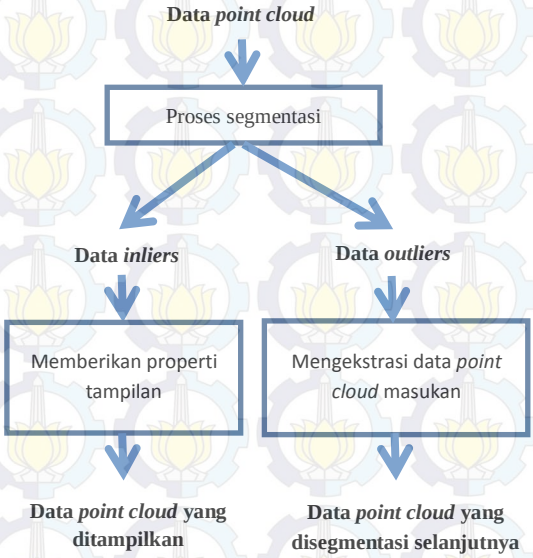

**Gambar 3.15** Diagram alir proses visualisasi secara asinkronous

## 3.7 Perancangan Alur Multisegmentasi

Alur multisegmentasi dirancang untuk mendapatkan data *inliers* dari model dengan koefisien model terbaik kedua, ketiga dan seterusnya. Kondisi dikarenakan metoda Ransac yang diterapkan akan memilih koefisien terbaik dalam setiap proses segmentasi sebagai data *inliers*. Tahapan utama dalam alur proses multisegmentasi sebagai berikut,

1. Dilakukan proses segmentasi model sesuai jenis model dan parameter yang ditentukan. Dari proses segmentasi akan dihasilkan data *inliers* atau data yang merepresentasikan model yang tersegmentasi, dan data *outliers* atau data selain model yang tersegmentasi.
2. Data *point cloud* yang termasuk data *inliers* diberikan properti tampilan untuk ditampilkan dalam layar, sedangkan data *outliers* digunakan sebagai data masukan *point cloud* pada proses segmentasi selanjutnya.

Alur multisegmentasi dapat diilustrasikan dalam diagram alir pada gambar 3.16.




**Gambar 3.16** Diagram alir alur multisegmentasi

# BAB 4
# PENGUJIAN DAN ANALISA

Pada bab ini dibahas mengenai pengujian dari perangkat lunak yang telah direalisasikan untuk mengetahui apakah fungsi dari sistem yang direncanakan telah bekerja sesuai dengan harapan. Pengujian pada penelitian ini dilakukan dalam beberapa bagian, yakni pengujian terhadap performa segmentasi, jenis visualisasi, alur visualisasi, dan multisegmentasi. Pengujian dilakukan pada pada sistem operasi Windows 7 Home Premium dengan *Integrated Development Environment* (IDE) yang digunakan adalah Unity3D versi 4.5.2. Sedangkan kamera kedalaman yang digunakan adalah kamera Kinect XBOX 360. Adapun spesifikasi komputer yang digunakan dalam pengujian ini dijelaskan pada tabel 4.1.

**Tabel 4.1** Spesifikasi Komputer

| **Komponen** | **Spesifikasi** |
|---|---|
| Sistem Operasi | Windows 7 Home Premium 64-bit |
| *Processor* | Intel Core ™ i5-4200M CPU @2.5 GHz (4CPUs) ~2.5GHz |
| *Memory* | 4096MB RAM |
| Versi *DirectX* | DirectX 11 |
| Versi *DxDiag* | 6.01.7600.16385 32-bit Unicode |
| *Display Adapter Name* | Intel HD Graphic 4600 – AMD Radeon HD 8500M |
| *Approx. Total Memory* | 3858 |

## 4.1 Pengujian Segmentasi *Point Cloud Singleframe*

Pengujian ini dilakukan untuk mengetahui performa dari perangkat lunak yang telah direalisasikan dalam melakukan proses segmentasi dan visualisasi data *point cloud singleframe*. *File* PLY

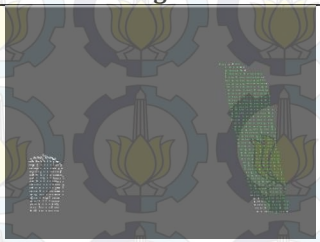
diperoleh dari hasil rekonstruksi dari kamera Kinect dengan bantuan pustaka Kinect Fusion.

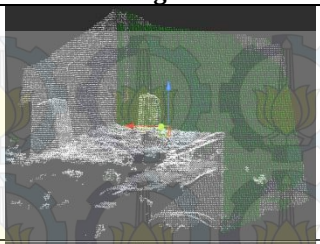
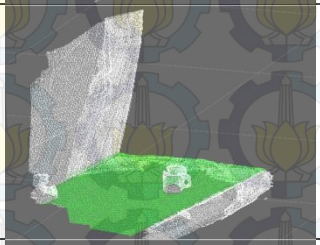
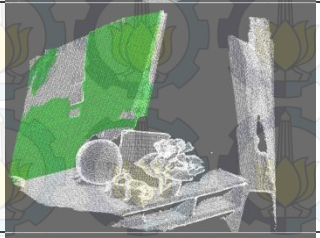
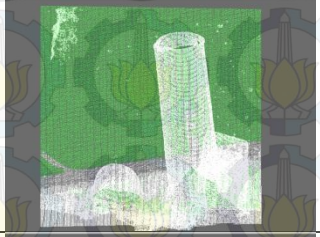
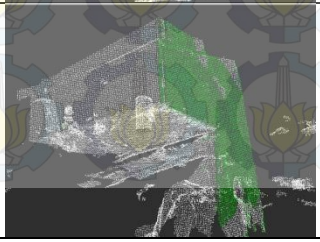
### 4.1.1 Segmentasi Model *Plane*

Jumlah data *point cloud* dari masukan *singleframe* dapat berbeda antar *file* Polygon. Kondisi ini dapat menyebabkan waktu tempuh dari proses segmentasi dan visualisasi berbeda-beda. Data Polygon yang diolah adalah bagian *vertex list* dengan elemen *x, y,* dan *z*. Pada pengujian tahap pertama dilakukan proses segmentasi model *plane* dengan visualisasi menggunakan Point Cloud Viewer. Pengujian ini bertujuan untuk mengetahui waktu tempuh dari proses segmentasi dengan jumlah *point* yang beragam. Parameter kontrol yang diatur pada pengujian tahap pertama sebagai berikut,

a. Jenis model : Normal Plane
b. Jumlah sampel : 3 *point*
c. *Threshold* model : 0,5
d. Probabilitas ditemukannya model : 0,99
e. Literasi maksimum : 1000
f. Visualisasi : Point Cloud Viewer
g. Data *Inliers* : *Point* berwarna hijau
h. Data *Outliers* : *Point* berwarna putih

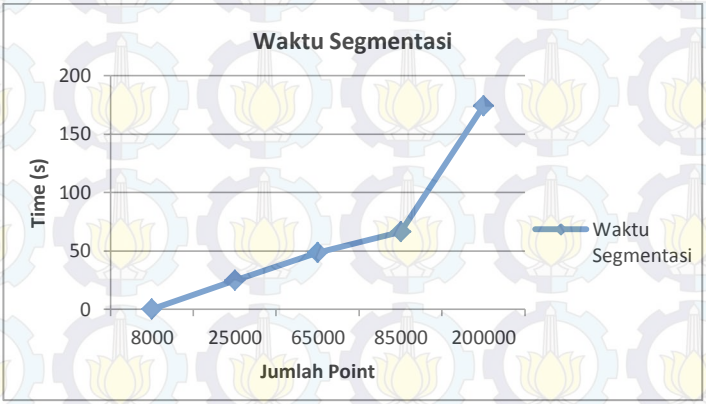
**Tabel 4.2** Hasil pengujian data masukan *singleframe* dengan variasi jumlah point

| No | Informasi | Hasil segmentasi |
|---|---|---|
| 1 | *Point*: 7830<br>*Trial*: 16<br>*Best Inliers*: 5000 (63,85%)<br>*Segmentation Time*(s) : 0,23<br>*Gameobject*:1 |  |

| No | Informasi | Hasil segmentasi |
|---|---|---|
| 2 | *Point*: 245130<br>*Trial*: 325<br>*Best Inliers*: 59208 (24,15%)<br>*Segmentation Time*(s) : 25,02<br>*Gameobject*:4 |  |
| 3 | *Point*: 645711<br>*Trial*: 232<br>*Best Inliers*: 174484 (27,02%)<br>*Segmentation Time*(s) : 48,47<br>*Gameobject*: 10 |  |
| 4 | *Point*: 838071<br>*Trial*: 247<br>*Best Inliers*: 221564 (26,43%)<br>*Segmentation Time*(s) : 66,56<br>*Gameobject*: 13 |  |
| 5 | *Point*: 1986735<br>*Trial*: 270<br>*Best Inliers*: 510414 (25,96%)<br>*Segmentation Time*(s) : 174,36<br>*Gameobject*: 31 |  |
| 6 | *Point*: 263934<br>*Trial*:314<br>*Best Inliers*: 64455 (24,42%)<br>*Segmentation Time*(s) : 25,54<br>*Gameobject*:5 |  |

Pada tabel 4.2, kolom informasi memuat informasi dari data masukan, dimana *point* adalah jumlah *point*, *trial* merupakan jumlah hipotesa persamaan yang diuji dalam mendapatkan koefisien model atau persamaan model terbaik, *best inliers* adalah jumlah *inliers* yang diperoleh, *segmentation* time merupakan waktu tempuh dari proses segmentasi dalam satuan detik, dan *gameobject* memuat jumlah *gameobject* yang terbentuk pada visualisasi dengan Point Cloud Viewer.

Berdasarkan hasil pengujian diatas, ditemukan ada dua faktor yang mempengaruhi waktu dari proses segmentasi. Yang pertama adalah jumlah masukan data *point cloud*. Pada grafik garis pada gambar 4.1, data grafik diperoleh dari hasil pengujian pada tabel 4.2 baris pertama hingga kelima, terlihat bahwa semakin banyak data masukan mengakibatkan waktu segmentasi yang semakin lama. Hal ini dikarenakan proses *voting* yang berjalan pada metoda Ransac akan menguji setiap hipotesa persamaan terhadap semua masukan *point*. Proses *voting* dilakukan untuk mendapatkan koefisien terbaik dalam proses segmentasi.

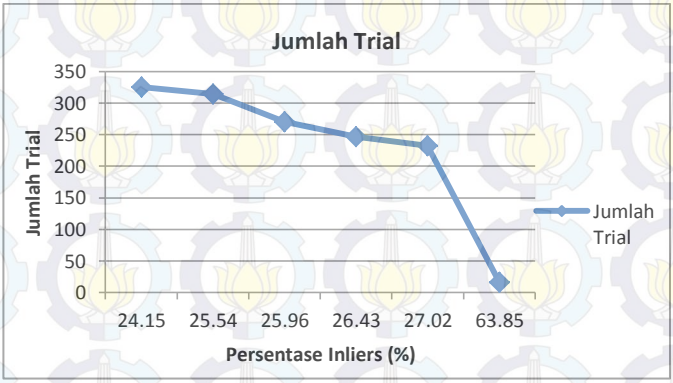



**Gambar 4.1** Grafik pengaruh jumlah *point* terhadap waktu segmentasi

Kedua adalah presentase *inliers* pada data masukan, hal ini dibuktikan berdasarkan kolom informasi pada baris kedua dan keenam tabel 4.2, dimana waktu proses segmentasi data masukan dengan perbedaan jumlah *point* sekitar 20 ribu *point* memiliki waktu segmentasi

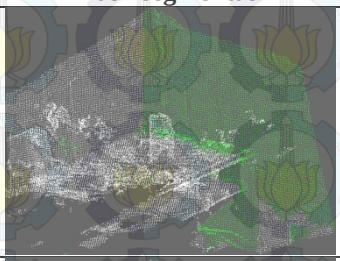
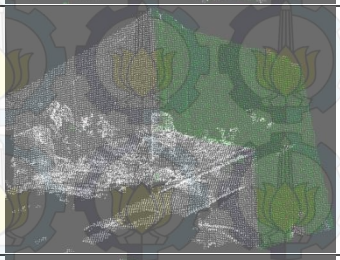
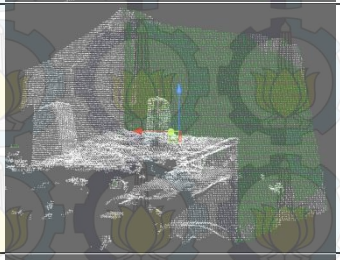
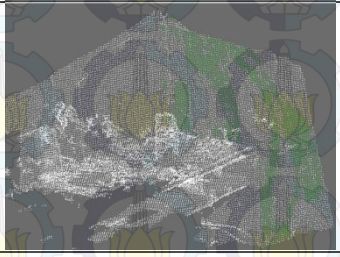
yang hampir sama. Kondisi tersebut dipengaruhi oleh jumlah *trial* pada hipotesa persamaan yang lebih sedikit walaupun memiliki jumlah *point* yang lebih banyak. Pada gambar 4.2, dengan jumlah masukan *point* yang berbeda, dapat ditarik kesimpulan bahwa semakin besar persentase *inliers* mengakibatkan semakin sedikit proses *trial* yang dikerjakan.

Selain kedua faktor tersebut, waktu proses segmentasi juga dipengaruhi oleh nilai *threshold*. Dari hasil pengujian yang lain dengan menggunakan data masukan berjumlah 245130 *point* didapatkan *time* total proses segmentasi dan visualisasi yang semakin besar saat nilai *threshold* diperkecil. Nilai *threshold* yang diatur dalam parameter awal proses segmentasi mempengaruhi presentase *inliers*. Sesuai dengan faktor kedua sebelumnya, presentase *inliers* mempengaruhi jumlah *trial* hipotesa persamaan dalam proses Ransac. Pengaturan nilai *threshold* yang tidak sesuai juga mengakibatkan hasil segmentasi yang tidak baik. Pengaruh tersebut dapat terlihat dari visualisasi data *inliers* pada kolom hasil segmentasi tabel 4.3. Berdasarkan hasil pada tabel tersebut, nilai *threshold* terbaik berkisar 0,5 hingga 1. Jika nilai *threshold* yang diatur lebih besar, maka terdapat bagian dari model lain yang dideteksi sebagai *inliers*. Saat nilai *threshold* lebih kecil, akan didapatkan bagian model yang dicari tidak termasuk kedalam *inliers*.




**Gambar 4.2** Grafik pengaruh persetase *inliers* terhadap jumlah *trial*

**Tabel 4.3** Hasil pengujian masukan *singleframe* dengan variasi *threshold*

| No | Informasi | Hasil segmentasi |
|---|---|---|
| 1 | *Threshold*: 5<br>*Trial*: 110<br>*Best Inliers*: 84547 (34,49%)<br>*Segmentation Time*(s) : 10,15<br>*Gameobject*:4 |  |
| 2 | *Threshold*: 1<br>*Trial*: 240<br>*Best Inliers*: 65428 (26,69%)<br>*Segmentation Time*(s) : 19,7<br>*Gameobject*:4 |  |
| 3 | *Threshold*: 0,5<br>*Trial*: 325<br>*Best Inliers*: 59208 (24,15%)<br>*Segmentation Time*(s) : 25.02<br>*Gameobject*:4 |  |
| 4 | *Threshold*: 0,1<br>*Trial*: 4561<br>*Best Inliers*: 24589 (18,6%)<br>*Segmentation Time*(s) : 346,52<br>*Gameobject*:4 |  |

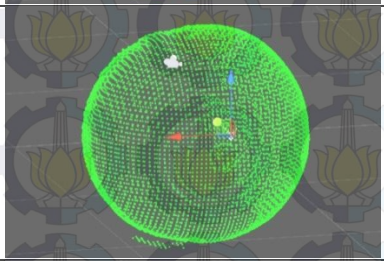
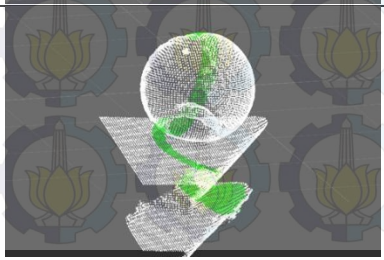
.

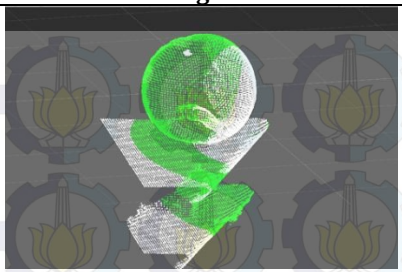
### 4.1.2 Segmentasi Model *Sphere*

Pengujian segmentasi dari model *sphere* dilakukan dengan parameter-parameter sebagai berikut,

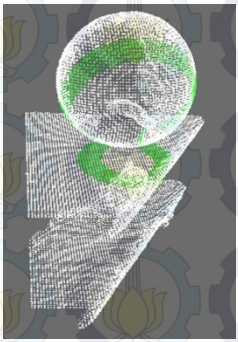
a. *Min Radius* : 0
b. *Max Radius* : 2
c. *Probability* : 0,5
d. Warna *inliers* : hijau
e. Warna *outliers* : putih
f. Visualisasi secara sinkronous dengan *Particle System*

**Tabel 4.4** Hasil segmentasi model *sphere*

| No | Informasi | *Threshold* | Hasil Segmentasi |
|---|---|---|---|
| 1 | *Point*: 55880<br>*Segmentation Time*(s) : 0,82<br>*Best Inliers*: 55753 (99,77%) | 0,01 |  |
| 2 | *Point*: 55880<br>*Segmentation Time*(s) : 0,81<br>*Best Inliers*: 55880 (100%) | 0,05 |  |
| 3 | *Point*: 132952<br>*Segmentation Time*(s) : 1,88<br>*Best Inliers*: 14695 (11,05%) | 0,01 |  |

| No | Informasi | *Threshold* | Hasil Segmentasi |
|---|---|---|---|
| 4 | *Point*: 132952<br>*Segmentation Time*(s) : 1,86<br>*Best Inliers*: 44798 (35,95%) | 0,05 |  |

Berdasarkan hasil pada tabel 4.4, waktu proses segmentasi dari model *sphere* dipengaruhi pula dengan jumlah *point*, presentase *inliers*, dan nilai *threshold*. Pada data masukan berupa objek gabungan, pada baris ketiga dan keempat digabungkan dengan model *plane*, objek lain tersebut masih terdeteksi sebagai *inliers* dari model *sphere*. Kondisi ini diakibatkan parameter *radius* maksimal yang diatur adalah 2 meter. Pada gambar 4.3, hasil segmentasi model *sphere* dengan parameter *radius* maksimal 0,1 meter mendekat model yang sebenarnya, namun masih ada bagian dari model *plane* yang termasuk *inliers*.



**Gambar 4.3** Segmentasi model *sphere* dengan *radius* maksimum 0,1

### 4.1.3 Segmemtasi Model Cylinder

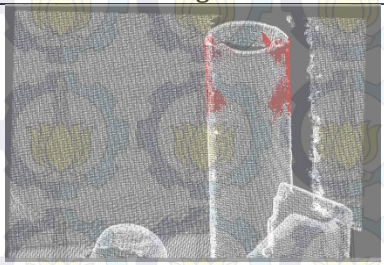
Pada pengujian dari model *cylinder* menggunakan data *point cloud* berjumlah *point* dengan parameter sebagai berikut,

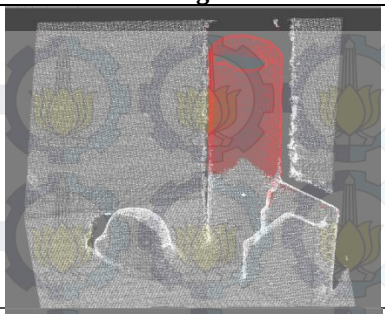
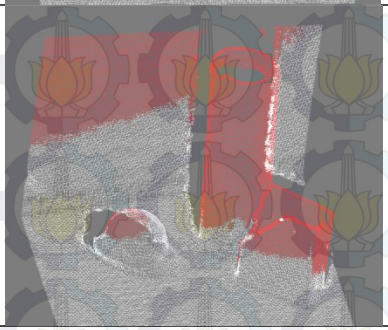
a. *Min Radius* : 0,05

b. *Max Radius* : 0,1
c. *Probability* : 0,9
d. Warna *inliers* : merah
e. Warna *outliers* : putih
f. Visualisasi secara sinkronous dengan *Particle System*
g. *Axis* : 0,5; 0; 0
h. *Viewport* : 0; 0; 0 (*default*)
i. *Eps angle* : 0,1

Pada segmentasi model *cylinder*, dibutuhkan normal vektor dari sampel *point* yang diambil. Parameter *viewport* digunakan dalam mendapatkan normal vektor setelah didapatkan parameter *plane* dari *point* tetangga. Parameter *axis* dan *eps angle* merupakan toleransi dari arah model *cylinder* yang sesuai. Kedua parameter tersebut berkaitan dengan koefisien *axis* model *cylinder* yang didapatkan. *Max radius* dan *min radius* berkaitan dengan jari-jari model. Sama seperti segmentasi model *plane* dan *sphere*, nilai *threshold* mempengaruhi hasil segmentasi dan waktu segmentasi.

**Tabel 4.5** Hasil segmentasi model *cylinder*

| No | Informasi | *Threshold* | Hasil Segmentasi |
|---|---|---|---|
| 1 | *Point*: 931299<br>*Segmentation Time*(s) : 73,365<br>*Best Inliers*: 25168 (2,7%) | 0,1 |  |

| No | Informasi | *Threshold* | Hasil Segmentasi |
|---|---|---|---|
| 2 | *Point*: 931299 *Segmentation Time*(s) : 73,178 *Best Inliers*: 116269 (12,48%) | 0,3 |  |
| 3 | *Point*: 931299 *Segmentation Time*(s) : 72,938 *Best Inliers*: 375479 (40,31%) | 0,5 |  |

Berdasarkan tabel 4.5, dengan jumlah *point* yang sama, nilai *threshold* yang sesuai adalah 0,3, walaupun dengan nilai *threshold* 0,5 bidang cylinder yang didapatkan lebih banyak, namun model lain seperti *plane* yang terdeteksi sebagai *inliers*. Sedangkan pada waktu segmentasi, dari variasi nilai *threshold* tersebut, didapatkan waktu yang hampir sama.

## 4.2 Pengujian Jenis Visualisasi

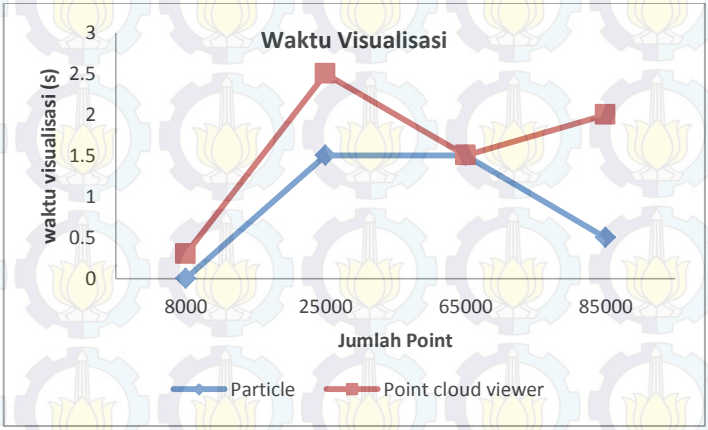
Pada tugas akhir ini disusun rancangan dua buah jenis visualisasi untuk mendapatkan metode terbaik dari proses visualisasi hasil segmentasi objek model. Ada dua jenis visualisasi. Pertama dengan menggunakan *Particle System* dan kedua dengan Point Cloud Viewer. Pada pengujian sebanyak lima kali pada masing jumlah *point* dan jenis visualisasi maka dapat didapatkan bahwa waktu tempuh pada proses segmentasi dengan kedua jenis visualisasi diatas adalah hampir sama,

walaupun visualisasi dengan Point Cloud Viewer melalui proses menyimpan *gameobject* pada *storage*. Hasil pengujian tersebut dapat dilihat pada tabel 4.6.

**Tabel 4.6** Waktu proses segmentasi dan visualisasi

| No | Jumlah *Point* | *Particle (s)* | Point Cloud Viewer (*s*) |
|---|---|---|---|
| 1 | 7830 | 0,26379744 | 0,56103858 |
| 2 | 245130 | 26,41036 | 27,80112 |
| 3 | 645711 | 49,608698 | 49,709412 |
| 4 | 838071 | 66,904486 | 68,651786 |
| 5 | 1986735 | 173,3752 | 174,5148333 |

Apabila dikorelasikan dengan waktu segmentasi pada tabel 4.2 maka didapatkan grafik pengaruh jumlah point terhadap waktu visualisasi dari kedua jenis visualisasi yang digunakan.




**Gambar 4.4** Grafik pengaruh jumlah *point* terhadap waktu visualisasi

Dari grafik pada gambar 4.4, waktu visualisasi dari *Particle System* cenderung lebih cepat daripada Point Cloud Viewer dengan data masukan point 8000 *point* hingga 85000 *point*. Jika diamati hasil visualisasi, maka visualisasi dengan Point Cloud Viewer lebih mudal

untuk melakukan pengamatan terhadap hasil segmentasi karena terdiri dari beberapa *gameobject.*

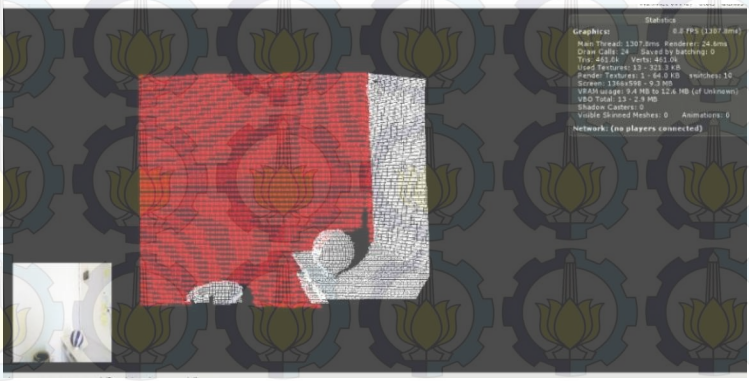
Pada *multiframe*, besar FPS (*frame perseconds*) dari kedua jenis visualisasi ini hampir sama, yakni 0,5 hingga 0,7 pada visualisasi secara sinkronous. Data masukan dari pengujian jenis visualisasi dari *multiframe* sebanyak 76800 *point.*

## 4.3 Pengujian Segmentasi *Point Cloud Multiframe*

Pada pengujian ini dilakukan untuk mengetahui performa dari perangkat lunak yang telah direalisasikan dalam melakukan proses segmentasi dan visualisasi data masukan *point cloud multiframe*.

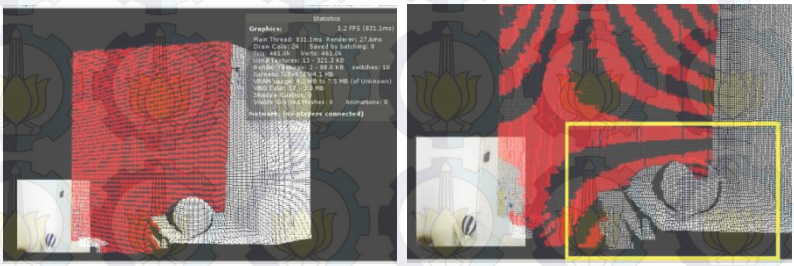
### 4.3.1 Pengujian Data Masukan *Multiframe* secara *Real Time*

Jumlah data *point cloud* dari masukan *multiframe* adalah sama setiap *frame* karena bergantung pada resolusi peta kedalaman yang dihasilkan oleh kamera Kinect. Pada tahap pengujian ini, data *point cloud* yang disegmentasi dan ditampilkan berjumlah 76800 *point*. Setiap *point* tersebut memiliki elemen *x, y,* dan *z*. Lingkungan nyata yang diproses berada pada jarak 1-2 meter dari kamera Kinect dengan obyek yang sama. Pada pengujian dengan visualisasi secara sinkronous, didapatkan nilai FPS antara 0,5 hingga 1,1. Divergensi dari FPS pada pengujian ini diakibatkan waktu proses segmentasi yang berbeda-beda dari tiap *frame* walaupun dengan parameter yang sama.




**Gambar 4.5** Hasil visualisasi secara sinkronous dengan data masukan *multiframe*

Hasil *screen capture* dari visualisasi secara sinkronous dengan data masukan *multiframe* secara *real-time* dapat dilihat pada gambar 4.5. Sedangkan pada pengujian dengan visualisasi secara Asinkronous besar FPS yang didapatkan berkisar antara 0,9 hingga 1,7.



(a) (b)

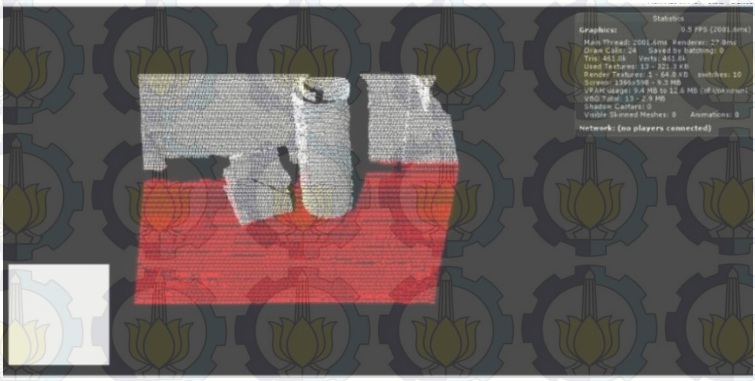
**Gambar 4.6** (a) Hasil visualisasi secara asinkronous dengan data masukan *multiframe*, (b) *Inliers* yang tidak sesuai dengan model

Visualisasi secara asinkronous memiliki FPS yang lebih tinggi daripada sinkronous, namun terdapatkan *inliers* yang tidak sesuai dengan model. Kondisi ini dapat dilihat pada gambar 4.6 (b), dimana terdapat *inliers* yang tidak sesuai dengan model yang diharapkan, berbeda dengan hasil visualisasi pada gambar 4.6 (a). Kondisi ini disebabkan oleh data *inliers* yang ditampilkan masih menggunakan data hasil segmentasi yang lama akibat proses segmentasi dari *frame* tersebut yang belum selesai.

### 4.3.2 Pengujian Data Masukan *Multiframe* Hasil Rekaman

Data rekam yang digunakan pada tahap pengujian ini memiliki 76800 *point* dengan setiap *point* memiliki elemen *x, y,* dan *z*. Pada masukan hasil rekaman, untuk visualisasi secara sinkronous baik dengan Point Cloud Viewer atau *Particles System* didapatkan FPS sebesar 0,5 hingga 0,7 sesuai dengan FPS dengan data masukan secara *stream real-time*. Data yang disimpan dalam *file* binary hanya berisikan data *depth* sehingga tidak terlihat *view color* dari lingkungan yang telah direkam. Kondisi ini dapat dilihat pada gambar 4.7. Pada pengujian ini,

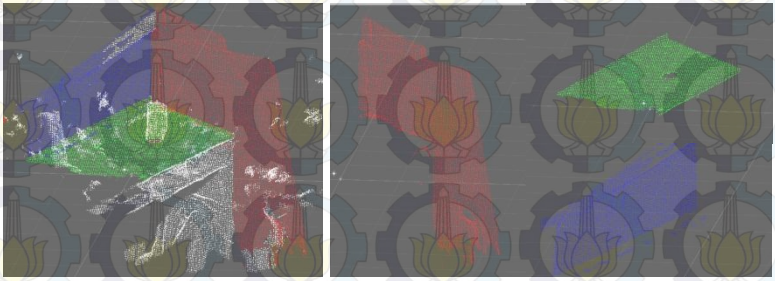
parameter *playback speed* diatur sebesar 0,0333 detik sehingga terdapat beberapa *frame* yang dilewati.



**Gambar 4.7** Hasil visualisasi proses segmentasi data masukan *file* Binary

## 4.4 Pengujian Alur Multisegmentasi

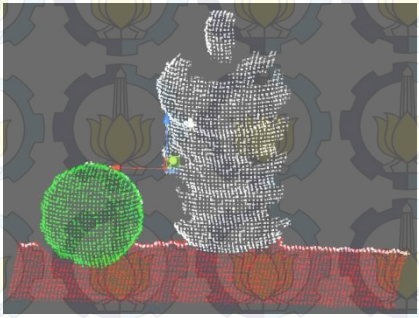
Pengujian alur multisegmentasi pada data masukan *singleframe* dilakukan dengan dua data masukan. Pengujian pertama dilakukan pada data *point cloud* dengan 245130 *point*. Model segmentasi yang digunakan adalah bidang *plane* dengan dilakukan proses segmentasi tiga kali. Besar *threshold* yang digunakan pada seluruh proses segmentasi adalah 0,5 dan besar probabilitas ditemukannya model 0,99.



(a) (b)

**Gambar 4.8** (a) Hasil pengujian multisegmentasi *plane-plane-plane*, (b) Model-model yang tersegmentasi

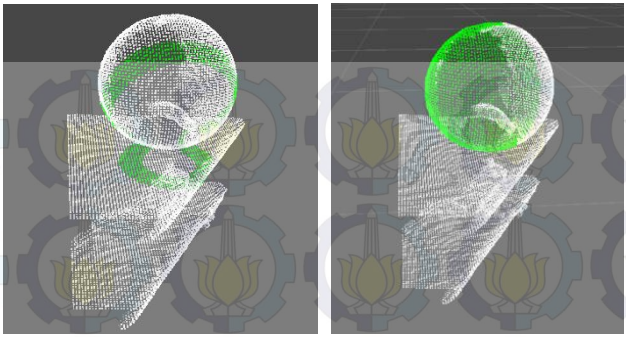
Berdasarkan gambar 4.8 (a), hasil segmentasi model *plane* yang pertama ditandai dengan *point* berwarna merah. Hasil tersebut merupakan model *plane* terbaik dari keseluruhan model *plane* pada data masukan *point cloud*. Warna hijau merupakan hasil segmentasi model plane kedua. Hasil tersebut diperoleh dari proses segmentasi dari data *point cloud* yang merupakan *outliers* dari hasil segmentasi pertama. Warna biru merupakan hasil segmentasi model *plane* yang ketiga. Sedangakan warna putih adalah *outliers* dari proses segmentasi ketiga.



**Gambar 4.9** Hasil pengujian multisegmentasi *plane-sphere*

Pengujian kedua dilakukan dengan data *point cloud* dengan 97893 *point*. Besar *threshold* yang digunakan adalah 0,01 dan probabilitas ditemukan model *plane* 0,99. Sedangkan pada model *sphere*, probabilitas ditemukan model sebesar 0,5 dan *radius* maksimum 0,1. Pada gambar 4.9, model *plane* yang tersegmentasi ditandai warna merah sedangkan model *sphere* ditandai warna hijau. Proses segmentasi model *sphere* dilakukan setelah segmentasi *plane* memberikan hasil yang bagus pada model *sphere*. Hal ini dapat dibandingkan pada gambar 4.10 (a) dan 4.10 (b). Hasil yang kurang memuaskan pada segmentasi bidang *sphere* diakibatkan oleh pengambilan *point* sampel secara acak karena sangat dimungkinkan semua *point* sampel yang diambil dalam satu bidang plane atau dalam satu garis lurus.

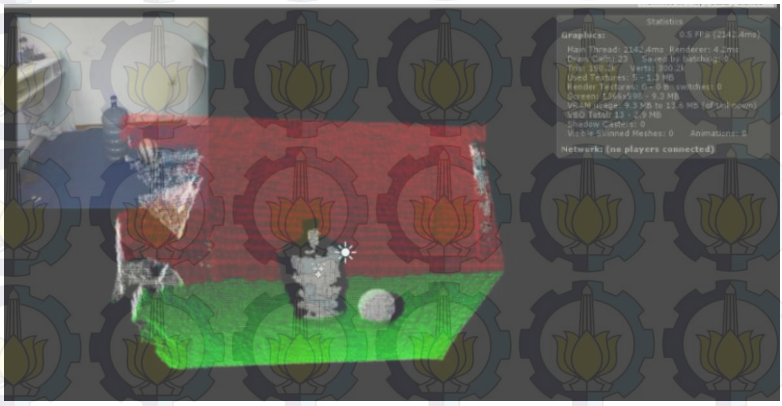
Pada pengujian multisegmentasi pada data masukan *multiframe* didapatkan FPS berkisar antara 0,4 hingga 0,6 *fps*, untuk proses segmentasi *plane-plane*. Sedangkan pada proses segmentasi *plane-plane-plane* didapatkan FPS dibawah 0,2 *fps*.



(a) (b)

**Gambar 4.10** (a) Hasil segmentasi *sphere*, (b) Hasil multisegmentasi *plane-sphere*

Proses multisegmentasi bekerja lebih lambat pada segmentasi pertama karena data *point cloud* yang diolah lebih banyak, proses segmentasi selanjutnya berjalan lebih cepat karena data masukan merupakan *outlier* dari hasil sebelumnya. Jika segmentasi pertama mampu mengeliminasi lebih dari 75% data *point cloud*, maka *fps* yang dihasilkan lebih besar akibat probabilitas *inliers* proses tersebut besar. Hasil *screen capture* dari proses visualisasi alur multisegmentasi dapat dilihat pada gambar 4.11.



**Gambar 4.11** Multisegmentasi pada *stream* Kinect secara *real-time*

# DAFTAR PUSTAKA


[1] Budi, Wahyu Setya. "Raycasting In Three-Dimentional Augmented Reality". Publikasi Tugas Akhir. Surabaya: Institut Teknologi Sepuluh Nopember, 2014.

[2] Rusu, Radu Bogdan. "3D Perception. 50% better, Point Cloud Library". Publikasi Willow Garage, 2010.

[3] Sitek et al. "Tomographic Reconstruction Using an Adaptive Tetrahedral Mesh Defined by a Point Cloud" IEEE Trans. Med. Imag. p.25, 2006.

[4] Tombari, F. D. Holz. "Welcome & Intoduction Point Cloud Library". Prosiding IEEE International Conference on Robotics and Automation (ICRA), 2013.

[5] Rusu, Radu Bogdan, Steve Cousins. "3D is here: Point Cloud Library (PCL)". Prosiding IEEE International Conference on Robotics and Automation (ICRA), 2011.

[6] "Point Cloud Library Documentation". <URL: http://pointclouds.org/documentation/>, Mei, 2015

[7] M.A Fischler, R. C. Bolles. "Random Sample Consensus : A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography". Proseding SRI International, 1981

[8] Ashari, Riska Wahyu. "Fixed Point Augmented Reality Menggunakan Kinect". Publikasi Tugas Akhir. Surabaya: Institut Teknologi Sepuluh Nopember, 2013.

[9] MacCormick, John. "How does the Kinect Work?". Presentasi Dickinson College United States of America.

[10] "Particle Systems". <URL: http://docs.unity3d.com/Manual/ ParticleSystems.html>, Mei, 2015.

[11] Reeves, William T. "Particle Systems: A Technique for Modeling a Class of Fuzzy Objects". ACM Transactions on Graphics, April 1983.

[12] "Particle System". <URL: http://graphics.wikia.com/wiki/ Particle_system>, Mei, 2015.

[13] "Point Cloud Free Viewer". <URL: https://www.assetstore.unity3d.com/en/#!/content/19811>, Mei, 2015.

[14] Izadi, Zahram. “KinectFusion: Real-time 3D Reconstruction and Interaction Using a Moving Depth Camera”. Microsoft Research. Cambridge University, 2011.
[15] “Polygon File Introduction”. <URL: http://paulbourke.net/dataformats/ply/>, Mei, 2015.
[16] “The PLY File Format”. <URL: http://www.cc.gatech.edu/projects/large_models/ply.html>, Mei, 2015.
[17] Weisstein, Eric W. "Plane, MathWorld--A Wolfram Web Resource”. <URL: http://mathworld.wolfram.com/Plane.html>, Mei, 2015.
[18] Schmitt, Stephen R. “Center and Radius of a Sphere from Four Points”. <URL:http://www.abecedarical.com/zenosamples/zs_sphere4pts.html>, Mei, 2015.
[19] Garcia, Sergion. “Fitting Primitive Shapes to Point Clouds for Robotic Grasping”. Publikasi Thesis : KTH Computer Science and Communication, 2009

# BAB 5
# PENUTUP

## 5.1 Kesimpulan

Setelah melalui perancangan aplikasi, implementasi, dan pengujian, akhirnya diperoleh beberapa kesimpulan :

1. Data *point cloud* dapat dipisahkan antara model primitif tiga dimensi satu dengan lain melalui proses segmentasi dengan metode *Random Sample Consensus*. Pada pengujian segmentasi model *plane, sphere,* dan *cylinder*, data *point cloud* dapat dipisahkan antara data *inliers*, data yang sesuai model, dan data *outliers* atau data selain *inliers*.
2. Nilai *threshold*, jumlah *point*, dan presentase *inliers* dalam *point cloud* mempengaruhi waktu segmentasi. Parameter *threshold* juga mempengaruhi kesesuaian hasil segmentasi dengan model yang sebenarnya.
3. Pada pengujian segmentasi tunggal model *plane, frame per second* yang dihasilkan sebesar 0,5 hingga 1,1 pada visualisasi secara sinkronous, sedangkan pada asinkronous diperoleh sebesar *frame per second* 0,9 hingga 1,7.
4. Pada pengujian multisegmentasi model *plane, frame per second* yang dihasilkan lebih kecil daripada segmentasi tunggal model *plane*.

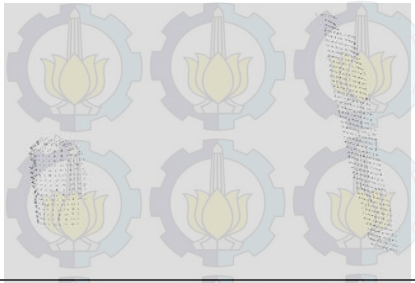
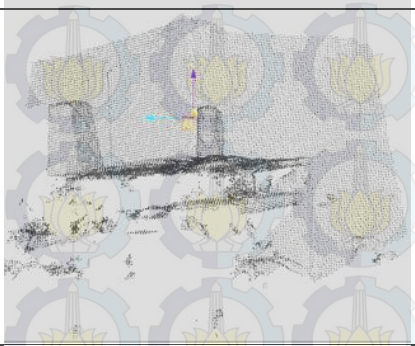
## 5.2 Saran

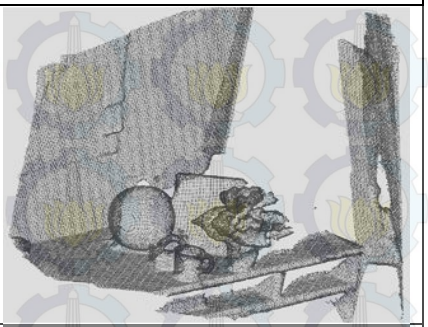
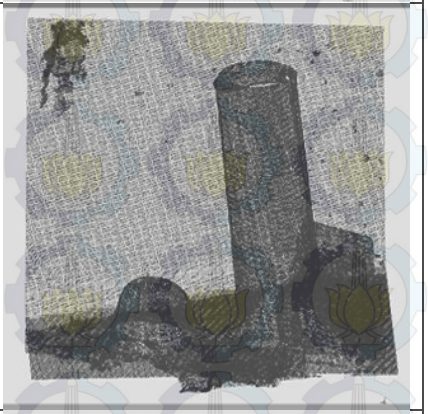
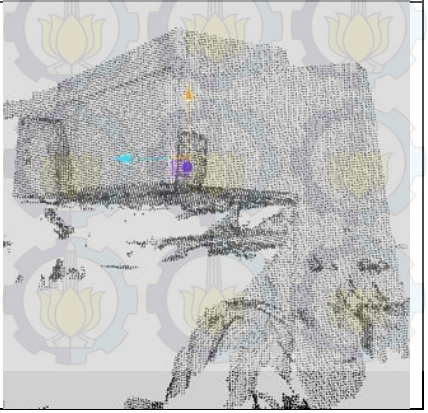
Untuk pengembangan lebih lanjut mengenai tugas akhir ini, penulis memberikan saran:

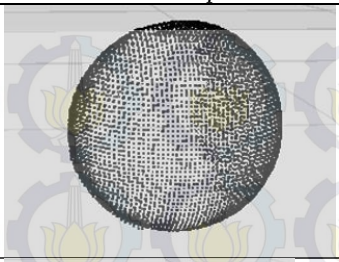
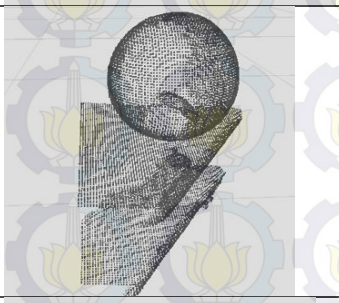
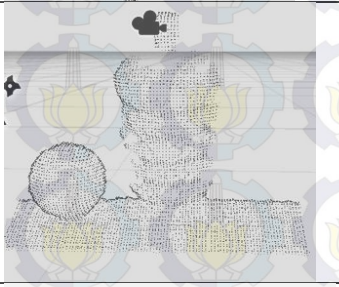
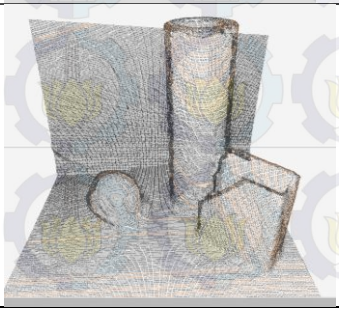
a. Memanfaatkan dan mengembangkan proses segmentasi untuk pengembangan riset tentang *augmented reality* kedepan misalnya proses identifikasi atau pengenalan model atau objek yang lain.
b. Melakukan proses segmentasi dengan perangkat atau sensor yang lain.

# LAMPIRAN

**Visualisasi *file* PLY yang Digunakan dalam Pengujian**

| Model | Jumlah *point cloud* | Visualisasi *point cloud* |
|---|---|---|
| *Plane* | 7830 |  |
| *Plane* | 245130 |  |
| *Plane* | 645711 |  |

| Model | Jumlah *point cloud* | Visualisasi *point cloud* |
|---|---|---|
| *Plane* | 838071 |  |
| *Plane* | 1986735 |  |
| *Plane* | 263934 |  |

| Model | Jumlah *point cloud* | Visualisasi *point cloud* |
|---|---|---|
| *Sphere* | 55880 |  |
| *Sphere* | 132952 |  |
| *Sphere* | 97893 |  |
| *Cylinder* | 931299 |  |

**Hasil Pengujian Segmentasi Model *Plane Singleframe***

**Pengujian 1.** Pengaruh jumlah *point* terhadap waktu segmentasi

*Threshold* : 0.5

Keterangan :
P1 : Pengujian Pertama
P2 : Pengujian Kedua
P3 : Pengujian Ketiga
P4 : Pengujian Keempat
P5 : Pengujian Kelima

| Jumlah Point | P1 | P2 | P3 | P4 | P5 | *Mean* |
|---|---|---|---|---|---|---|
| 7830 | 0.23 | 0.25 | 0.23 | 0.23 | 0.23 | 0.23 |
| 245130 | 24.88 | 25.51 | 24.50 | 25.07 | 25.15 | 25.02 |
| 645711 | 48.68 | 48.73 | 48.35 | 47.94 | 48.65 | 48.47 |
| 838071 | 66.64 | 66.73 | 66.55 | 66.86 | 66.05 | 66.56 |
| 1986735 | 173.72 | 174.72 | 178.29 | 172.78 | 172.29 | 174.36 |
| 263934 | 26.30 | 25.54 | 24.91 | 25.17 | 25.78 | 25.54 |

**Pengujian 2.** Pengaruh *threshold* terhadap waktu segmentasi

*Jumlah Point* : 245130

| *Threshold* | P1 | P2 | P3 | P4 | P5 | *Mean* |
|---|---|---|---|---|---|---|
| 5 | 10.29 | 10.20 | 10.20 | 10.05 | 10.02 | 10.15 |
| 1 | 19.67 | 19.70 | 19.81 | 19.58 | 19.72 | 19.7 |
| 0.5 | 24.88 | 25.51 | 24.50 | 25.07 | 25.15 | 25.02 |
| 0.1 | 353.35 | 343.17 | 340.6 | 342.91 | 352.57 | 346.52 |

**Hasil Pengujian Segmentasi Model *Sphere Singleframe***

**Pengujian 1.** Pengaruh jumlah *point* dan *threshold* terhadap waktu segmentasi

Threshold : 0.01

| Jumlah Point | P1 | P2 | P3 | P4 | P5 | *Mean* |
|---|---|---|---|---|---|---|
| 55880 | 0.81 | 0.81 | 0.81 | 0.82 | 0.83 | 0.82 |
| 132952 | 1.85 | 1.866 | 1.85 | 1.87 | 1.85 | 1.88 |

Threshold : 0.05

| Jumlah Point | P1 | P2 | P3 | P4 | P5 | *Mean* |
|---|---|---|---|---|---|---|
| 55880 | 0.81 | 0.81 | 0.81 | 0.81 | 0.81 | 0.81 |
| 132952 | 1.88 | 1.85 | 1.87 | 1.86 | 1.86 | 1.86 |

**Hasil Pengujian Segmentasi Model *Cylinder Singleframe***

**Pengujian 1.** Pengaruh *threshold* terhadap waktu segmentasi

Jumlah *point* : 931299

| *Threshold* | P1 | P2 | P3 | P4 | P5 | *Mean* |
|---|---|---|---|---|---|---|
| 0.1 | 72.55 | 73.6 | 74.24 | 73.31 | 73.10 | 73.36 |
| 0.3 | 73.74 | 72.58 | 73.30 | 73.22 | 73.03 | 73.17 |
| 0.5 | 73.67 | 73.34 | 72.64 | 72.49 | 72.54 | 72.94 |

**Hasil Pengujian Visualisasi Model *Plane Singleframe***

**Pengujian 1.** Waktu total segmentasi dan visualisasi dari visualisasi dengan Particle System

*Threshold* : 0.5

| Jumlah *Point* | P1 | P2 | P3 | P4 | P5 | *Mean* |
|---|---|---|---|---|---|---|
| 7830 | 0.26 | 0.26 | 0.25 | 0.26 | 0.26 | 0.26 |
| 245130 | 27.06 | 26.14 | 26 | 26.53 | 26.29 | 26.41 |
| 645711 | 51.83 | 48.62 | 50.40 | 48.36 | 48.81 | 49.60 |
| 838071 | 66.81 | 66.06 | 66.59 | 67.50 | 67.55 | 66.90 |
| 1986735 | 178.41 | 169.31 | 170.20 | 175.56 | | 173.37 |

**Pengujian 2.** Waktu total segmentasi dan visualisasi dari visualisasi dengan Point Cloud Viewer

*Threshold* : 0.5

| Jumlah *Point* | P1 | P2 | P3 | P4 | P5 | *Mean* |
|---|---|---|---|---|---|---|
| 7830 | 0.55 | 0.54 | 0.56 | 0.56 | 0.56 | 0.56 |
| 245130 | 28.87 | 27.82 | 27.66 | 27.55 | 27.09 | 27.80 |
| 645711 | 49.33 | 50.17 | 49.38 | 49.87 | 49.78 | 49.70 |
| 838071 | 69.39 | 67.79 | 68.35 | 69.15 | 68.55 | 68.65 |
| 1986735 | 174.58 | 173.96 | 174.99 | 174.75 | | 174.57 |

**FINAL PROJECT – TE 141599**

# *THREE-DIMENSIONAL PRIMITIVE MODEL SEGMENTATION BASED ON POINT CLOUD USING RANDOM SAMPLE CONSENSUS*

Aditya Rifa Utama
NRP 2211100066

Advisors
Dr. Eko Mulyanto Yuniarno, ST., MT.
Christyowidiasmoro, ST., MT.

Department of Electrical Engineering
Faculty of Industrial Technology
Institut Teknologi Sepuluh Nopember
Surabaya 2015

# ABSTRAK

Nama Mahasiswa : Aditya Rifa Utama
Judul Tugas Akhir : Segmentasi Model Primitif Tiga Dimensi Berbasis *Point Cloud* menggunakan *Random Sample Consensus*
Pembimbing : 1. Dr. Eko Mulyanto Yuniarno, ST., MT.
2. Christyowidiasmoro, ST., MT.

Permukaan dari suatu objek nyata dapat direpresentasikan kedalam data *point cloud* dengan setiap *point* memiliki komponen *x, y,* dan *z*. Data *point cloud* dimanfaatkan dalam mengembangkan teknologi *augmented reality* yang lebih interaktif. Dalam penelitian *augmented reality* sebelumnya, data *point cloud* yang diproses belum dipisahkan antar model primitif. Pemisahan antar model diperlukan dalam proses identifikasi objek atau pengolahan *point cloud* lebih lanjut. Tujuan dari tugas akhir ini adalah melakukan segmentasi model primitif tiga dimensi dari data *point cloud,* hasil tangkapan kamera Kinect, dengan menggunakan metode *Random Sample Consensus*. Model primitif yang disegmentasi adalah model *plane*, *sphere*, dan *cylinder*. Dari hasil pengujian, *point cloud* dapat dipisahkan menjadi data *inliers* yaitu data yang sesuai model berdasarkan nilai *threshold*, dan data *outliers* atau selain data *inliers*. Pada segmentasi tunggal bidang *plane* dengan masukan *multiframe* data *point cloud* didapatkan *frame persecond* sebesar 0,5 hingga 1,1 pada visualisasi sinkronous, sedangkan pada visualisasi asinkronous dihasilkan 0,7 hingga 1,9 *fps*. Sedangkan pada multisegmentasi dihasilkan nilai *fps* yang lebih kecil.

Kata kunci : Model primitif, *Point cloud*, *Random Sample Consensus*, Segmentasi

# *ABSTRACT*

*Name* : Aditya Rifa Utama
*Title* : *Three-Dimensional Primitive Model Segmentation Based On Point Cloud Using Random Sample Consensus*
*Advisors* : 1. Dr. Eko Mulyanto Yuniarno, ST., MT.
2. Christyowidiasmoro, ST., MT.

*The surface of real object can be represented in point cloud data with each point containing components x, y, z. Point cloud data is used to make augmented reality more interactive. In latest research of augmented reality, point cloud has not yet been separated from each model to another when it was processed. Separating each object is required in order to identify an object or more complex process. The purpose of this final project is to do three-dimensional primitive model segmentation from point cloud, in which the point cloud comes from Kinect, using Random Sample Consensus. Plane, sphere and cylinder are three-dimensional primitive model that is segmented. From the results, point cloud data can be separated into inliers data or represented data of primitive model based on threshold and outliers. In single-segmentation of plane with multi-frame point cloud input, synchronous visualization generates 0.5-1.1 frame per second. Whereas in asynchronous visualization generates 0.7-1.9 fps. On multi-segmentation process, the fps amounts less than single-segmentation.*

*Keywords: Primitive model, Point cloud, Random Sample Consensus, Segmentation*

# KATA PENGANTAR

Puji dan Syukur kehadirat Tuhan YME atas segala limpahan berkah, serta rahmat-Nya, penulis dapat menyelesaikan tugas akhir ini dengan judul : ***Segmentasi Model Primitif Tiga Dimensi Berbasis Point Cloud Menggunakan Random Sample Consensus***.

Tugas akhir ini disusun dalam rangka pemenuhan bidang riset di Jurusan Teknik Elektro ITS, Bidang Studi Teknik Komputer dan Telematika, serta digunakan sebagai persyaratan menyelesaikan pendidikan S-1. Tugas akhir ini dapat terselesaikan tidak lepas dari bantuan berbagai pihak. Oleh karena itu, penulis mengucapkan terima kasih kepada:

1. Keluarga, Ibu, Bapak dan Saudara yang telah memberikan dorongan spiritual dan material serta seluruh kerabat dan kolega penulis yang banyak membantu proses dalam menyelesaikan buku penelitian ini.
2. Bapak Dr. Tri Arief Sardjono, S.T., M.T. selaku Ketua Jurusan Teknik Elektro, Fakultas Teknologi Industri, Institut Teknologi Sepuluh Nopember
3. Secara khusus penulis mengucapkan terima kasih yang sebesar-besarnya kepada Bapak Dr. Eko Mulyanto Yuniarno, ST., MT. dan Bapak Christyowidiasmoro, ST., MT. atas bimbingan selama mengerjakan tugas akhir.
4. Bapak-ibu dosen pengajar Bidang Studi Teknik Komputer dan Telematika, atas pengajaran, bimbingan, serta perhatian yang diberikan kepada penulis selama ini.
5. Rekan tim Niltava Labs atas fasilitas dan dukungan selama pengerjaan tugas akhir ini.
6. Seluruh teman-teman angkatan e-51 serta teman-teman B201crew Laboratorium Bidang Studi Teknik Komputer dan Telematika.

Kesempurnaan hanya milik Tuhan, untuk itu penulis memohon segenap kritik dan saran yang membangun serta menghatur maaf atas segala kekurangan yang ada dalam penulisan buku ini.

Surabaya, Juli 2015
Penulis

# DAFTAR ISI

# DAFTAR TABEL

# DAFTAR GAMBAR

# BIOGRAFI PENULIS

Aditya Rifa Utama lahir di Kediri pada tanggal 21 Desember 1993. Ia menyelesaikan jenjang Sekolah Dasar di SDN Tengger Lor pada tahun 2005, kemudian melanjutkan pendidikan SMP di SMPN 1 Kunjang dan lulus pada tahun 2008. Pada tahun 2011, penulis menyelesaikan pendidikan SMA di SMAN 2 Pare Kediri. Setelah lulus dari jenjang SMA, penulis melanjutkan pendidikan di Institut Teknologi Sepuluh Nopember dengan mengambil Jurusan Teknik Elektro. Pada semester kelima, penulis mengambil konsentrasi bidang studi Teknik Komputer dan Telematika dan aktif sebagai asisten laboratorium B201. Penulis Selama menempuh pendidikan kuliah, penulis bergabung dalam Himpunan Mahasiswa Teknik Elektro dan mengikuti beberapa kompetisi karya ilmiah seperti Program Kreatifitas Mahasiswa dan Olimpiade Sains Nasional Pertamina.

# BAB 1
# PENDAHULUAN

## 1.1 Latar Belakang

Permukaan dari suatu objek nyata dapat direpresentasikan kedalam data *point cloud* tiga dimensi dengan setiap *point* memiliki komponen *x, y,* dan *z*. Data *point cloud* tersebut diperoleh dari hasil tangkapan kamera kedalaman, seperti kamera Kinect. Data *point cloud* dapat dimanfaatkan dalam membuat model 3D CAD (*Computer-aided Design*), animasi dan *augmented reality*. Pada penelitian sebelumnya dengan judul *Raycasting In Three-Dimentional Augmented Reality*[1], objek *virtual* pada *augmented reality* dapat ditampilkan ke lingkungan nyata dengan mengolah data *point cloud*, namun lokasi objek *virtual* tersebut ditentukan secara manual berdasarkan posisi kamera. Data *point cloud* yang digunakan pada penelitian tersebut belum dapat dipisahkan antar model primitif. Pada pengerjaan tugas akhir ini diharapkan mampu melakukan segmentasi model primitif pada data *point cloud* sehingga dapat memisahkan antar model dan menampilkan hasil segmentasi pada layar untuk mengetahui model yang telah terpisah. Proses segmentasi yang telah diimplementasikan dapat dimanfaatkan sebagai tahap *pre-processing* dalam pengolahan data *point cloud* atau referensi dalam pengembangan teknologi *augmented reality* yang lebih interaktif.

## 1.2 Permasalahan

Pada penelitian teknologi *augmented reality*, data *point cloud* dari hasil tangkapan kamera kedalaman belum dapat dipisahkan antar model primitif, sehingga tidak bisa dibedakan antara model satu dengan model yang lain.

## 1.3 Tujuan

Tujuan dari pengerjaan tugas akhir ini adalah melakukan segmentasi model primitif pada data *point cloud* tiga dimensi dari hasil tangkapan kamera kedalaman.

## 1.4 Batasan Masalah

Dalam pengerjaan tugas akhir ini, diberikan beberapa batasan masalah, diantaranya sebagai berikut:

1. Lingkungan nyata berupa lingkungan yang sudah diatur sebelumnya.
2. Model primitif yang disegmentasi adalah *plane*, *sphere* dan *cylinder*.
3. Kamera yang digunakan adalah kamera Kinect.
4. Target platform produk adalah komputer dengan sistem operasi Windows.

## 1.5 Sistematika Penulisan

Laporan penelitian tugas akhir ini tersusun dalam sistematika dan terstruktur sehingga lebih mudah dipahami dan dipelajari oleh pembaca maupun seseorang yang hendak melanjutkan penelitian ini. Alur sistematika penulisan laporan penelitian ini yaitu :

1. Bab I Pendahuluan
   Bab ini berisi uraian tentang latar belakang, permasalahan, tujuan, metodologi, sistematika laporan dan relevansi.
2. Bab II Tinjauan Pustaka
   Pada bab ini berisi tentang uraian secara sistematis teori-teori yang berhubungan dengan permasalahan yang dibahas pada tugas akhir ini. Teori-teori ini digunakan sebagai dasar dalam tugas akhir, yaitu pemodelan objek primitif, metode *Random Sample Consensus*, *point cloud*, dan teori-teori penunjang lainya.
3. Bab III Desain dan Implementasi Sistem
   Bab ini berisi tentang penjelasan terkait sistem yang dibuat. Guna mendukung itu digunakanlah diagram alir, *flowchart,* dan *pseudocode* agar sistem mudah dipahami dan diimplementasikan.
4. Bab IV Pengujian dan Analisa
   Bab ini menjelaskan tentang pengujian yang dilakukan terhadap sistem dalam penelitian ini dan menganalisa sistem. Spesifikasi perangkat keras dan perangkat lunak yang digunakan juga disebutkan dalam bab ini. Tujuannya adalah sebagai variabel kontrol dari pengujian yang dilakukan.

5. Bab V Penutup
   Bab ini merupakan penutup yang berisi kesimpulan yang diambil dari penelitian dan pengujian yang telah dilakukan. Saran dan kritik yang membangun untuk pengembangan lebih lanjut juga dituliskan pada bab ini.

## 1.6 Relevansi

Penelitian ini memiliki relevansi dengan penelitian-penelitian tentang *augmented reality* sebelumnya dan pengembangan metode segmentasi data *point cloud* yang telah dilakukan oleh pengembang pustaka Point Cloud Library. Penelitian ini juga berkaitan dengan pengembangan aplikatif kamera Kinect pada *Unity3d.*

# BAB 2
# TINJAUAN PUSTAKA

Untuk mendukung penelitian dalam tugas akhir ini, dibutuhkan beberapa teori penunjang sebagai bahan acuan dan referensi. Dengan demikian penelitian ini lebih terarah.

## 2.1 *Point Cloud*

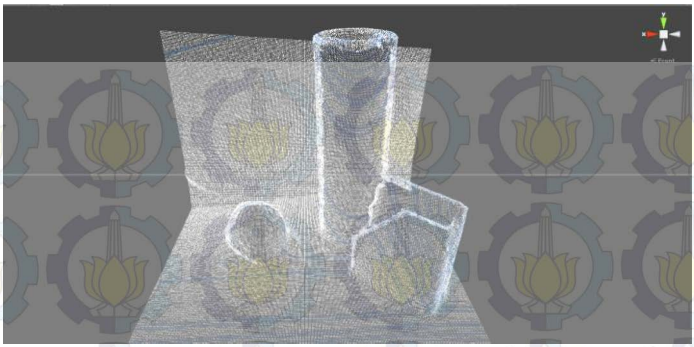
*Point cloud* adalah struktur data yang merepresentasikan kumpulan dari *multi-dimentional points* dan sering digunakan untuk mendeskripsikan data pada objek tiga dimensi. Pada *point cloud* tiga dimensi dari sebuah objek, setiap *point* mewakili nilai koordinat geometri *x, y,* dan *z* dari permukaan objek tersebut.

$$p_1 = \{x_1, y_1, z_1\} \rightarrow P = \{p_1, p_2, p_3, \dots, p_n\} \quad (2.1)$$

Selain data XYZ, setiap *point* dapat memuat informasi tambahan yang lain, seperti spektrum warna, besar intensitas, jarak, dan besar vektor normal. Setiap dimensi data dari *point cloud* disimpan dalam berbagai format untuk mengurangi proses komputasi dan ruang penyimpanan, seperti data XYZ yang disimpan dalam *format Polygon*.

*Point cloud* sangat bermanfaat dalam berbagai kebutuhan. Selain digunakan untuk merepresentasikan geometri dari suatu objek, *point cloud* dapat melengkapi dan menggantikan gambar ketika data gambar memiliki dimensi yang tinggi [2], membuat model 3D CAD (*Computer-aided Design*), metrologi atau inspeksi kualitas, aplikasi kustomisasi visualisasi, animasi dan pengembangan *augmented reality*.

Ada berbagai macam sensor yang dapat menghasilkan informasi tiga dimensi dari suatu objek, seperti laser atau sensor Lidar, kamera *stereo, Time of Flight Cameras*. Setiap sensor memiliki metode tersendiri dalam mendapatkan data *point cloud*. Metode *triagulasi* merupakan metode yang sering digunakan untuk mendapatkan data *point cloud* dengan mendapatkan persamaan dari hasil citra pada dua buah kamera *stereo* dan arah pantulan sinar laser, seperti yang diterapkan pada kamera Kinect. Gambar 2.1 merupakan contoh visualisasi data *point cloud* dari hasil tangkapan kamera Kinect.



**Gambar 2.1** Visualisasi *point cloud* tiga dimensi

Pada umumnya, data *point cloud* tidak dapat langsung digunakan oleh sebagian besar aplikasi 3D, oleh karena itu biasanya dikonversi ke *polygon* atau model segitiga *mesh*, model *NURBS surface*, atau model *CAD* dengan melalui proses yang umum disebut sebagai rekonstruksi permukaan. Ada banyak teknik untuk mengubah *point cloud* ke permukaan tiga dimensi, seperti *Alpha Shapes* dan *Ball Pivoting* [1].

Salah satu aplikasi di mana *point cloud* secara langsung dapat digunakan adalah pada metrologi industri atau inspeksi. Pada hasil produksi, *point cloud* dapat disesuaikan dengan model *CAD* (atau bahkan *point cloud* lainnya), dan dibandingkan untuk mengetahui perbedaannya. Perbedaan ini dapat ditampilkan sebagai *color maps* yang memberikan indikator visual dari deviasi antara bagian produksi dan model *CAD*. Dimensi geometris dan toleransi juga dapat diperoleh langsung dari *point cloud* [3].

## 2.2 Point Cloud Library

Point Cloud Library merupakan *open source framework* untuk pengolahan data *point cloud*. Point Cloud Library berada dibawah lisensi dari BSD (*Barkeley Software Distribution*) dan didukung lebih dari 450 *developer* atau kontibutor [4]. Point Cloud Library berisikan berbagai algoritma yang berkaitan dengan pengolahan citra dua dimensi dan tiga dimensi, seperti *filtering*, estimasi fitur, rekonstruksi permukaan objek, registrasi, *model fitting*, dan segmentasi objek.

Point Cloud Library dapat dijalankan dalam berbagai sistem operasi yang berbeda atau *cross-platform* yakni pada Linux, MacOs, Windows, Android/iOS. Point Cloud Library telah diintegrasikan secara penuh dengan ROS (*Robot Operating System*) dan telah digunakan dalam berbagai proyek dalam komunitas robot. Selain itu, Point Cloud Library mampu mendukung OpenMP dan Intel Threading Building Blocks (TBB) *library* dalam *multi-core parallelization* [5]. Dasar alur proses pada *point cloud library* sebagai berikut,

1. Membuat modul pengolahan, misalkan *filter*, estimasi fitur, atau segmentasi,
2. Menggunakan *setInputCloud* untuk mendapatkan masukan data *point cloud* pada modul pengolahan,
3. Mengatur beberapa parameter sesuai modul pengolahan,
4. Memanggil fungsi *compute, filter, segment* atau fungsi lain yang sejenis untuk mendapatkan hasil.

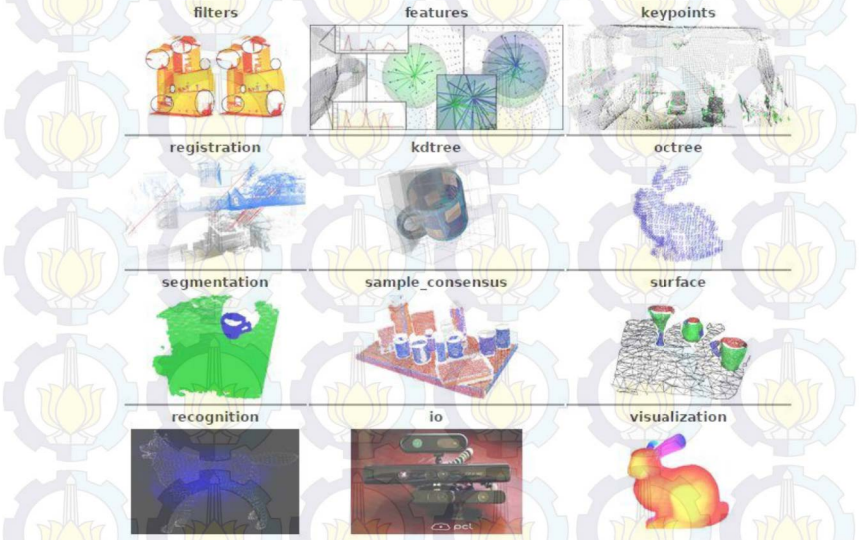
Point Cloud Library dibagi menjadi beberapa seri pustaka yang lebih kecil untuk mempermudah pengembangan dan dapat digunakan secara terpisah. Fitur-fitur dari Point Cloud Library dapat dilihat pada gambar 2.2 dimana dibagi menjadi beberapa fitur, yakni,

a. *Libpcl_filters*, pustaka yang mengimplementasikan metode *filter* data seperti *downsampling*, seleksi *outlier*, ekstrasi indek, dan proyeksi.
b. *Libpcl_features*, pustaka yang mengimplementasikan pengolahan fitur 3D seperti vektor normal dan kelengkungan dari permukaan, estimasi titik batas, dan *moment invariants.*
*c*. *Libpcl_io*, pustaka yang mengimplementasikan operasi I/O seperti menulis dan membaca *file* PCD (*Point Cloud Data*).
d. *Libpcl_segmentation*, pustaka yang mengimplementasikan metode ekstrasi *cluster* dan *model fitting* dengan metode *sample consensus* untuk berbagai model parametrik.
e. *Libpcl_surface*, pustaka yang mengimplementasikan teknik rekonstruksi permukaan, *meshing*, dan *moving least squares*.
f. *Libpcl_registration*, pustaka yang mengimplementasikan metode registrasi data *point cloud,* seperti ICP.
g. *Libpcl_keypoints*, pustaka yang mengimplementasikan metode ekstrasi perbedaan *keypoint* yang dapat digunakan sebagai langkah *preprocessing*.

h. *Libpcl_range_images*, pustaka yang mengimplementasikan pembuatan gambar dengan data *point cloud*.

Point Cloud Library dilengkapi dengan pustaka visualisasi sendiri berbasis VTK (The Visualization Toolkit). VTK menawarkan dukungan *multi-platform* untuk proses *rendering* 3D data *point cloud* dan permukaan termasuk dukungan untuk visualisasi tensor, tekstur, dan metode *volumetric*. Pada versi 0.2, pustaka *visualization* meliputi,

a. Metode untuk rendering dan pengaturan properti visual seperti warna, ukuran dan *opacity* pada setiap *point*.
b. Motode untuk menggambar bentuk 3D dasar di layar seperti *cylinder*, bola, garis, dan poligon baik dari dataset *point* atau persamaan parametrik.
c. Modul visualisasi *histogram* untuk plot 2D.
d. *Multitude* dari geometri dan warna pada koordinat kartesian tiga dimensi.
e. Modul visualisasi untuk *range image*.




**Gambar 2.2** Fitur-fitur pada Point Cloud Library [6]

## 2.3 *Random Sample Consensus*

Salah satu metode dalam proses segmentasi *point cloud* pada Point Cloud Library dengan menggunakan metode Ransac (*Random Sample Consensus*). Ransac merupakan algoritma acak yang digunakan untuk melakukan pendekatan dalam pemodelan objek. Ransac dapat menafsirkan dan merapikan data yang memiliki presentasi *gross error* yang signifikan. Metode ini cocok untuk diaplikasikan dalam analisis citra secara otomatis. Metode *Random Sample Consensus* sering digunakan untuk memecahkan masalah penetapan lokasi atau LDP (*Location Determination Problem*), dimana tujuannya adalah untuk menentukan *point* dalam ruang yang memproyeksikan gambar kedalam satu set *landmark* dengan lokasi yang diketahui [7].

Asumsi dasar dari metode *Random Sample Consensus* adalah bahwa data terdiri dari *inliers*, yaitu data yang distribusinya dapat dijelaskan oleh beberapa set dari parameter model, dan *outliers*, yaitu data yang tidak sesuai dengan model. *Outliers* dapat muncul dikarenakan nilai *noise* yang ekstrim, kesalahan pengukuran, atau hipotesa yang salah tentang interpretasi data.

Algoritma dari Ransac adalah teknik *learning* untuk memperkirakan parameter dari model dengan cara mengambil data *sampling* dari data yang diamati secara acak. Skema *voting* digunakan untuk mendapatkan hasil terbaik karena adanya *inliers* dan *outliers*. Elemen data pada *dataset* yang digunakan untuk memilih satu atau beberapa model. Implementasi dari skema *voting* tersebut didasarkan pada dua asumsi. Asumsi yang pertama adalah fitur *noise* tidak akan memilih secara konsisten untuk setiap model tunggal. Asumsi kedua adalah jumlah fitur *inliers* yang cukup untuk menentukan model terbaik. Algoritma Ransac pada dasarnya terdiri dari dua langkah yang iteratif diulang,

1. *Subset* sampel minimal memiliki data item yang dipilih secara acak dari data masukan. Parameter model didapatkan berdasarkan data *subset* sampel tersebut.
2. Algoritma mencari elemen pada seluruh data masukan yang konsisten dengan parameter model yang didapatkan pada langkah pertama. Sebuah elemen akan dianggap *outliers* jika tidak sesuai dengan parameter model diluar ambang batas kesalahan.

Dua langkah pada algoritma Ransac akan diulang sampai didapatkannya cukup data *inliers* atau sampai batas jumlah literasi yang ditentukan. Jumlah literasi dari metode Ransac dapat ditentukan dengan persamaan 2.2, dimana $k$ adalah jumlah literasi, $p$ adalah probabilitas algoritma menemukan hasil terbaik, $w$ adalah probabilitas didapatkannya *inliers* setiap *point* dipilih, $n$ adalah jumlah minimum data sampel untuk mendapatkan parameter model.

$$k = \frac{\log(1-p)}{\log(1-w/n)} \tag{2.2}$$

Kelebihan dari metode Ransac adalah kemampuan untuk melakukan *robust estimation* terhadap parameter model. Ransac dapat mengestimasikan parameter model dengan akurasi yang tinggi walaupun terdapat banyak *outliers* pada data masukan. Sedangkan kelemahannya adalah tidak adanya konsistensi waktu dalam menetukan parameter dari model.

## 2.4 Kamera Kinect

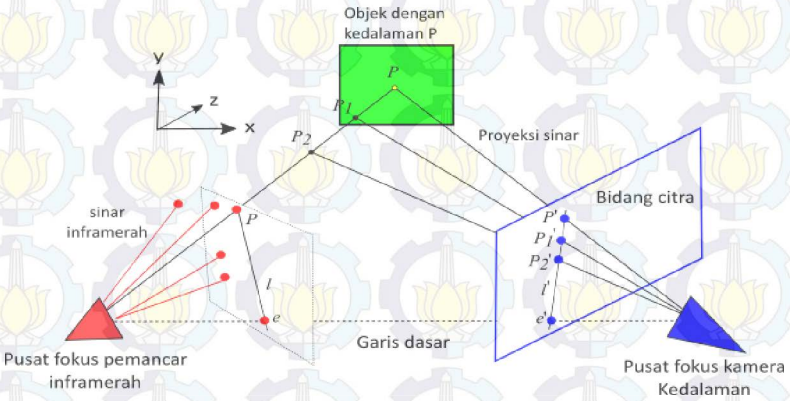
Kinect merupakan kamera *stereovision*, dimana untuk proses pengambilan citra menggunakan dua arah sudut pandang yang berbeda. Kinect memiliki dua buah kamera utama, yaitu kamera *depth* dan kamera RGB, dan sebuah pemancar *inframerah* [1]. Kamera *depth* digunakan untuk mengetahui jarak kedalaman objek dari kamera, yaitu dengan menangkap *inframe*rah yang dipancarkan oleh *IR Transmitter* dan dipantulkan kembali oleh objek. Sedangkan kamera RGB digunakan untuk mengetahui bentuk tekstur atau permukaan dari objek.

Kamera *depth* Kinect memiliki jarak optimal yang harus dipenuhi ketika melakukan proses penangkapan citra yaitu 1.2 meter sampai 3.5 meter. Apabila jarak yang digunakan melebihi atau kurang dari jarak tersebut, maka objek tidak akan diketahui jaraknya oleh kamera *depth* sehingga jarak kedalaman yang dihasilkan bernilai 0.

Pada kamera Kinect terdapat *microphone array* yang digunakan untuk merekam atau menginputkan suara. Kinect juga dilengkapi dengan *tilt* motor dimana dapat digunakan untuk mengatur sudut pandang kamera, sehingga area yang bisa ditangkap oleh kamera menjadi luas. Pengaturan sudut pandang kamera dengan jangkauan sudut kurang lebih 27 derajat dapat menggunakan program tertentu [8].

Kamera RGB memiliki resolusi yang lebih luas apabila dibandingkan dengan kamera *depth*, dimana hasil dari kamera tersebut memiliki resolusi maksimal 640x480 piksel dengan rentang 30 fps berupa informasi citra RGB. Sedangkan kamera *depth* memiliki resolusi maksimal sebesar 320x240 piksel dengan rentang 30 fps.

Pemancar *inframerah* berfungsi untuk memancarkan titik-titik *inframerah* dengan pola yang tidak teratur dan memiliki intensitas yang acak. Sinar *inframerah* yang dipancarkan tersebut akan mengenai objek yang berada didepan kamera dan dipantulkan kembali sehingga dapat dikenali secara cepat dan tepat oleh kamera *depth*. Pada kamera Kinect, posisi objek diketahui melalui dua proses. Pertama, menghitung kedalaman area dengan menganalisa partikel sinar *inframerah* yang tersebar pada area atau *structured light*. Kedua melalui pendekatan lokasi melalui metode *machine learning* dari hasil citra yang didapatkan [9]. Ilustrasi dari proses pancaran dari sinar inframerah dari pemancar menuju sensor kedalaman kamera Kinect dapat dilihat pada gambar 2.3.




**Gambar 2.3** Pancaran sinar *inframerah* kamera Kinect

Data *depth* bisa ditampilkan pada *depth image* dalam beberapa macam warna sesuai yang diinginkan. Warna tersebut untuk mengetahui tingkat jarak kedalaman suatu objek terhadap posisi dari kamera Kinect. Untuk jarak objek yang dekat dengan kamera, warna yang ditampilkan semakin gelap dan apabila semakin jauh jarak objek warna yang ditampilkan semakin terang hingga batas jarak pandang

kamera paling jauh. Kamera Kinect memiliki jarak pandang minimum dan jarak pandang maksimum, sehingga objek yang berada diluar batas jarak tersebut tidak memiliki data *depth*. Demikian juga untuk objek yang tidak bisa memantulkan pancaran dari *inframerah* yang mengakibatkan kamera *depth* tidak menerima data dari objek tersebut sehingga tidak ada data yang diterima[8].

## 2.5 Unity3D

Unity3D merupakan *Integrated Development Environment* (IDE) yang digunakan dalam tugas akhir ini. IDE ini didukung beberapa *plugin* atau *third-party software* yang memudahkan dalam proses perancangan suatu aplikasi.

### 2.5.1 Microsoft Kinect Unity SDK

Microsoft Kinect Unity SDK berisikan pustaka dasar yang mengakses fungsi-fungsi pada kamera Kinect. Pustaka ini menerjemahkan *file* DLL (*Dynamic-link library*) dari Microsoft Kinect kedalam fungsi-fungsi dasar yang dapat diterapkan pada Unity3D, seperti *stream depth data* dan *stream color data*.

### 2.5.2 Particle System

Pada Unity3D, *particles* merupakan gambar atau *mesh* kecil dan sederhana yang ditampilkan serta digerakkan dalam jumlah besar oleh *Particle System* [10]. Sedangkan *Particle System* sendiri merupakan teknik dalam komputer grafis dan *game physics* yang menggunakan banyak *sprites,* gambar dua dimensi kecil, atau objek lain untuk mensimulasikan beberapa jenis fenomena *fuzzy* yang sangat sulit untuk dilakukan dengan teknik rendering biasa. Fenomena tersebut bisa berupa proses reaksi kimia atau fenomena alam seperti api, ledakan, asap, air terjun, dan awan [11].

Pada dasarnya, posisi dan pergerakan dari *Particle System* dikontrol oleh *emmiter*. *Emmiter* bertindak sebagai sumber dari partikel dan lokasinya di ruang 3D menentukan lokasi partikel akan ditampilkan. Geometri dasar 3D seperti kubus dan bidang datar dapat digunakan sebagai *emmiter*. Pada *emmiter* dapat diatur beberapa parameter seperti tingkat *spawning* atau berapa banyak partikel yang ditampilkan setiap unit waktu, kecepatan awal dari partikel, range waktu partikel, dan warna partikel.

Tahap pembaruan dari *Particle System* yang dilakukan untuk setiap *frame* animasi dapat dipisahkan menjadi dua tahap yang berbeda [12]. Kedua tahap tersebut adalah,

1. *Simulation Stage,* selama tahap simulasi, jumlah partikel baru yang harus diciptakan dihitung berdasarkan tingkat *spawning* dan interval antar *update* serta setiap partikel ditempatkan pada pada posisi tertentu dalam ruang 3D berdasarkan posisi *emmiter* dan lokasi yang telah ditentukan.
2. *Rendering Stage,* setelah tahap *update* selesai, setiap partikel akan direnderer sebagai *pixel* tunggal dalam resolusi kecil.

### 2.5.3 Point Cloud Viewer

Point Cloud Viewer merupakan *plugin* dari Unity3D untuk menampilkan data *point cloud.* Keunggulan dari *plugin* ini adalah kemampuan untuk memvisualisasikan lebih dari 10 juta RGB *points* dengan data masukan berupa *file* bertipe data OFF [13].

Metode visualisasi dari Point Cloud Viewer adalah dengan membuat *gameobject* baru sesuai jumlah *point* group yang terbentuk. Masing-masing *gameobject* akan direnderer terpisah dan disimpan dalam *storage* sehingga dapat mempercepat tahap visualisasi pada *point cloud* yang pernah diproses.

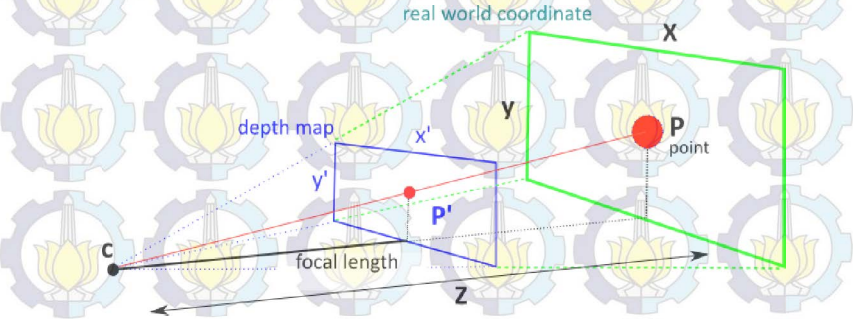
## 2.6 Konversi Peta Kedalaman

Peta kedalaman atau *depth map* adalah gambar dua dimesi yang berisikan informasi tentang jarak permukaan sebuah objek terhadap sudut pandang atau *view point* tertentu. Jarak ini biasanya berhubungan dengan data kedalaman yang berupa koordinat $z$ dari sebuah koordinat kartesian. Dalam kasus ini, sumbu $z$ yang dimaksud adalah sumbu $z$ yang relatif terhadap view dari kamera, dan bukan merupakan sumbu $z$ dari koordinat dunia. Untuk mendapatkan data *point cloud* tiga dimensi, maka peta kedalaman harus dikonversi dari data dua dimensi menjadi data tiga dimensi yang memiliki nilai terhadap sumbu $x$, $y$, dan $z$ pada bidang kartesian. Skema proses konversi peta kedalaman ke koordinat dunia dapat dilihat pada gambar 2.4.

Dalam proses konversi *depth map* menjadi data *point cloud* tiga dimensi melibatkan parameter intrinsik kamera yang disebut sebagai $K$ dan juga melibatkan data *depth map* yang disebut sebagai $D$. Konversi ini dilakukan secara terpisah pada masing-masing titik di *depth map* yang

dilambangkan sebagai $u$. Berdasarkan kedua data tersebut, persamaan 2.3 digunakan untuk melakukan konversi *depth map* [14].

$$V_i(u) = D_i(u)K^{-1}[u, 1] \quad (2.3)$$




**Gambar 2.4** Skema proses konversi *depth map* ke *point cloud*

## 2.7 Polygon *File*

Polygon *file* format atau PLY adalah deskripsi objek sederhana yang dirancang sebagai format yang nyaman bagi para peneliti yang bekerja dengan model *polygonal* [15]. Versi awal dari format ini digunakan oleh peneliti di Standford University dan UNC Chapel Hill.

PLY menggambarkan sebuah objek sebagai kumpulan simpul, wajah atau permukaan, dan unsur-unsur lain, termasuk properti warna dan arah vektor normal yang dapat disisipkan ke dalam tiap element. Secara sederhana, *file* PLY berisikan daftar *x*, *y*, dan *z* setiap simpul dan daftar *faces* yang diurutkan berdasarkan indeks. Struktur dari *file* PLY terdiri dari tiga bagian utama. Bagian tersebut adalah,

a. *Header*, yang mencangkup deskripsi dari setiap jenis elemen, termasuk nama dari elemen, jumlah elemen dari objek, dan daftar properti yang berhubungan dengan elemen. *Header* menentukan berapa banyak simpul dan *polygon* dalam *file* serta menyatakan properti yang terkait pada setiap simpul atau titik, seperti elemen koodinat kartesian (*x, y, z*), vektor normal, dan warna. [16]. Pada *header* juga menjelaskan *format* data berupa *binary* atau ASCII.
b. *Vertex list,* merupakan bagian dari *file* PLY yang memuat nilai dari tiap element yang telah dijelaskan pada *header*. Pada

umumnya, *vertex list* berisikan koordinat *point* dengan memiliki tiga properti, yakni properti *x, y, z*.

c. *Face list,* merupakan bagian dari *file* PLY setelah *vertex list*. Deskripsi dari *face list* juga telah dideskripsikan pada *header* setelah deskripsi dari *vertex list*. *Face list* dapat memuat informasi berupa *mesh* dari data *point cloud*.

## 2.8 Bidang Geometri Primitif

Istilah geometri primitif dalam komputer grafik dan sistem CAD telah digunakan dalam berbagai aspek, dengan makna umum yakni bidang geometri paling sederhana yang dapat digambar atau disimpan. Geometri primitif pada umumunya terdiri dari titik, garis, lingkaran, *triangle*, kurva, dan poligon. Sedangkan pada bidang geometri primitif tiga dimensi terdiri dari *plane*, *sphere, cubes, cylinder, pyramid, dan toroid*. Bidang geometri primitif tiga dimensi merupakan bidang yang memiliki komponen *x, y,* dan *z* dalam koordinat kartesian. Masing-masing bidang geometri primitif memiliki persamaan matematika yang dapat dibentuk dari beberapa titik pada permukaan bidang tersebut.

### 2.8.1 Bidang *Plane*

Persamaan dasar dari bidang geometri *plane* [17] adalah

$$ax + by + cz + d = 0 \tag{2.4}$$

Untuk mendapatkan koefisien $a$, $b$, $c$, dan $d$ dapat dibutuhkan tiga sampel titik dari permukaan *plane*. Sedangkan pada ruang *Euclidean* dimensi berapapun, persamaan dari bidang *plane* dapat ditentukan dengan empat cara, yakni

a. Tiga titik non-linier, titik tersebut tidak berada dalam satu garis,
b. Sebuah garis dan titik yang tidak berada di garis tersebut,
c. Dua garis yang saling sejajar,
d. Dua garis yang berbeda namun berpotongan.

Metode *Cramer's rule*, merupakan salah satu metode untuk menentukan koefisien dari persamaan *plane* dengan informasi koordinat dari tiga titik pada bidang kartesian tiga dimensi. Misalkan titik-titik tersebut,

$$p_1 = (x_1, y_1, z_1),\ p_2 = (x_2, y_2, z_2)\ , p_3 = (x_3, y_3, z_3) \quad (2.5)$$

maka akan didapatkan nilai deskriminan melalui persamaan 2.6.

$$D = \begin{bmatrix} x_1 & y_1 & z_1 \\ x_2 & y_2 & z_2 \\ x_3 & y_3 & z_3 \end{bmatrix} \quad (2.6)$$

Koefisien $a$, $b$, $c$, dan $d$ pada persamaan 2.4 didapatkan dari mensubsitusikan koordinat titik-titik tersebut dalam persamaan 2.7 hingga 2.9.

$$a = \frac{-d}{D} \begin{bmatrix} 1 & y_1 & z_1 \\ 1 & y_2 & z_2 \\ 1 & y_3 & z_3 \end{bmatrix} \quad (2.7)$$

$$b = \frac{-d}{D} \begin{bmatrix} x_1 & 1 & z_1 \\ x_2 & 1 & z_2 \\ x_3 & 1 & z_3 \end{bmatrix} \quad (2.8)$$

$$c = \frac{-d}{D} \begin{bmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ x_3 & y_3 & 1 \end{bmatrix} \quad (2.9)$$

Persamaan tersebut parametrik pada nilai $d$, nilai dari $d$ sama dengan bilangan *non-zero* dan mensubsitusikan nilai $d$ akan menghasilkan satu set solusi. Jarak suatu titik ke bidang *plane* dapat dicari dengan mensubsitusi persamaan *plane* dan suatu titik ke dalam persamaan 2.10.

$$Ds = \frac{|ax_1 + by_1 + cz_1 + d|}{\sqrt{a^2 + b^2 + c^2}} \quad (2.10)$$

Jika hasil $Ds$ = 0 maka titik berada pada permukaan bidang plane tersebut.

### 2.8.2 Bidang *Sphere*

Bidang *sphere* adalah sebuah objek geometri yang bulat sempurna dalam ruang tiga-dimensi. Jika dianalogikan dengan lingkaran pada ruang dua dimensi, bidang *sphere* didefinisikan secara matematika sebagai himpunan titik-titik yang memiliki jarak $R$ yang

sama dari suatu titik dalam koordinat kartesian. Nilai *R* tersebut adalah jari-jari dari *sphere*, dan suatu titik tersebut adalah pusat dari *sphere*. Jarak lurus terbesar dari titik di permukaan *sphere* ke titik lain dan melewati pusat dari *sphere* merupakan dua kali dari nilai *R* atau disebut dengan diameter. Didalam geometri analitik, persamaan dari bidang *sphere* dengan koordinat titik pusat $(x_0, y_0, z_0)$ dan jari-jari *R* adalah

$$(x - x_0)^2 + (y - y_0)^2 + (z - z_0)^2 = R^2 \qquad (2.11)$$

dimana $x$, $y$, $z$ merupakan koordinat dari semua titik yang berada di permukaan bola tersebut pada bidang kartesian. Selain persamaan tersebut, semua titik pada permukaan *sphere* juga akan memenuhi persamaan 2.12.

$$\begin{gathered} x = x_0 + R\cos\theta \ \sin\varphi, \\ y = y_0 + R\sin\theta \ \sin\varphi, \\ z = z_0 + R\cos\varphi, \\ (0 \le \theta \le 2\pi \ dan \ 0 \le \varphi \le \pi) \end{gathered} \qquad (2.12)$$

Persamaan model *sphere* dapat dicari dari empat buah titik pada permukaan bidang *sphere* [18]. Empat titik tersebut tidak berada pada bidang *plane* yang sama. Misalkan titik-titik tersebut,

$$\begin{gathered} p_1 = (x_1, y_1, z_1), \\ p_2 = (x_2, y_2, z_2), \\ p_3 = (x_3, y_3, z_3), \\ p_4 = (x_4, y_4, z_4) \end{gathered} \qquad (2.13)$$

Maka dapat diselesaikan melalui persamaan determinan berikut,

$$\begin{vmatrix} x^2+y^2+z^2 & x & y & z & 1 \\ {x_1}^2+{y_1}^2+{z_1}^2 & x_1 & y_1 & z_1 & 1 \\ {x_2}^2+{y_2}^2+{z_2}^2 & x_2 & y_2 & z_2 & 1 \\ {x_3}^2+{y_3}^2+{z_3}^2 & x_3 & y_3 & z_3 & 1 \\ {x_4}^2+{y_4}^2+{z_4}^2 & x_4 & y_4 & z_4 & 1 \end{vmatrix} = 0 \qquad (2.14)$$

Persamaan 2.14 dapat disederhanakan dengan mengevaluasi *cofactors* untuk baris pertama pada persamaan determinan tersebut,

$$(x^2 + y^2 + z^2)M_{11} - xM_{12} + yM_{13} - zM_{14} + M_{15} = 0 \quad (2.15)$$

Dengan mensubsitusi persamaan 2.11, maka koordinat titik pusat $(x_0,y_0,z_0)$ dan jari-jari $R$ diperoleh dari persamaan 2.16 hingga 2.19.

$$x_0 = +0.5\,\frac{M_{12}}{M_{11}} \quad (2.16)$$

$$y_0 = -0.5\,\frac{M_{13}}{M_{11}} \quad (2.17)$$

$$z_0 = +0.5\,\frac{M_{14}}{M_{11}} \quad (2.18)$$

$$R^2 = {x_0}^2 + {y_0}^2 + {z_0}^2 - \frac{M_{15}}{M_{11}} \quad (2.19)$$

Tidak akan ditemukan solusi persamaan *sphere* apabila $M_{11}$ bernilai 0. Jika persamaan *sphere* yang ditemukan tidak termasuk himpunan titik pada permukaan *sphere* yang diinginkan, maka seluruh titik yang disubsitusikan berada dalam bidang *plane* yang sama atau tiga titik tersebut berada dalam satu garis lurus.

### 2.8.3 Bidang *Cylinder*

Bidang *cylinder* didefinisikan sebagai himpunan semua titik dalam ruang *euclidean* tiga dimensi atau $R^3$ yang terletak pada jarak tetap dari garis lurus yang merupakan sumbu dari bidang *cylinder*. Dalam geometri analitik, *cylinder* didefinisikan oleh jari-jari, tinggi dan arah bidang *cylinder*. Untuk dapat menghitung bidang *cylinder* dari *minimum set of points* (MSS), tiga buah titik harus tidak kolinier [19]. Dari ketiga titik tersebut akan diperoleh lingkaran setelah menyelesaikan persamaan bidang *plane*, persamaan 2.4. Besar jari-jari *r* dan titik tengah dari bidang *cylinder* $c = (x_0, y_0, z_0)$ dapat diperoleh dengan persamaan 2.20 dan 2.21.

$$d(p_1, c) = d(p_2, c) = d(p_3, c) = r \qquad (2.20)$$

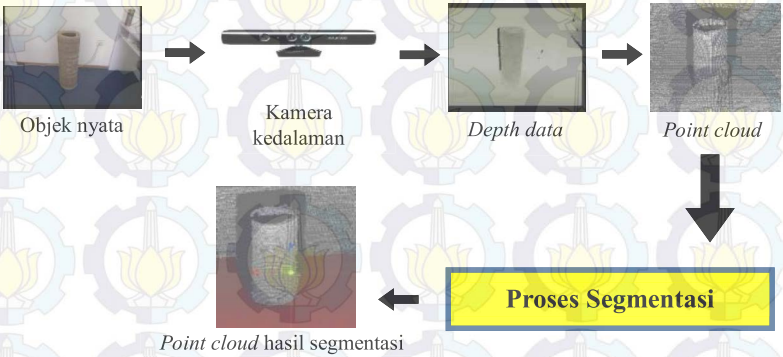
$$d(p_1, c) = \sqrt{(x_i - x_0)^2 + (y_i - y_0)^2 + (z_i - z_0)^2} \qquad (2.21)$$

sedangkan pada arah dari bidang *cylinder* diperoleh dari normal vektor dari bidang *plane* yang diestimasi sebelumnya.

# BAB 3
# DESAIN DAN IMPLEMENTASI SISTEM

## 3.1 Desain Sistem

Tugas akhir ini bertujuan untuk membuat aplikasi yang mampu melakukan segmentasi model primitif tiga dimensi berdasarkan data *point cloud* pada lingkungan nyata. Data *point cloud* diperoleh melalui perekaman secara langsung dengan kamera Kinect, hasil *recording* dari kamera Kinect dalam bentuk *file* Binary dan *file* PLY. Visualisasi dari data *point cloud* akan ditampilkan berupa *point* yang mewakili koordinat dari tiap data *point cloud* sesuai dengan koordinat dunia. Data *point cloud* yang merepresentasikan objek tiga dimensi sederhana yang tersegmentasi akan ditampilkan dengan warna yang berbeda dengan objek yang lain. Gambaran umum dari sistem dari tugas akhir ini diilustrasikan dalam gambar 3.1.




**Gambar 3.1** Gambaran umum desain sistem

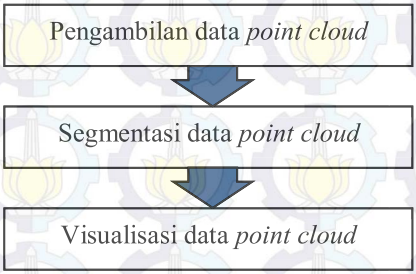
## 3.2 Alur Kerja Sistem

Pada tahap ini akan dijelasan mengenai rancangan sistem pada perangkat lunak. Rancangan sistem perangkat lunak menggunakan pustaka Point Cloud Library dan Microsoft Kinect Unity SDK serta didesain pada IDE Unity3D, IDE tersebut juga sebagai perender grafis

untuk visualisasi data *point cloud*. Tahapan proses dari tugas akhir ini sebagai berikut,

a. Pengambilan data *point cloud*
b. Segmentasi data *point cloud*
c. Visualisasi data *point cloud*

Untuk ketiga proses ini dapat diilustrasikan pada gambar 3.2.




**Gambar 3.2** Diagram alir alur kerja sistem

## 3.3 Pengambilan Data *Point Cloud*

Data *point cloud* diperoleh dengan berdasarkan peta kedalaman atau *depth map* dari hasil tangkapan kamera Kinect. Pada tugas akhir ini, terdapat dua cara dalam proses pengambilan data *point cloud*.

### 3.3.1 *Multiframe*

*Multiframe* pada proses pengambilan data *point cloud* adalah kumpulan peta kedalaman dua dimensi dalam jangka waktu tertentu. Cara pengambilan ini diterapkan pada saat *stream* dari kamera kedalaman secara langsung dan *replay* hasil rekaman dari *stream* sebelumnya. Hasil rekam dari proses *stream* pada kamera Kinect berupa *file* Binary yang berisikan kumpulan peta kedalaman selama proses merekam. Tahapan dalam cara pengambilan secara *multiframe* dibagi menjadi dua bagian,

#### *3.3.1.1 Menyimpan dan membaca file binary*

Proses menyimpan dan membaca *file* binary merupakan tahapan yang tidak selalu dilaksanakan dalam cara pengambilan data *point cloud* secara *multiframe*. Tahapan ini dapat dilakukan jika *user* ingin

memisahkan proses merekam peta kedalaman dari lingkungan nyata dengan proses segmentasi. Dari hasil *stream* pada kamera kinect maka didapatkan peta kedalaman dengan lebar 320 *pixel* dan tinggi 240 *pixel*. Setiap *cell* pada peta kedalaman tersebut memuat data *depth* dari suatu titik pada permukaan objek. Setiap *frame* dari peta kedalaman akan di-*serialize* dan disimpan dengan format *file* Binary pada *storage* saat proses merekam berakhir. *File* Binary akan dibaca berdasarkan *playback speed* tertentu untuk didapatkan kembali data peta kedalaman sebelum melalui proses konversi ke data *point cloud.*

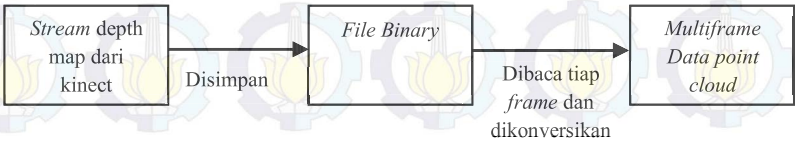
### *3.3.1.2 Mengkonversikan peta kedalaman*

Peta kedalaman yang didapatkan kamera Kinect sebenarnya adalah matriks dua dimensi yang berisi data kedalaman. Data ini akan dikonversi menjadi data tiga dimensi atau *point cloud* dengan menggunakan persamaan 2.3. Persamaan tersebut diaplikasikan dalam *pseudocode* pada gambar 3.3.

```
FUNCTION DepthToPointCloud (float x, float y, float depth) {
DEKLARASI
   Point as float[3] contain zero
BEGIN
   IF (depth > 0) DO
      Point[2] = depth/ 100
      Point[0] = (x - 0.5) * 0.003501 * Point[2] * 320
      Point[1] = (0.5 - y)* 0.003501 * Point[2] * 240
   END IF
   RETURN Point
END FUNCTION
```

**Gambar 3.3** *Pseudocode* konversi peta kedalaman

Jika tidak melalui tahapan menyimpan dan membaca *file* Binary, maka peta kedalaman dari *stream* kamera Kinect akan langsung dikonversikan ke data *point cloud* tiap *frame*. Ilustrasi dari proses pengambilan data *point cloud* secara *multiframe* dapat dilihat pada gambar 3.4.




**Gambar 3.4** Diagram alir pengambilan data *point cloud multiframe*

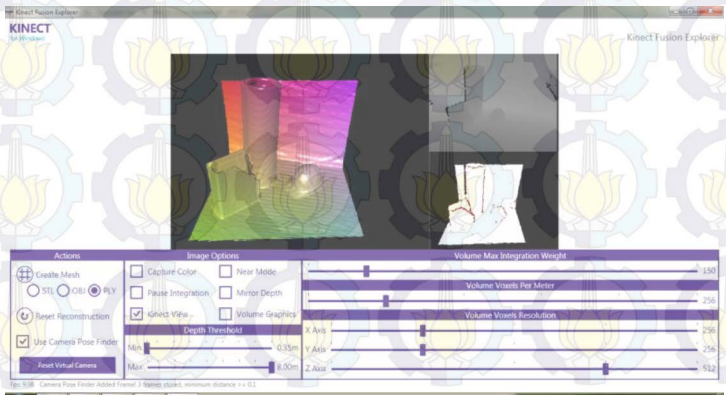
### 3.3.2 *Singleframe*

*Singleframe* dirancang dengan masukan data *point cloud* berupa *file* Polygon atau PLY dari *storage*. *File* Polygon dapat diperoleh dari proses rekonstruksi dengan bantuan pustaka Kinect Fusion. *File* Polygon dapat pula didapatkan dari mengunduh dari internet. Tahapan dalam proses pengambilan data *point cloud* secara *singleframe* pada tugas akhir ini diilustrasikan dalam diagram alir pada gambar 3.5.


**Gambar 3.5** Diagram alir pengambilan data *point cloud singleframe*

#### *3.3.2.1 Menyimpan Hasil Rekonstruksi* Kinect Fusion

Hasil *stream* lingkungan nyata dari kamera Kinect direkonstruksi dengan pustaka Kinect Fusion dan disimpan dalam format *file* Polygon. Tampilan dari pustaka Kinect Fusion yang digunakan sesuai gambar 3.6. Hasil rekonstruksi tersebut merupakan *singleframe* data *point cloud* tiga dimensi yang memuat elemen *x, y, z* dalam koordinat dunia sehingga tidak memerlukan konversi dari peta kedalaman menjadi data *point cloud.*



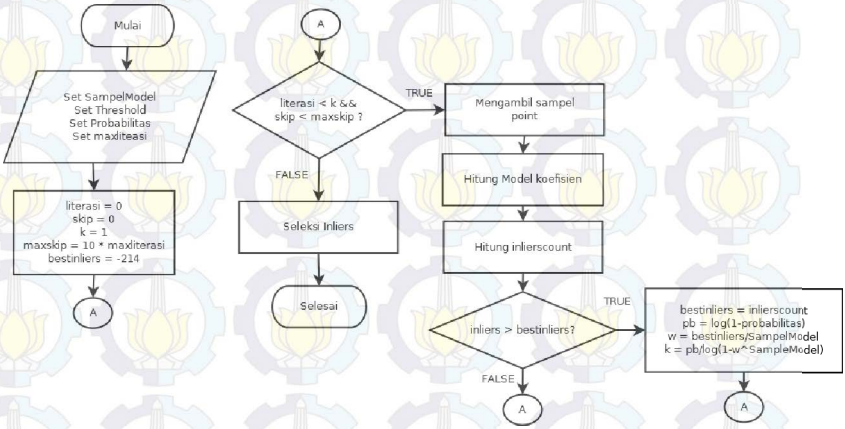

**Gambar 3.6** *Interface* dari Kinect Fusion

### *3.3.2.2 Membaca File Polygon*

Setiap *point* dari data *point cloud* diperoleh dari membaca perbaris pada bagian *vertex list*. Informasi jumlah point dalam satu *frame* diperoleh pada bagian *header file* beserta elemen-elemen dari setiap *vertex*.

## 3.4 Segmentasi Data *Point Cloud*

Segmentasi data *point cloud* dilakukan setiap *frame* baik pada masukan berupa *multiframe* maupun *singleframe.* Proses segmentasi menggunakan metode Random Sample Consensus dengan luaran data *inliers* atau list *point* dari data *point cloud* yang merupakan model tersegmentasi, dan data *outliers* atau list *point* yang tidak termasuk model tersegmentasi. Proses segmentasi dari data point cloud digambarkan dalam *flowchart* pada gambar 3.7.




**Gambar 3.7** *Flowchart* proses segmentasi data *point cloud*

Tahapan proses dalam proses segmentasi data *point cloud* adalah,

### 3.4.1 Mengatur Parameter Segmentasi

Terdapat empat buah parameter penting dalam setiap proses segmentasi pada seluruh model yang dirancang. Parameter tersebut adalah,

*point*. *Pseudocode* dari algoritma menghitung koefisien model bidang *plane* dijelaskan pada gambar 3.8.

```
FUNCTION KoefisienModelPlane
DECLARATION
   p1p0, p2p0 as float[3]
   Dot, Norm as float
   Input cloud as float[3,3]
   Output coefficients as float[4]
BEGIN
   p1p0 = cloud[1] - cloud[0]
   p2p0 = cloud[2] - cloud[0]
   IF ((p1p0, p2p0) is collinear?) THEN
      RETURN false
   ENDIF
   coefficients[0]=(p1p0[1]*p2p0[2])-(p1p0[2]*p2p0[1])
   coefficients[1]=(p1p0[2]*p2p0[0])-(p1p0[0]*p2p0[2])
   coefficients[2]=(p1p0[0]*p2p0[1])-(p1p0[1]*p2p0[0])
   coefficients[3]=0
   Dot = sum of sqr(coefficients)
   Norm = sqrt(Dot)
   coefficients = coefficients / norm
  RETURN true
END FUNCTION
```

**Gambar 3.8** *Pseudocode* fungsi untuk menghitung koefisien bidang *plane*

Koefisien yang diperoleh dalam *pseudocode* pada gambar 3.8 merupakan nilai $a$, $b$, $c$, dan $d$ dari persamaan 2.4. Variabel *cloud* merupakan sampel data dari proses sebelumnya, sedangkan variable *coefficients* merupakan koefisien model. Koefisien dari bidang *sphere* pada persamaan 2.9, yakni titik pusat ($x_0$, $y_0$, $z_0$) dan jari-jari $R$ diperoleh dari *pseudocode* pada gambar 3.9. Model *cylinder* memiliki tujuh koefisien, dimana tiga koefisien merupakan koordinat *x, y, z* suatu titik pada *axis* model *cylinder,* tiga koefisien menjelaskan arah dari *axis* model *cylinder,* dan satu koefisien merupakan jari-jari model *cylinder*. Dua buah sampel *point* yang digunakan akan dihitung vektor normalnya berdasarkan *point* tetangga. Metoda yang digunakan dalam pencarian *point* tetangga adalah *Kdtree*. Dari *point* tetangga akan didapatkan persamaan bidang *plane* untuk mendapatkan nilai vektor normal dari sampel *point* tersebut. Data koordinat *x, y, z* dan vektor normal dari sampel *point* akan digunakan untuk mendapatkan koefisien dari persamaan model *cylinder*.

```
FUNCTION KoefisienModelSphere
DECLARATION
   temp as float[4,4]
   m11, m12, m13, m14, m15 as float
   Input cloud as float[4,3]
   Output coeff as float[4]
BEGIN
   FOR i <- 0 to 4 DO
      temp[i][0] = cloud[i][0]
      temp[i][1] = cloud[i][1]
      temp[i][2] = cloud[i][2]
      temp[i][3] = 1
   ENDFOR
   m11 = Determinant(temp)
   IF (m11 == 0)
      RETURN false
   ENDIF
   FOR j <- 0 to 4 DO
      temp[i][0] = square of cloud[j][0] + square of//
      cloud[j][1] +  square of cloud[j][2]
   ENDFOR
   m12 = Determinant(temp)
   FOR k <- 0 to 4 DO
      temp[k][1] = temp[k][0]
      temp[k][0] = cloud[k][0]
   ENDFOR
   m13 = Determinant(temp)
   FOR l <- 0 to 4 DO
      temp[l][2] = temp[l][1]
      temp[l][1] = cloud[l][1]
   ENDFOR
   m14 = Determinant(temp)
   FOR m <- 0 to 4 DO
      temp[m][0] = temp[m][2]
      temp[m][1] = cloud[m][0]
      temp[m][2] = cloud[m][1]
      temp[m][3] = cloud[m][2]
   ENDFOR
   m15 = Determinant(temp)
   coeff[0] = 0.5f * m12 / m11
   coeff[1] = 0.5f * m13 / m11
   coeff[2] = 0.5f * m14 / m11
   coeff[3] = Sqrt(square of coeff[0]+square of coeff[1]//
   +square of coeff[2] - m15 / m11);
   RETURN true
END FUNCTION
```

**Gambar 3.9** *Pseudocode* fungsi untuk menghitung koefisien bidang *sphere*

### 3.4.4 Menentukan Koefisien Model Terbaik

Koefisien model terbaik ditentukan dari jumlah *inliers* terbanyak dari semua hipotesa persamaan yang didapatkan selama proses literasi berjalan. Jumlah *inliers* merupakan jumlah *point* yang memiliki jarak yang sesuai dengan threshold yang telah ditentukan dari koordinat *point* tersebut terhadap persamaan hipotesa model. Jumlah *inliers* yang didapatkan setiap pengujian hipotesa persamaan akan mempengaruhi nilai *k* sesuai persamaan 2.2. *Pseudocode* dari algoritma menghitung jumlah *inliers* dari koefisien model *plane* sesuai dengan persamaan 2.8 dijelaskan pada gambar 3.10.

```
FUNCTION CountInliersPlane
DECLARATION
   Point as float[3]
   Dot as double
   Input cloud as float[n,3]
   Input coefficients as float[4]
   Input threshold as double
   Output npoint as int
BEGIN
   For i<-0 to n DO
      Point = [cloud[i][0], cloud[i][1], cloud[i][2], 1]
      Dot = sum (Point*coefficients)
      IF (Dot < threshold)
        Increment npoint
      ENDIF
   ENDFOR
END FUNCTION
```

**Gambar 3.10** *Pseudocode* fungsi untuk menghitung *inliers* model *plane*

Sebelum dihitung jumlah *inliers,* validitas dari setiap hipotesa persamaan ditentukan berdasarkan jumlah koefisien model yang didapatkan. Pada model *sphere,* jumlah *inliers* diperoleh dengan menggunakan *pseudocode* pada gambar 3.11 .

### 3.4.5 Seleksi *Inliers* Berdasarkan Koefisien Model Terbaik

Setelah mendapatkan koefisien terbaik dari proses literasi, setiap *point* akan dihitung jaraknya kembali terhadap persamaan model terbaik. *Point* yang memiliki jarak yang sesuai *threshold* termasuk data *inliers*. Sedangkan *point* yang memiliki jarak tidak sesuai *threshold* termasuk data *outliers*. Algoritma dari seleksi *inliers* dari model *plane* memiliki

```
FUNCTION CountInliersSphere
DECLARATION
   Input cloud as float[n,3]
   Input coeff as float[4]
   Input double threshold
   Output npoint as int
BEGIN
   FOR i <- 0 to n DO
      IF (Abs(Sqrt(Sqr(cloud[i][0]-coeff[0])+ //
      Sqr(cloud[i][1]-coeff[1])+ //
      Sqr(cloud[i][2]-coeff[2]))-coeff[3] ) < threshold )
        npoint++
      ENDIF
   ENDFOR
END FUNCTION
```

**Gambar 3.11** *Pseudocode* fungsi untuk menghitung *inliers* model *sphere*

struktur yang sama dengan algoritma pada gambar 3.10. Setiap *point* yang memiliki jarak kurang dari *threshold* akan disimpan dalam larik sebagai data *inliers*. Pada model *sphere*, algoritma menghitung seleksi *inliers* memiliki struktur yang sama dengan algoritma menghitung jumlah *inlier* model *sphere*.

## 3.5 Visualisasi Data *Point Cloud*

Visualisasi dari data *point cloud* menggunakan *Particle System* dan Point Cloud Manager. Data *inliers* diberikan warna yang berbeda dengan data *point cloud* yang lain atau *outliers* untuk membedakan antar objek.

### 3.5.1 Visualisasi dengan *Particle System*

Diagram alur visualisasi dengan *Particle System* diilustrasikan pada gambar 3.12.


**Gambar 3.12** Diagram alir proses visualisasi dengan *Particle System*

Tahapan proses visualisasi dengan *Particle System* terdiri atas,

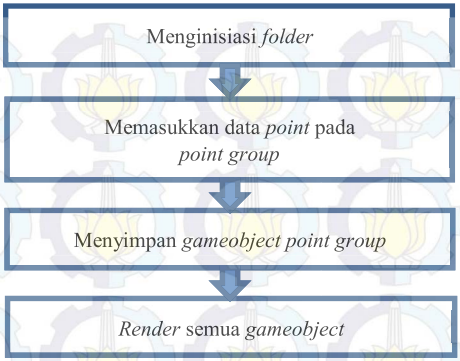
1. Menginisiasi setiap partikel
   Jumlah partikel yang diinisiasi sejumlah data *point* pada *point cloud.* Beberapa elemen dari partikel dipersiapkan, seperti ukuran, koordinat dan warna.
2. Memasukkan data *point* pada partikel
   Data koordinat dari setiap *point cloud* dimasukkan kedalam setiap point, sedangkan elemen warna dikorelasikan dengan data *inliers* untuk membedakan antara data *inliers* dan data *outliers.*
3. *Render gameobject Particle System*
   Proses *render* pada *Particle System* dilakukan secara keseluruhan setelah proses memasukkan data *point* cloud beserta properti tampilan sehingga dihasilkan satu *gameobject.*

### 3.5.2 Visualisasi dengan Point Cloud Viewer

Proses visualisasi dengan Point Cloud Viewer memiliki empat tahapan. Tahapan tersebut terdiri atas,

1. Menginisiasi *folder*
   Point Cloud Viewer mempunyai mekanisme untuk menyimpan *gameobject.* Setiap *gameobject* tersebut disimpan dalam *file asset*. Pada awal proses dilakukan pengecekkan *folder resource* sebagai tempat penyimpanan *asset* dan akan mempersiapkan *folder* jika tidak tersedia. Apabila ingin menampilkan visualisasi yang telah tersimpan, maka cukup dengan mengakses *gameobject* yang telah ada.
2. Memasukkan data *point cloud* pada *point group*
   Koordinat dan warna dari setiap *point* akan disimpan dalam larik. Larik dibagi menjadi sejumlah *point group*. Kapasitas dari setiap *point group* ditentukan pada awal proses visualisasi.
3. Menyimpan *gameobject point group*
   *Gameobject* dari setiap *point group* disimpan dalam *file asset* pada *folder* yang telah diinisiasi.
4. *Render* semua *gameobject*
   *Gameobject* yang telah disimpan digabungkan dan ditampilkan dalam satu *prefab.*

Keempat tahapan dalam visualisasi dengan Point Cloud Viewer dapat diilustrasikan kedalam diagram alur pada gambar 3.13.




**Gambar 3.13** Diagram alir proses visualisasi dengan Point Cloud Viewer

## 3.6 Perancangan Alur Visualisasi

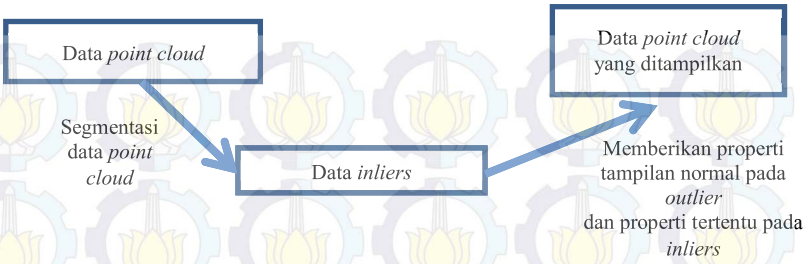
Proses segmentasi data *point cloud* yang membutuhkan waktu yang cukup lama, maka disusunlah rancangan alur visualisasi dari data *point cloud* agar proses visualisasi dapat berjalan dengan baik.

### 3.6.1 Visualisasi Secara Sinkronous

Pada visualisasi secara sinkronous, data *point cloud* yang ditampilkan ke layar akan diproses setelah tahap segmentasi berakhir. Alur data *point cloud* pada rancangan visualisasi ini adalah searah. Setelah proses segmentasi berakhir, data *inliers* akan dikorelasikan dengan data masukan *point cloud* untuk menentukan properti tampilan. Dalam tugas akhir ini properti yang dibedakan antara *inliers* dan *outliers* adalah warna. Ilustrasi dari perancangan visualisasi secara sinkronous dapat dilihat pada gambar 3.14.

### 3.6.2 Visualisasi Secara Asinkronous

Pada visualisasi secara asinkronous, data *point cloud* yang ditampilkan pada layar memiliki dua alur. Yang pertama saat didapatkan data masukan *point cloud*. Pada kondisi ini, jika tidak tersedia data *inliers* maka data *point cloud* akan ditampilkan seluruhnya sebagai *outliers*. Jika data *inliers* telah tersedia dari proses segmentasi




**Gambar 3.14** Diagram alir proses visualisasi secara sinkronous

sebelumnya, maka data masukan tersebut akan dikorelasikan dengan data inliers tersebut. Kondisi yang lain adalah saat didapatkan data *inliers* baru, maka properti tampilan dari *point cloud* yang akan ditampilkan akan disesuaikan dengan data tersebut. Ilustrasi dari perancangan visualisasi secara asinkronous dapat dilihat pada gambar 3.15.

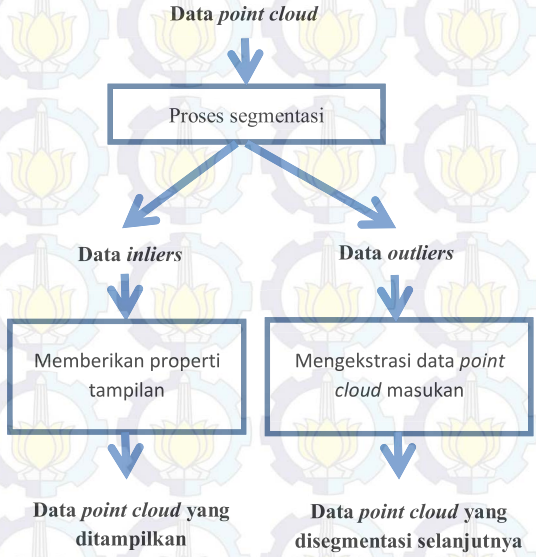

**Gambar 3.15** Diagram alir proses visualisasi secara asinkronous

## 3.7 Perancangan Alur Multisegmentasi

Alur multisegmentasi dirancang untuk mendapatkan data *inliers* dari model dengan koefisien model terbaik kedua, ketiga dan seterusnya. Kondisi dikarenakan metoda Ransac yang diterapkan akan memilih koefisien terbaik dalam setiap proses segmentasi sebagai data *inliers*. Tahapan utama dalam alur proses multisegmentasi sebagai berikut,

1. Dilakukan proses segmentasi model sesuai jenis model dan parameter yang ditentukan. Dari proses segmentasi akan dihasilkan data *inliers* atau data yang merepresentasikan model yang tersegmentasi, dan data *outliers* atau data selain model yang tersegmentasi.
2. Data *point cloud* yang termasuk data *inliers* diberikan properti tampilan untuk ditampilkan dalam layar, sedangkan data *outliers* digunakan sebagai data masukan *point cloud* pada proses segmentasi selanjutnya.

Alur multisegmentasi dapat diilustrasikan dalam diagram alir pada gambar 3.16.




**Gambar 3.16** Diagram alir alur multisegmentasi

# BAB 4
# PENGUJIAN DAN ANALISA

Pada bab ini dibahas mengenai pengujian dari perangkat lunak yang telah direalisasikan untuk mengetahui apakah fungsi dari sistem yang direncanakan telah bekerja sesuai dengan harapan. Pengujian pada penelitian ini dilakukan dalam beberapa bagian, yakni pengujian terhadap performa segmentasi, jenis visualisasi, alur visualisasi, dan multisegmentasi. Pengujian dilakukan pada pada sistem operasi Windows 7 Home Premium dengan *Integrated Development Environment* (IDE) yang digunakan adalah Unity3D versi 4.5.2. Sedangkan kamera kedalaman yang digunakan adalah kamera Kinect XBOX 360. Adapun spesifikasi komputer yang digunakan dalam pengujian ini dijelaskan pada tabel 4.1.

**Tabel 4.1** Spesifikasi Komputer

| **Komponen** | **Spesifikasi** |
|---|---|
| Sistem Operasi | Windows 7 Home Premium 64-bit |
| *Processor* | Intel Core ™ i5-4200M CPU @2.5 GHz (4CPUs) ~2.5GHz |
| *Memory* | 4096MB RAM |
| Versi *DirectX* | DirectX 11 |
| Versi *DxDiag* | 6.01.7600.16385 32-bit Unicode |
| *Display Adapter Name* | Intel HD Graphic 4600 – AMD Radeon HD 8500M |
| *Approx. Total Memory* | 3858 |

## 4.1 Pengujian Segmentasi *Point Cloud Singleframe*

Pengujian ini dilakukan untuk mengetahui performa dari perangkat lunak yang telah direalisasikan dalam melakukan proses segmentasi dan visualisasi data *point cloud singleframe*. *File* PLY

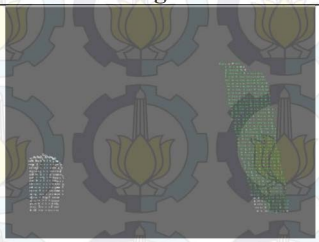
diperoleh dari hasil rekonstruksi dari kamera Kinect dengan bantuan pustaka Kinect Fusion.

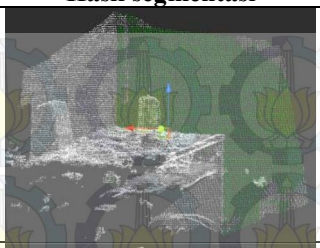
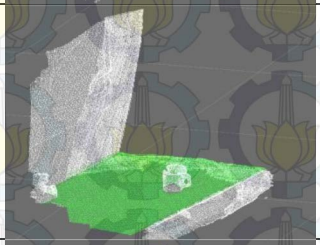
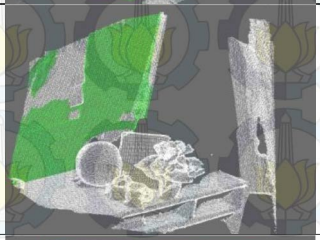
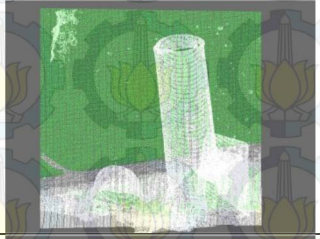
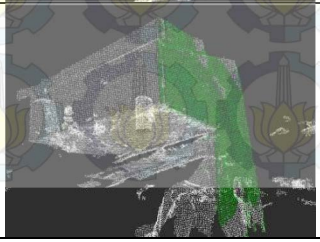
### 4.1.1 Segmentasi Model *Plane*

Jumlah data *point cloud* dari masukan *singleframe* dapat berbeda antar *file* Polygon. Kondisi ini dapat menyebabkan waktu tempuh dari proses segmentasi dan visualisasi berbeda-beda. Data Polygon yang diolah adalah bagian *vertex list* dengan elemen *x, y,* dan *z*. Pada pengujian tahap pertama dilakukan proses segmentasi model *plane* dengan visualisasi menggunakan Point Cloud Viewer. Pengujian ini bertujuan untuk mengetahui waktu tempuh dari proses segmentasi dengan jumlah *point* yang beragam. Parameter kontrol yang diatur pada pengujian tahap pertama sebagai berikut,

a. Jenis model : Normal Plane
b. Jumlah sampel : 3 *point*
c. *Threshold* model : 0,5
d. Probabilitas ditemukannya model : 0,99
e. Literasi maksimum : 1000
f. Visualisasi : Point Cloud Viewer
g. Data *Inliers* : *Point* berwarna hijau
h. Data *Outliers* : *Point* berwarna putih

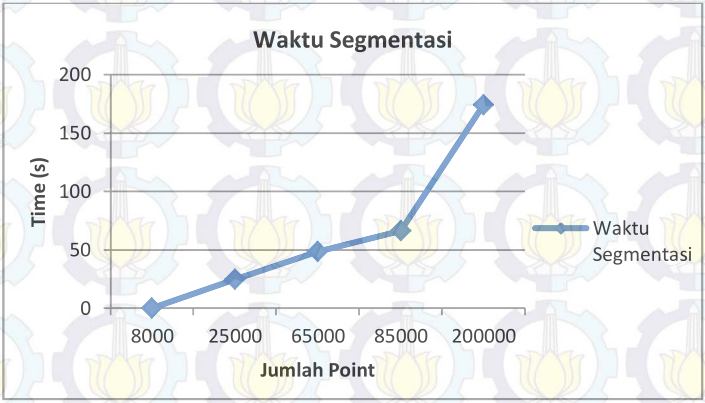
**Tabel 4.2** Hasil pengujian data masukan *singleframe* dengan variasi jumlah point

| No | Informasi | Hasil segmentasi |
|---|---|---|
| 1 | *Point*: 7830<br>*Trial*: 16<br>*Best Inliers*: 5000 (63,85%)<br>*Segmentation Time*(s) : 0,23<br>*Gameobject*:1 |  |

| No | Informasi | Hasil segmentasi |
|---|---|---|
| 2 | *Point*: 245130<br>*Trial*: 325<br>*Best Inliers*: 59208 (24,15%)<br>*Segmentation Time*(s) : 25,02<br>*Gameobject*:4 |  |
| 3 | *Point*: 645711<br>*Trial*: 232<br>*Best Inliers*: 174484 (27,02%)<br>*Segmentation Time*(s) : 48,47<br>*Gameobject*: 10 |  |
| 4 | *Point*: 838071<br>*Trial*: 247<br>*Best Inliers*: 221564 (26,43%)<br>*Segmentation Time*(s) : 66,56<br>*Gameobject*: 13 |  |
| 5 | *Point*: 1986735<br>*Trial*: 270<br>*Best Inliers*: 510414 (25,96%)<br>*Segmentation Time*(s) : 174,36<br>*Gameobject*: 31 |  |
| 6 | *Point*: 263934<br>*Trial*:314<br>*Best Inliers*: 64455 (24,42%)<br>*Segmentation Time*(s) : 25,54<br>*Gameobject*:5 |  |

Pada tabel 4.2, kolom informasi memuat informasi dari data masukan, dimana *point* adalah jumlah *point*, *trial* merupakan jumlah hipotesa persamaan yang diuji dalam mendapatkan koefisien model atau persamaan model terbaik, *best inliers* adalah jumlah *inliers* yang diperoleh, *segmentation* time merupakan waktu tempuh dari proses segmentasi dalam satuan detik, dan *gameobject* memuat jumlah *gameobject* yang terbentuk pada visualisasi dengan Point Cloud Viewer.

Berdasarkan hasil pengujian diatas, ditemukan ada dua faktor yang mempengaruhi waktu dari proses segmentasi. Yang pertama adalah jumlah masukan data *point cloud.* Pada grafik garis pada gambar 4.1, data grafik diperoleh dari hasil pengujian pada tabel 4.2 baris pertama hingga kelima, terlihat bahwa semakin banyak data masukan mengakibatkan waktu segmentasi yang semakin lama. Hal ini dikarenakan proses *voting* yang berjalan pada metoda Ransac akan menguji setiap hipotesa persamaan terhadap semua masukan *point*. Proses *voting* dilakukan untuk mendapatkan koefisien terbaik dalam proses segmentasi.

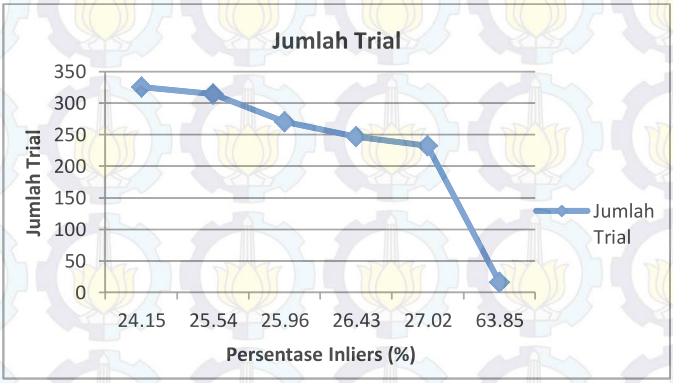



**Gambar 4.1** Grafik pengaruh jumlah *point* terhadap waktu segmentasi

Kedua adalah presentase *inliers* pada data masukan, hal ini dibuktikan berdasarkan kolom informasi pada baris kedua dan keenam tabel 4.2, dimana waktu proses segmentasi data masukan dengan perbedaan jumlah *point* sekitar 20 ribu *point* memiliki waktu segmentasi

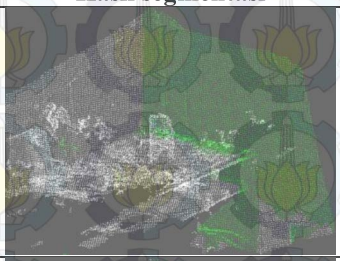
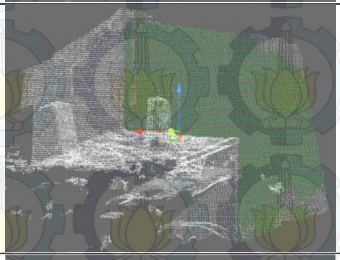
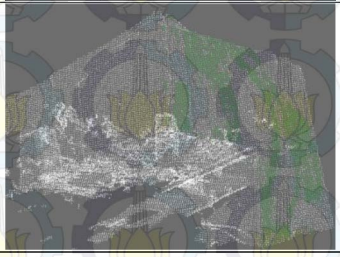
yang hampir sama. Kondisi tersebut dipengaruhi oleh jumlah *trial* pada hipotesa persamaan yang lebih sedikit walaupun memiliki jumlah *point* yang lebih banyak. Pada gambar 4.2, dengan jumlah masukan *point* yang berbeda, dapat ditarik kesimpulan bahwa semakin besar persentase *inliers* mengakibatkan semakin sedikit proses *trial* yang dikerjakan.

Selain kedua faktor tersebut, waktu proses segmentasi juga dipengaruhi oleh nilai *threshold*. Dari hasil pengujian yang lain dengan menggunakan data masukan berjumlah 245130 *point* didapatkan *time* total proses segmentasi dan visualisasi yang semakin besar saat nilai *threshold* diperkecil. Nilai *threshold* yang diatur dalam parameter awal proses segmentasi mempengaruhi presentase *inliers*. Sesuai dengan faktor kedua sebelumnya, presentase *inliers* mempengaruhi jumlah *trial* hipotesa persamaan dalam proses Ransac. Pengaturan nilai *threshold* yang tidak sesuai juga mengakibatkan hasil segmentasi yang tidak baik. Pengaruh tersebut dapat terlihat dari visualisasi data *inliers* pada kolom hasil segmentasi tabel 4.3. Berdasarkan hasil pada tabel tersebut, nilai *threshold* terbaik berkisar 0,5 hingga 1. Jika nilai *threshold* yang diatur lebih besar, maka terdapat bagian dari model lain yang dideteksi sebagai *inliers*. Saat nilai *threshold* lebih kecil, akan didapatkan bagian model yang dicari tidak termasuk kedalam *inliers*.




**Gambar 4.2** Grafik pengaruh persetase *inliers* terhadap jumlah *trial*

**Tabel 4.3** Hasil pengujian masukan *singleframe* dengan variasi *threshold*

| No | Informasi | Hasil segmentasi |
|---|---|---|
| 1 | *Threshold*: 5<br>*Trial*: 110<br>*Best Inliers*: 84547 (34,49%)<br>*Segmentation Time*(s) : 10,15<br>*Gameobject*:4 |  |
| 2 | *Threshold*: 1<br>*Trial*: 240<br>*Best Inliers*: 65428 (26,69%)<br>*Segmentation Time*(s) : 19,7<br>*Gameobject*:4 |  |
| 3 | *Threshold*: 0,5<br>*Trial*: 325<br>*Best Inliers*: 59208 (24,15%)<br>*Segmentation Time*(s) : 25.02<br>*Gameobject*:4 |  |
| 4 | *Threshold*: 0,1<br>*Trial*: 4561<br>*Best Inliers*: 24589 (18,6%)<br>*Segmentation Time*(s) : 346,52<br>*Gameobject*:4 |  |

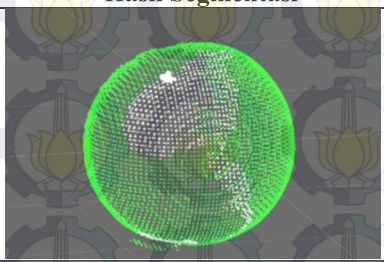
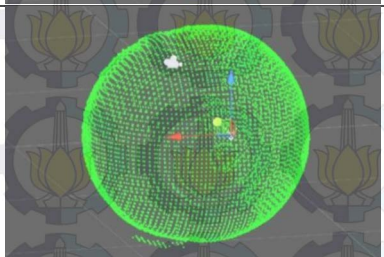
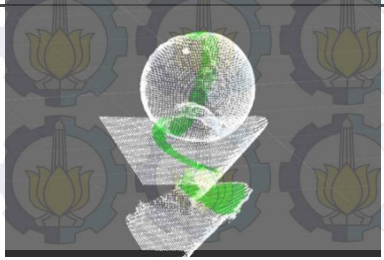
.

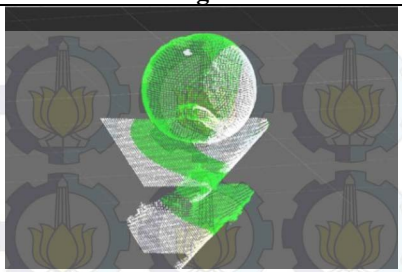
### 4.1.2 Segmentasi Model *Sphere*

Pengujian segmentasi dari model *sphere* dilakukan dengan parameter-parameter sebagai berikut,

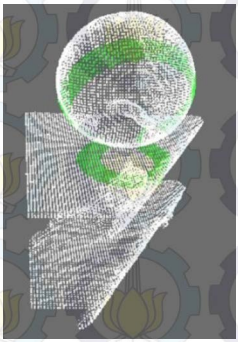
a. *Min Radius* : 0
b. *Max Radius* : 2
c. *Probability* : 0,5
d. Warna *inliers* : hijau
e. Warna *outliers* : putih
f. Visualisasi secara sinkronous dengan *Particle System*

**Tabel 4.4** Hasil segmentasi model *sphere*

| No | Informasi | *Threshold* | Hasil Segmentasi |
|---|---|---|---|
| 1 | *Point*: 55880<br>*Segmentation Time*(s) : 0,82<br>*Best Inliers*: 55753 (99,77%) | 0,01 |  |
| 2 | *Point*: 55880<br>*Segmentation Time*(s) : 0,81<br>*Best Inliers*: 55880 (100%) | 0,05 |  |
| 3 | *Point*: 132952<br>*Segmentation Time*(s) : 1,88<br>*Best Inliers*: 14695 (11,05%) | 0,01 |  |

| No | Informasi | *Threshold* | Hasil Segmentasi |
|---|---|---|---|
| 4 | *Point*: 132952 *Segmentation Time*(s) : 1,86 *Best Inliers*: 44798 (35,95%) | 0,05 |  |

Berdasarkan hasil pada tabel 4.4, waktu proses segmentasi dari model *sphere* dipengaruhi pula dengan jumlah *point*, presentase *inliers*, dan nilai *threshold*. Pada data masukan berupa objek gabungan, pada baris ketiga dan keempat digabungkan dengan model *plane*, objek lain tersebut masih terdeteksi sebagai *inliers* dari model *sphere*. Kondisi ini diakibatkan parameter *radius* maksimal yang diatur adalah 2 meter. Pada gambar 4.3, hasil segmentasi model *sphere* dengan parameter *radius* maksimal 0,1 meter mendekat model yang sebenarnya, namun masih ada bagian dari model *plane* yang termasuk *inliers*.



**Gambar 4.3** Segmentasi model *sphere* dengan *radius* maksimum 0,1

### 4.1.3 Segmemtasi Model Cylinder

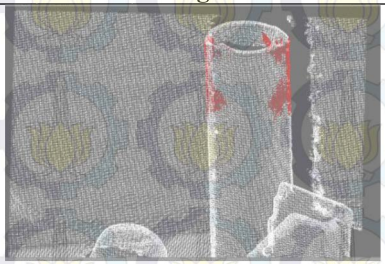
Pada pengujian dari model *cylinder* menggunakan data *point cloud* berjumlah *point* dengan parameter sebagai berikut,

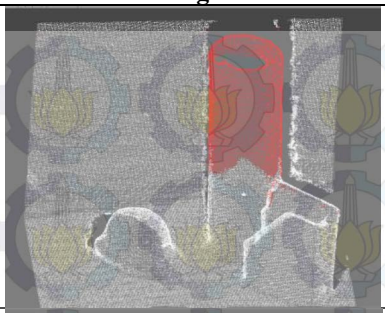
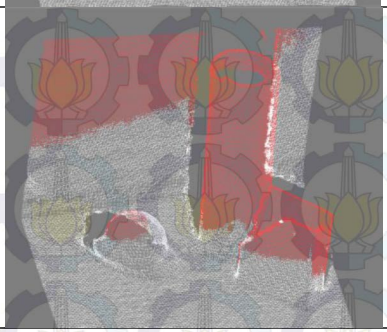
a. *Min Radius* : 0,05

b. *Max Radius* : 0,1
c. *Probability* : 0,9
d. Warna *inliers* : merah
e. Warna *outliers* : putih
f. Visualisasi secara sinkronous dengan *Particle System*
g. *Axis* : 0,5; 0; 0
h. *Viewport* : 0; 0; 0 (*default*)
i. *Eps angle* : 0,1

Pada segmentasi model *cylinder,* dibutuhkan normal vektor dari sampel *point* yang diambil. Parameter *viewport* digunakan dalam mendapatkan normal vektor setelah didapatkan parameter *plane* dari *point* tetangga. Parameter *axis* dan *eps angle* merupakan toleransi dari arah model *cylinder* yang sesuai. Kedua parameter tersebut berkaitan dengan koefisien *axis* model *cylinder* yang didapatkan. *Max radius* dan *min radius* berkaitan dengan jari-jari model. Sama seperti segmentasi model *plane* dan *sphere,* nilai *threshold* mempengaruhi hasil segmentasi dan waktu segmentasi.

**Tabel 4.5** Hasil segmentasi model *cylinder*

| No | Informasi | *Threshold* | Hasil Segmentasi |
|---|---|---|---|
| 1 | *Point*: 931299<br>*Segmentation Time*(s) : 73,365<br>*Best Inliers*: 25168 (2,7%) | 0,1 |  |

| No | Informasi | *Threshold* | Hasil Segmentasi |
|---|---|---|---|
| 2 | *Point*: 931299 *Segmentation Time*(s) : 73,178 *Best Inliers*: 116269 (12,48%) | 0,3 |  |
| 3 | *Point*: 931299 *Segmentation Time*(s) : 72,938 *Best Inliers*: 375479 (40,31%) | 0,5 |  |

Berdasarkan tabel 4.5, dengan jumlah *point* yang sama, nilai *threshold* yang sesuai adalah 0,3, walaupun dengan nilai *threshold* 0,5 bidang cylinder yang didapatkan lebih banyak, namun model lain seperti *plane* yang terdeteksi sebagai *inliers*. Sedangkan pada waktu segmentasi, dari variasi nilai *threshold* tersebut, didapatkan waktu yang hampir sama.

## 4.2 Pengujian Jenis Visualisasi

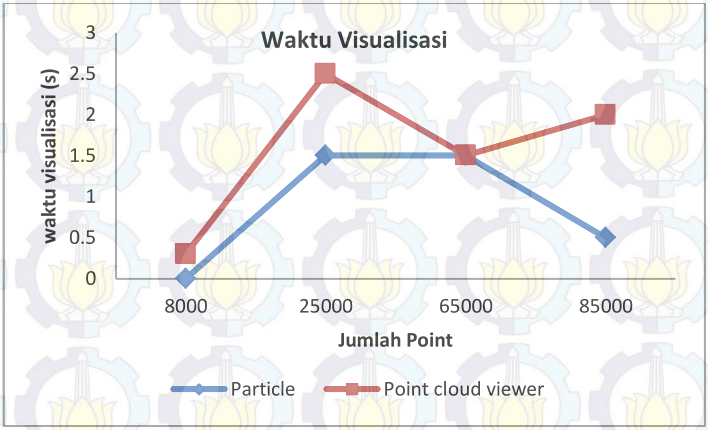
Pada tugas akhir ini disusun rancangan dua buah jenis visualisasi untuk mendapatkan metode terbaik dari proses visualisasi hasil segmentasi objek model. Ada dua jenis visualisasi. Pertama dengan menggunakan *Particle System* dan kedua dengan Point Cloud Viewer. Pada pengujian sebanyak lima kali pada masing jumlah *point* dan jenis visualisasi maka dapat didapatkan bahwa waktu tempuh pada proses segmentasi dengan kedua jenis visualisasi diatas adalah hampir sama,

walaupun visualisasi dengan Point Cloud Viewer melalui proses menyimpan *gameobject* pada *storage*. Hasil pengujian tersebut dapat dilihat pada tabel 4.6.

**Tabel 4.6** Waktu proses segmentasi dan visualisasi

| No | Jumlah *Point* | *Particle (s)* | Point Cloud Viewer (*s*) |
|---|---|---|---|
| 1 | 7830 | 0,26379744 | 0,56103858 |
| 2 | 245130 | 26,41036 | 27,80112 |
| 3 | 645711 | 49,608698 | 49,709412 |
| 4 | 838071 | 66,904486 | 68,651786 |
| 5 | 1986735 | 173,3752 | 174,5148333 |

Apabila dikorelasikan dengan waktu segmentasi pada tabel 4.2 maka didapatkan grafik pengaruh jumlah point terhadap waktu visualisasi dari kedua jenis visualisasi yang digunakan.




**Gambar 4.4** Grafik pengaruh jumlah *point* terhadap waktu visualisasi

Dari grafik pada gambar 4.4, waktu visualisasi dari *Particle System* cenderung lebih cepat daripada Point Cloud Viewer dengan data masukan point 8000 *point* hingga 85000 *point*. Jika diamati hasil visualisasi, maka visualisasi dengan Point Cloud Viewer lebih mudal

untuk melakukan pengamatan terhadap hasil segmentasi karena terdiri dari beberapa *gameobject.*

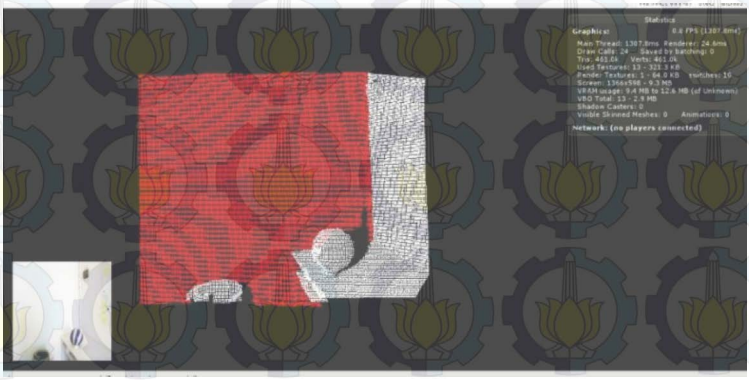
Pada *multiframe*, besar FPS (*frame perseconds*) dari kedua jenis visualisasi ini hampir sama, yakni 0,5 hingga 0,7 pada visualisasi secara sinkronous. Data masukan dari pengujian jenis visualisasi dari *multiframe* sebanyak 76800 *point.*

## 4.3 Pengujian Segmentasi *Point Cloud Multiframe*

Pada pengujian ini dilakukan untuk mengetahui performa dari perangkat lunak yang telah direalisasikan dalam melakukan proses segmentasi dan visualisasi data masukan *point cloud multiframe*.

### 4.3.1 Pengujian Data Masukan *Multiframe* secara *Real Time*

Jumlah data *point cloud* dari masukan *multiframe* adalah sama setiap *frame* karena bergantung pada resolusi peta kedalaman yang dihasilkan oleh kamera Kinect. Pada tahap pengujian ini, data *point cloud* yang disegmentasi dan ditampilkan berjumlah 76800 *point*. Setiap *point* tersebut memiliki elemen *x, y,* dan *z*. Lingkungan nyata yang diproses berada pada jarak 1-2 meter dari kamera Kinect dengan obyek yang sama. Pada pengujian dengan visualisasi secara sinkronous, didapatkan nilai FPS antara 0,5 hingga 1,1. Divergensi dari FPS pada pengujian ini diakibatkan waktu proses segmentasi yang berbeda-beda dari tiap *frame* walaupun dengan parameter yang sama.




**Gambar 4.5** Hasil visualisasi secara sinkronous dengan data masukan *multiframe*

Hasil *screen capture* dari visualisasi secara sinkronous dengan data masukan *multiframe* secara *real-time* dapat dilihat pada gambar 4.5. Sedangkan pada pengujian dengan visualisasi secara Asinkronous besar FPS yang didapatkan berkisar antara 0,9 hingga 1,7.

(a) (b)

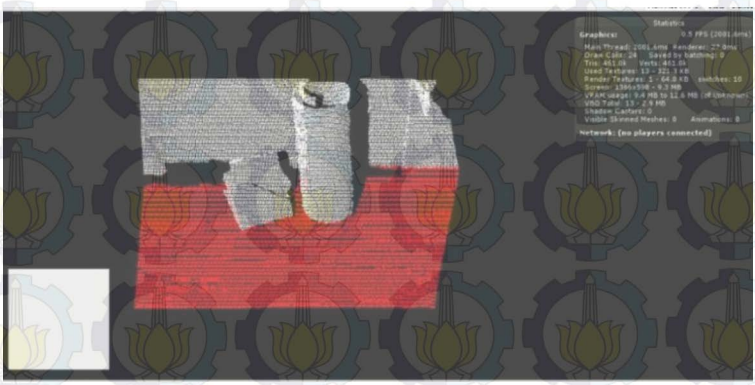
**Gambar 4.6** (a) Hasil visualisasi secara asinkronous dengan data masukan *multiframe,* (b) *Inliers* yang tidak sesuai dengan model

Visualisasi secara asinkronous memiliki FPS yang lebih tinggi daripada sinkronous, namun terdapatkan *inliers* yang tidak sesuai dengan model. Kondisi ini dapat dilihat pada gambar 4.6 (b), dimana terdapat *inliers* yang tidak sesuai dengan model yang diharapkan, berbeda dengan hasil visualisasi pada gambar 4.6 (a). Kondisi ini disebabkan oleh data *inliers* yang ditampilkan masih menggunakan data hasil segmentasi yang lama akibat proses segmentasi dari *frame* tersebut yang belum selesai.

### 4.3.2 Pengujian Data Masukan *Multiframe* Hasil Rekaman

Data rekam yang digunakan pada tahap pengujian ini memiliki 76800 *point* dengan setiap *point* memiliki elemen *x, y*, dan *z*. Pada masukan hasil rekaman, untuk visualisasi secara sinkronous baik dengan Point Cloud Viewer atau *Particles System* didapatkan FPS sebesar 0,5 hingga 0,7 sesuai dengan FPS dengan data masukan secara *stream real-time*. Data yang disimpan dalam *file* binary hanya berisikan data *depth* sehingga tidak terlihat *view color* dari lingkungan yang telah direkam. Kondisi ini dapat dilihat pada gambar 4.7. Pada pengujian ini,

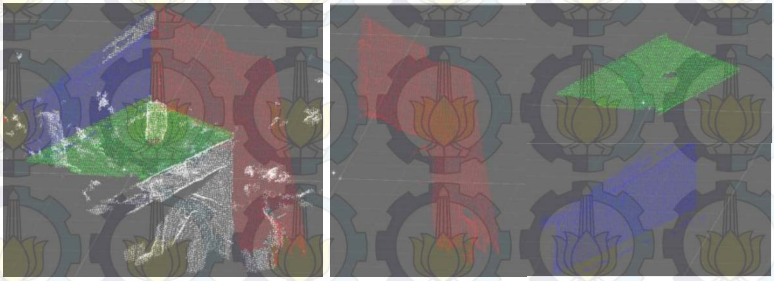
parameter *playback speed* diatur sebesar 0,0333 detik sehingga terdapat beberapa *frame* yang dilewati.



**Gambar 4.7** Hasil visualisasi proses segmentasi data masukan *file* Binary

## 4.4 Pengujian Alur Multisegmentasi

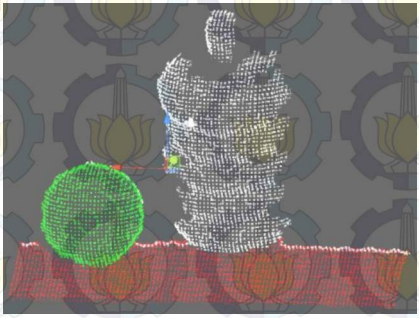
Pengujian alur multisegmentasi pada data masukan *singleframe* dilakukan dengan dua data masukan. Pengujian pertama dilakukan pada data *point cloud* dengan 245130 *point*. Model segmentasi yang digunakan adalah bidang *plane* dengan dilakukan proses segmentasi tiga kali. Besar *threshold* yang digunakan pada seluruh proses segmentasi adalah 0,5 dan besar probabilitas ditemukannya model 0,99.



(a) (b)

**Gambar 4.8** (a) Hasil pengujian multisegmentasi *plane-plane-plane,* (b) Model-model yang tersegmentasi

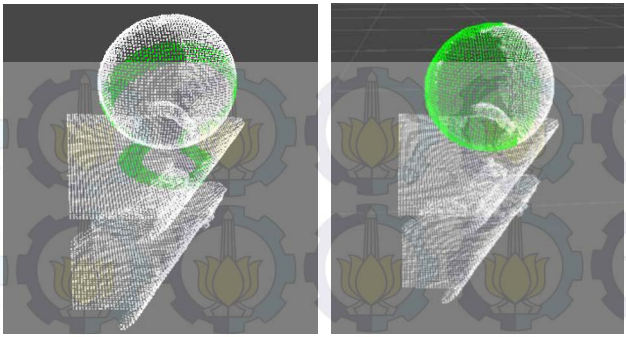
Berdasarkan gambar 4.8 (a), hasil segmentasi model *plane* yang pertama ditandai dengan *point* berwarna merah. Hasil tersebut merupakan model *plane* terbaik dari keseluruhan model *plane* pada data masukan *point cloud*. Warna hijau merupakan hasil segmentasi model plane kedua. Hasil tersebut diperoleh dari proses segmentasi dari data *point cloud* yang merupakan *outliers* dari hasil segmentasi pertama. Warna biru merupakan hasil segmentasi model *plane* yang ketiga. Sedangakan warna putih adalah *outliers* dari proses segmentasi ketiga.



**Gambar 4.9** Hasil pengujian multisegmentasi *plane-sphere*

Pengujian kedua dilakukan dengan data *point cloud* dengan 97893 *point*. Besar *threshold* yang digunakan adalah 0,01 dan probabilitas ditemukan model *plane* 0,99. Sedangkan pada model *sphere,* probabilitas ditemukan model sebesar 0,5 dan *radius* maksimum 0,1. Pada gambar 4.9, model *plane* yang tersegmentasi ditandai warna merah sedangkan model *sphere* ditandai warna hijau. Proses segmentasi model *sphere* dilakukan setelah segmentasi *plane* memberikan hasil yang bagus pada model *sphere*. Hal ini dapat dibandingkan pada gambar 4.10 (a) dan 4.10 (b). Hasil yang kurang memuaskan pada segmentasi bidang *sphere* diakibatkan oleh pengambilan *point* sampel secara acak karena sangat dimungkinkan semua *point* sampel yang diambil dalam satu bidang plane atau dalam satu garis lurus.

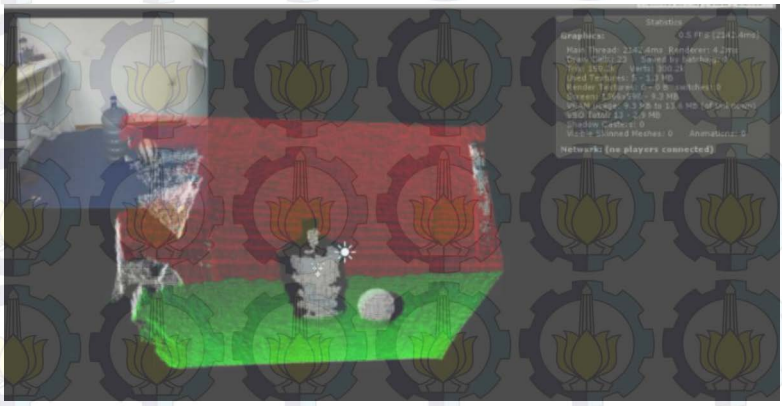
Pada pengujian multisegmentasi pada data masukan *multiframe* didapatkan FPS berkisar antara 0,4 hingga 0,6 *fps,* untuk proses segmentasi *plane-plane.* Sedangkan pada proses segmentasi *plane-plane-plane* didapatkan FPS dibawah 0,2 *fps.*



(a) (b)

**Gambar 4.10** (a) Hasil segmentasi *sphere,* (b) Hasil multisegmentasi *plane-sphere*

Proses multisegmentasi bekerja lebih lambat pada segmentasi pertama karena data *point cloud* yang diolah lebih banyak, proses segmentasi selanjutnya berjalan lebih cepat karena data masukan merupakan *outlier* dari hasil sebelumnya. Jika segmentasi pertama mampu mengeliminasi lebih dari 75% data *point cloud,* maka *fps* yang dihasilkan lebih besar akibat probabilitas *inliers* proses tersebut besar. Hasil *screen capture* dari proses visualisasi alur multisegmentasi dapat dilihat pada gambar 4.11.



**Gambar 4.11** Multisegmentasi pada *stream* Kinect secara *real-time*

# DAFTAR PUSTAKA


[1] Budi, Wahyu Setya. “Raycasting In Three-Dimentional Augmented Reality”. Publikasi Tugas Akhir. Surabaya: Institut Teknologi Sepuluh Nopember, 2014.

[2] Rusu, Radu Bogdan. “3D Perception. 50% better, Point Cloud Library”. Publikasi Willow Garage, 2010.

[3] Sitek et al. "Tomographic Reconstruction Using an Adaptive Tetrahedral Mesh Defined by a Point Cloud" IEEE Trans. Med. Imag. p.25, 2006.

[4] Tombari, F. D. Holz. “Welcome & Intoduction Point Cloud Library”. Prosiding IEEE International Conference on Robotics and Automation (ICRA), 2013.

[5] Rusu, Radu Bogdan, Steve Cousins. “3D is here: Point Cloud Library (PCL)”. Prosiding IEEE International Conference on Robotics and Automation (ICRA), 2011.

[6] “Point Cloud Library Documentation”. <URL: http://pointclouds.org/documentation/>, Mei, 2015

[7] M.A Fischler, R. C. Bolles. “Random Sample Consensus : A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography”. Proseding SRI International, 1981

[8] Ashari, Riska Wahyu. “Fixed Point Augmented Reality Menggunakan Kinect”. Publikasi Tugas Akhir. Surabaya: Institut Teknologi Sepuluh Nopember, 2013.

[9] MacCormick, John. “How does the Kinect Work?”. Presentasi Dickinson College United States of America.

[10] “Particle Systems”. <URL: http://docs.unity3d.com/Manual/ ParticleSystems.html>, Mei, 2015.

[11] Reeves, William T. “Particle Systems: A Technique for Modeling a Class of Fuzzy Objects”. ACM Transactions on Graphics, April 1983.

[12] “Particle System”. <URL: http://graphics.wikia.com/wiki/ Particle_system>, Mei, 2015.

[13] “Point Cloud Free Viewer”. <URL: https://www.assetstore.unity3d.com/en/#!/content/19811>, Mei, 2015.

[14] Izadi, Zahram. “KinectFusion: Real-time 3D Reconstruction and Interaction Using a Moving Depth Camera”. Microsoft Research. Cambridge University, 2011.
[15] “Polygon File Introduction”. <URL: http://paulbourke.net/dataformats/ply/>, Mei, 2015.
[16] “The PLY File Format”. <URL: http://www.cc.gatech.edu/projects/large_models/ply.html>, Mei, 2015.
[17] Weisstein, Eric W. "Plane, MathWorld--A Wolfram Web Resource”. <URL: http://mathworld.wolfram.com/Plane.html>, Mei, 2015.
[18] Schmitt, Stephen R. “Center and Radius of a Sphere from Four Points”. <URL:http://www.abecedarical.com/zenosamples/zs_sphere4pts.html>, Mei, 2015.
[19] Garcia, Sergion. “Fitting Primitive Shapes to Point Clouds for Robotic Grasping”. Publikasi Thesis : KTH Computer Science and Communication, 2009

# BAB 5
# PENUTUP

## 5.1 Kesimpulan

Setelah melalui perancangan aplikasi, implementasi, dan pengujian, akhirnya diperoleh beberapa kesimpulan :

1. Data *point cloud* dapat dipisahkan antara model primitif tiga dimensi satu dengan lain melalui proses segmentasi dengan metode *Random Sample Consensus*. Pada pengujian segmentasi model *plane, sphere,* dan *cylinder,* data *point cloud* dapat dipisahkan antara data *inliers*, data yang sesuai model, dan data *outliers* atau data selain *inliers*.
2. Nilai *threshold*, jumlah *point*, dan presentase *inliers* dalam *point cloud* mempengaruhi waktu segmentasi. Parameter *threshold* juga mempengaruhi kesesuaian hasil segmentasi dengan model yang sebenarnya.
3. Pada pengujian segmentasi tunggal model *plane, frame per second* yang dihasilkan sebesar 0,5 hingga 1,1 pada visualisasi secara sinkronous, sedangkan pada asinkronous diperoleh sebesar *frame per second* 0,9 hingga 1,7.
4. Pada pengujian multisegmentasi model *plane*, *frame per second* yang dihasilkan lebih kecil daripada segmentasi tunggal model *plane*.

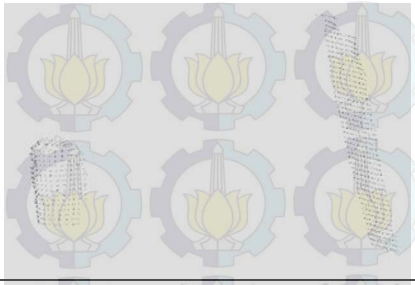
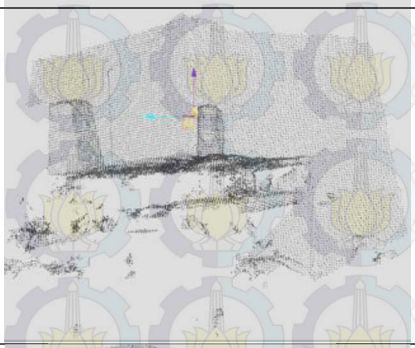
## 5.2 Saran

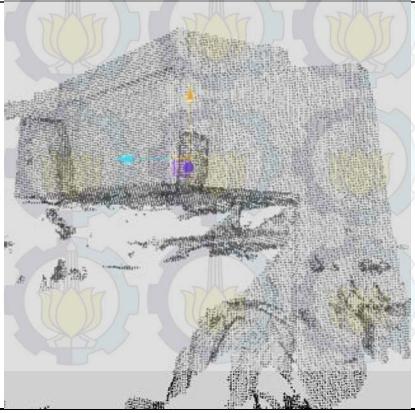
Untuk pengembangan lebih lanjut mengenai tugas akhir ini, penulis memberikan saran:

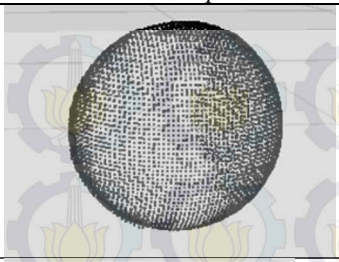
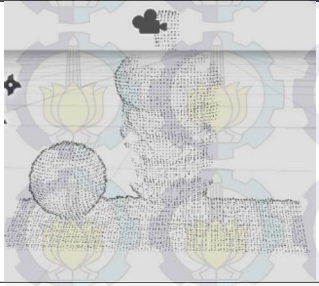
a. Memanfaatkan dan mengembangkan proses segmentasi untuk pengembangan riset tentang *augmented reality* kedepan misalnya proses identifikasi atau pengenalan model atau objek yang lain.
b. Melakukan proses segmentasi dengan perangkat atau sensor yang lain.

# LAMPIRAN

**Visualisasi *file* PLY yang Digunakan dalam Pengujian**

| Model | Jumlah *point cloud* | Visualisasi *point cloud* |
|---|---|---|
| *Plane* | 7830 |  |
| *Plane* | 245130 |  |
| *Plane* | 645711 |  |

| Model | Jumlah *point cloud* | Visualisasi *point cloud* |
|---|---|---|
| *Plane* | 838071 |  |
| *Plane* | 1986735 |  |
| *Plane* | 263934 |  |

| Model | Jumlah *point cloud* | Visualisasi *point cloud* |
|---|---|---|
| *Sphere* | 55880 |  |
| *Sphere* | 132952 |  |
| *Sphere* | 97893 |  |
| *Cylinder* | 931299 |  |

**Hasil Pengujian Segmentasi Model *Plane Singleframe***

**Pengujian 1.** Pengaruh jumlah *point* terhadap waktu segmentasi

*Threshold* : 0.5

Keterangan :
P1 : Pengujian Pertama
P2 : Pengujian Kedua
P3 : Pengujian Ketiga
P4 : Pengujian Keempat
P5 : Pengujian Kelima

| Jumlah Point | P1 | P2 | P3 | P4 | P5 | *Mean* |
|---|---|---|---|---|---|---|
| 7830 | 0.23 | 0.25 | 0.23 | 0.23 | 0.23 | 0.23 |
| 245130 | 24.88 | 25.51 | 24.50 | 25.07 | 25.15 | 25.02 |
| 645711 | 48.68 | 48.73 | 48.35 | 47.94 | 48.65 | 48.47 |
| 838071 | 66.64 | 66.73 | 66.55 | 66.86 | 66.05 | 66.56 |
| 1986735 | 173.72 | 174.72 | 178.29 | 172.78 | 172.29 | 174.36 |
| 263934 | 26.30 | 25.54 | 24.91 | 25.17 | 25.78 | 25.54 |

**Pengujian 2.** Pengaruh *threshold* terhadap waktu segmentasi

*Jumlah Point* : 245130

| *Threshold* | P1 | P2 | P3 | P4 | P5 | *Mean* |
|---|---|---|---|---|---|---|
| 5 | 10.29 | 10.20 | 10.20 | 10.05 | 10.02 | 10.15 |
| 1 | 19.67 | 19.70 | 19.81 | 19.58 | 19.72 | 19.7 |
| 0.5 | 24.88 | 25.51 | 24.50 | 25.07 | 25.15 | 25.02 |
| 0.1 | 353.35 | 343.17 | 340.6 | 342.91 | 352.57 | 346.52 |

**Hasil Pengujian Segmentasi Model *Sphere Singleframe***

**Pengujian 1.** Pengaruh jumlah *point* dan *threshold* terhadap waktu segmentasi

Threshold : 0.01

| Jumlah Point | P1 | P2 | P3 | P4 | P5 | *Mean* |
|---|---|---|---|---|---|---|
| 55880 | 0.81 | 0.81 | 0.81 | 0.82 | 0.83 | 0.82 |
| 132952 | 1.85 | 1.866 | 1.85 | 1.87 | 1.85 | 1.88 |

Threshold : 0.05

| Jumlah Point | P1 | P2 | P3 | P4 | P5 | *Mean* |
|---|---|---|---|---|---|---|
| 55880 | 0.81 | 0.81 | 0.81 | 0.81 | 0.81 | 0.81 |
| 132952 | 1.88 | 1.85 | 1.87 | 1.86 | 1.86 | 1.86 |

**Hasil Pengujian Segmentasi Model *Cylinder Singleframe***

**Pengujian 1.** Pengaruh *threshold* terhadap waktu segmentasi

Jumlah *point* : 931299

| *Threshold* | P1 | P2 | P3 | P4 | P5 | *Mean* |
|---|---|---|---|---|---|---|
| 0.1 | 72.55 | 73.6 | 74.24 | 73.31 | 73.10 | 73.36 |
| 0.3 | 73.74 | 72.58 | 73.30 | 73.22 | 73.03 | 73.17 |
| 0.5 | 73.67 | 73.34 | 72.64 | 72.49 | 72.54 | 72.94 |

**Hasil Pengujian Visualisasi Model *Plane Singleframe***

**Pengujian 1.** Waktu total segmentasi dan visualisasi dari visualisasi dengan Particle System

*Threshold* : 0.5

| Jumlah *Point* | P1 | P2 | P3 | P4 | P5 | *Mean* |
|---|---|---|---|---|---|---|
| 7830 | 0.26 | 0.26 | 0.25 | 0.26 | 0.26 | 0.26 |
| 245130 | 27.06 | 26.14 | 26 | 26.53 | 26.29 | 26.41 |
| 645711 | 51.83 | 48.62 | 50.40 | 48.36 | 48.81 | 49.60 |
| 838071 | 66.81 | 66.06 | 66.59 | 67.50 | 67.55 | 66.90 |
| 1986735 | 178.41 | 169.31 | 170.20 | 175.56 | | 173.37 |

**Pengujian 2.** Waktu total segmentasi dan visualisasi dari visualisasi dengan Point Cloud Viewer

*Threshold* : 0.5

| Jumlah *Point* | P1 | P2 | P3 | P4 | P5 | *Mean* |
|---|---|---|---|---|---|---|
| 7830 | 0.55 | 0.54 | 0.56 | 0.56 | 0.56 | 0.56 |
| 245130 | 28.87 | 27.82 | 27.66 | 27.55 | 27.09 | 27.80 |
| 645711 | 49.33 | 50.17 | 49.38 | 49.87 | 49.78 | 49.70 |
| 838071 | 69.39 | 67.79 | 68.35 | 69.15 | 68.55 | 68.65 |
| 1986735 | 174.58 | 173.96 | 174.99 | 174.75 | | 174.57 |