

# **Pemetaan Ruangan Menggunakan Ar.Drone Dengan Metode *LSD-SLAM***

**SKRIPSI**

**KEMINATAN TEKNIK KOMPUTER**

Untuk memenuhi sebagian persyaratan  
Memperoleh gelar Sarjana Komputer

Disusun oleh:  
Yanottama Oktabrian  
NIM: 135150301111035



PROGRAM STUDI TEKNIK INFORMATIKA  
JURUSAN TEKNIK INFORMATIKA  
FAKULTAS ILMU KOMPUTER  
UNIVERSITAS BRAWIJAYA  
MALANG  
2020

## DAFTAR ISI

DAFTAR ISI .....	ii
DAFTAR GAMBAR .....	iv
DAFTAR TABEL .....	v
BAB 1 PENDAHULUAN .....	1
1.1    Latar Belakang .....	1
1.1.1    Rumusan Masalah .....	2
1.1.2    Tujuan .....	2
1.1.3    Manfaat .....	2
1.1.4    Batasan Masalah .....	2
1.1.5    Sistematika Pembahasan.....	3
BAB 2 LANDASAN KEPUSTAKAAN .....	4
2.1    Kajian Pustaka.....	4
2.2    Dasar Teori.....	4
2.2.1    Unmanned Aerial Vehicle (UAV) Quadcopter.....	5
2.2.2    Simultaneous Localization And Mapping (SLAM) .....	8
2.2.3    Large Scale Direct Monocular SLAM (LSD-SLAM) .....	9
BAB 3 METODOLOGI .....	11
3.1    Metode Penelitian .....	11
3.2    Analisis Kebutuhan .....	11
3.3    Perancangan Sistem .....	12
3.4    Implementasi Sistem .....	12
3.5    Pengujian dan Analisis Sistem .....	12
3.6    Penarikan Kesimpulan dan Saran .....	12
BAB 4 REKAYASA KEBUTUHAN .....	13
4.1    Gambaran Umum Sistem .....	13
4.2    Analisis Kebutuhan Sistem .....	14
4.2.1    Kebutuhan Pengguna .....	20
4.2.2    Kebutuhan Sistem .....	20
4.3    Kebutuhan Fungsional.....	14

4.4 Kebutuhan Non-Fungsional.....	18
4.4.1 Karakteristik Pengguna .....	18
4.4.2 Lingkungan Operasi.....	18
4.4.3 Asumsi dan Ketergantungan .....	19
4.4.4 Batasan Perancangan dan Implementasi.....	19
DAFTAR PUSTAKA.....	83

## **DAFTAR GAMBAR**

Gambar 2.1 Pergerakan motor quadcopter.....	5
Gambar 2.2 Gerakan roll quadcopter .....	6
Gambar 2.3 Gerakan pitch quadcopter .....	6
Gambar 2.4 Gerakan yaw quadcopter .....	7
Gambar 2.5 Gerakan throttle quadcopter .....	7
Gambar 2.6 Parrot AR.Drone .....	8
Gambar 2.7 Peta SLAM .....	9
Gambar 2.8 Komponen LSD-SLAM.....	10
Gambar 3.1 Alur Metodologi Penelitian .....	11
Gambar 4.1 Gambaran Umum Sistem .....	13
Gambar 4.2 Analisis Kebutuhan Sistem .....	14
Gambar 4.3 Parrot Ar.Drone 2.0 .....	21

## **DAFTAR TABEL**

Tabel 4.1 Lingkungan Operasi .....	20
Tabel 4.2 Tampilan Antarmuka Sistem .....	20
Tabel 4.3 Parrot Ar.Drone 2.0 .....	21
Tabel 4.4 Spesifikasi Kamera Ar.Drone 2.0 .....	22
Tabel 4.5 Spesifikasi Komputer GCS.....	22
Tabel 4.6 Spesifikasi Wi-Fi Adapter.....	23
Tabel 4.7 Sistem Operasi.....	23
Tabel 4.8 Sistem Operasi.....	24
Tabel 4.9 Ardrone Autonomy.....	24
Tabel 4.10 TUM Ardrone.....	24
Tabel 4.11 Wi-Fi .....	16
Tabel 4.12 Inisialisasi.....	16
Tabel 4.13 Tampilan Kamera Depan .....	17
Tabel 4.14 Tampilan LSD-SLAM.....	17
Tabel 4.15 Tombol Kontrol Drone.....	17
Tabel 4.16 Mengakuisisi Citra .....	15
Tabel 4.17 Navigasi Dalam Ruangan .....	15
Tabel 4.18 Melakukan Deteksi Kesalahan.....	15
Tabel 4.19 Karakteristik Pengguna.....	18
Tabel 4.20 Lingkungan Operasi .....	18
Tabel 4.21 Asumsi dan Ketergantungan .....	19
Tabel 4.22 Batasan Perancangan dan Implementasi .....	19

# BAB 1 PENDAHULUAN

## 1.1 Latar Belakang

Perkembangan teknologi yang pesat menjadikan manusia lebih mudah dalam mengerjakan berbagai rutinitas. Rutinitas-rutinitas ini mulai dari pekerjaan yang kecil seperti menyapu lantai hingga melihat-lihat keadaan sekitar. Salah satu cara manusia dalam mengenali lingkungan sekitar adalah dengan pemetaan. Pemetaan baik skala kecil seperti pemetaan ruangan ataupun skala besar seperti pemetaan geologis. Masalah pemetaan ini muncul karena keterbatasan pandangan manusia serta cara menginterpretasikan apa yang dilihat ke dalam suatu model. Model-model inilah yang akan digunakan untuk mempresentasikan keadaan sebuah lingkungan. Pada robotika permasalahan yang sama juga terjadi. Permasalahan ini adalah pembuatan peta dimana robot berada, serta bagaimana robot bernaligasi di lingkungan tersebut dengan tetap memantau posisi dan orientasi robot.

SLAM (*Simultaneous Localization And Mapping*) merupakan sebuah konsep yang dapat menyelesaikan masalah krusial dalam dunia robotika. Permasalahan pertama adalah pemetaan (*mapping*) yaitu penciptaan sebuah peta dari lingkungan tempat robot berada. Permasalahan kedua lokalisasi (*localisation*) merupakan navigasi robot dalam lingkungan ini menggunakan peta yang telah dibangun dan dalam waktu bersamaan melacak pergerakan posisi dan orientasi robot. Dengan mengatasi permasalahan pada SLAM kemungkinan dalam otomatisasi robot akan terbuka lebar (Yap, et al., 2016).

*Quadcopter* dapat dikendalikan secara bebas dengan memanfaatkan kombinasi baling-baling yang terpasang pada *quadcopter*. *Quadcopter* melakukan berbagai gerakan dengan menaikkan atau menurunkan kecepatan masing-masing motor sehingga *quadcopter* dapat berbelok, berputar, naik, turun, dan lain-lain (Hernandez-Martinez, et al., 2015). *Quadcopter* biasanya bergantung pada sensor-sensor untuk menuju tujuan yang diinginkan. Pada *quadcopter* pemula biasanya digunakan metode melihat langsung untuk mengontrol posisi dan orientasi *quadcopter*. Pada *quadcopter* yang lebih canggih biasanya digunakan *GLOBAL POSITIONING SYSTEM (GPS)* untuk navigasi waypoint. Teknologi ini memungkinkan *quadcopter* untuk terbang secara otomatis ke titik yang diinginkan. Sistem ini juga dapat mengatur kecepatan, ketinggian, dan dimana *quadcopter* harus terbang (Omega, 2017).

Penelitian ini merupakan implementasi dari pemetaan menggunakan *LSD-SLAM* pada *quadcopter Parrot AR.Drone 2.0*. *Quadcopter AR.Drone* dipilih karena memiliki sifat *open source* yaitu perangkat lunak dari *quadcopter* ini dapat dimodifikasi dan dikembangkan sesuai kebutuhan pengguna. Selain bersifat *open source* *AR.Drone* juga memiliki harga yang relatif terjangkau. *Quadcopter AR.Drone 2.0* memiliki dua buah sensor kamera. Kamera pertama letaknya berada di bagian depan *quadcopter*. Kamera kedua letaknya berada di bawah *quadcopter* dan menghadap ke bawah secara vertikal (Parrot, 2018). Kamera-kamera inilah yang nantinya akan digunakan untuk mendeteksi lingkungan ruang sekitar.

Dengan menggabungkan antara *quadcopter* dan algoritma LSD-SLAM diharapkan *quadcopter* dapat memetakan lingkungan tempat *quadcopter* saat ini berada.

### **1.1. Rumusan Masalah**

Seperti uraian latar belakang sebelumnya, maka perumusan masalah yang akan dibahas adalah sebagai berikut

1. Bagaimana cara pemanfaatan kamera *AR.Drone 2.0* sebagai perangkat dalam pemetaan ruangan?
2. Bagaimana implementasi metode *LSD SLAM* dalam sistem kendali *AR.Drone 2.0*?
3. Bagaimana tingkat keakuratan metode *LSD SLAM* dalam pemetaan ruangan dengan menggunakan *AR.Drone 2.0*?

### **1.2. Tujuan**

Tujuan dari penelitian ini menjawab rumusan masalah yang ada seperti

1. Mengetahui fungsi kamera pada *AR.Drone 2.0* sebagai perangkat pemetaan dalam ruangan.
2. Mengimplementasikan fungsi metode *LSD-SLAM* pada penelitian pemetaan ruangan menggunakan kamera *AR.Drone 2.0*.
3. Mengukur keakuratan metode *LSD-SLAM* pada pemetaan ruangan menggunakan *AR.Drone 2.0*.

### **1.3. Manfaat**

Manfaat yang diperoleh dengan adanya penelitian ini adalah

1. Membantu pengguna dalam menggunakan kamera *AR.Drone 2.0* sebagai perangkat pemetaan ruangan.
2. Dapat membantu memahami implementasi *LSD-SLAM* pada pemetaan ruangan menggunakan *AR.Drone 2.0*.
3. Mengetahui tingkat efektivitas penggunaan metode *LSD-SLAM* pada pemetaan ruangan menggunakan *AR.Drone 2.0*.

### **1.4. Batasan Masalah**

Adapun batasan masalah agar tidak menyimpang dari perumusan masalah adalah sebagai berikut

1. *Quadcopter* yang digunakan adalah Parrot *AR.Drone 2.0*.
2. Sistem yang dibuat diuji di dalam ruangan.
3. Algoritma SLAM (*Simultaneous Localization And Mapping*) yang digunakan adalah LSD (*Large Scale Direct Monocular*) SLAM.

## **1.5. Sistematika Pembahasan**

Sistematika penulisan bertujuan sebagai penjelasan umum dari bagian-bagian bab yang ada dalam penelitian ini agar memudahkan pembaca dalam mengikuti alur pembahasan penelitian. Adapun sistematika penulisan adalah sebagai berikut

### **BAB I: Pendahuluan**

Pada bab I memuat latar belakang permasalahan, rumusan masalah, pembatasan masalah, tujuan, manfaat, dan sistematika penulisan.

### **BAB II: Dasar Teori**

Pada bab II berisi tentang penjelasan teori-teori dasar yang menjadi acuan dalam melaksanakan penerapan penelitian ini. Beberapa penjelasannya dikutip dari beberapa studi literatur seperti *paper*, buku, dan lain-lain.

### **BAB III: Metodologi Penelitian**

Pada bab III ini berisi tentang metodologi penelitian beberapa hal yang akan dibahas pada bab ini di antaranya analisis kebutuhan, perancangan sistem, implementasi, pengujian serta analisisnya, dan yang terakhir berupa penarikan kesimpulan dan pemberian saran dari penelitian yang akan dilakukan.

### **BAB IV: Analisis Kebutuhan**

Pada Bab IV berisi penjelasan mengenai kebutuhan-kebutuhan yang terkait dalam penelitian ini seperti kebutuhan pengguna, Kebutuhan sistem ,kebutuhan perangkat lunak, dan kebutuhan perangkat keras.

### **BAB V: Perancangan sistem dan Implementasi**

Pada bab V berisi penjelasan mengenai perancangan dan implementasi sistem, seperti diagram komunikasi sistem dan diagram alur kerja sistem.

### **BAB VI: Pengujian dan Analisis**

Pada bab VI berisi penjelasan mengenai proses pengujian dimulai dari tujuan pengujian, prosedur pengujian, pelaksanaan pengujian, hasil pengujian dan dilakukan analisis hasil pengujian yang sudah dilakukan.

### **BAB VII: Kesimpulan dan Saran**

Pada bab VII berisi penjelasan kesimpulan yang diperoleh dari hasil penelitian yang dilakukan. Pada bab ini juga terdapat saran-saran yang dapat digunakan sebagai dasar untuk penelitian selanjutnya.

## BAB 2 LANDASAN KEPUSTAKAAN

### 2.1 Kajian Pustaka

Pada tinjauan pustaka ini akan dijelaskan tentang penelitian yang sudah pernah dilakukan yang berkaitan dengan penelitian ini yaitu Pemetaan Ruangan Menggunakan *Ar.Drone* Dengan Metode *LSD-SLAM*.

Penelitian sebelumnya yang meneliti tentang *SLAM* adalah penggunaan *LSD-SLAM: Large Scale Direct Monocular SLAM* (Engel & Cremers, 2014), penelitian ini membahas tentang Simultaneous Localization and Mapping (*SLAM*) dan rekonstruksi 3D. Penelitian ini berfokus untuk menciptakan algoritma baru untuk membangun peta berskala besar. Berdasarkan hasil tes yang dilakukan pembangunan peta dapat dilakukan dengan jarak lebih dari 500 meter, dengan kedalaman rata-rata kurang dari 20 cm hingga lebih dari 10 meter.

Penelitian rujukan kedua berjudul *Probabilistic Semi-Dense Mapping from Highly Accurate Feature-Based Monocular SLAM* (Mur-Artal & D. Tardos, 2015). Penelitian ini menggunakan sistem pemetaan *Semi-Dense* pada *keyframe*. Sistem tersebut dioptimasi menggunakan berbagai pengaturan yang menjadikannya dapat menghasilkan triangulasi lokasi yang akurat. Metode tersebut mencari kesesuaian, penggabungan pengukuran dan tes kedalaman *inter-keyframe* untuk memperoleh hasil rekonstruksi yang bersih.

Penelitian rujukan ketiga Comparative Analysis of ROS-based Monocular SLAM Methods for Indoor Navigation (Buyval, et al., 2017) membahas tentang penggunaan empat metode *SLAM* pada ROS. Metode ini adalah ORB-SLAM, REMODE, LSD-SLAM dan DPPTAM. Metode-metode ini diuji pada robot beroda dan di lingkungan dalam ruang. Robot ini dilengkapi dengan kamera berlensa lebar dan beresolusi Full HD. Hasil yang diperoleh dari demonstrasi yang dilakukan didapat hasil yang bagus dalam pendekripsi obyek bervolume, sudut-sudut, halangan-halangan dan berbagai obyek lainnya.

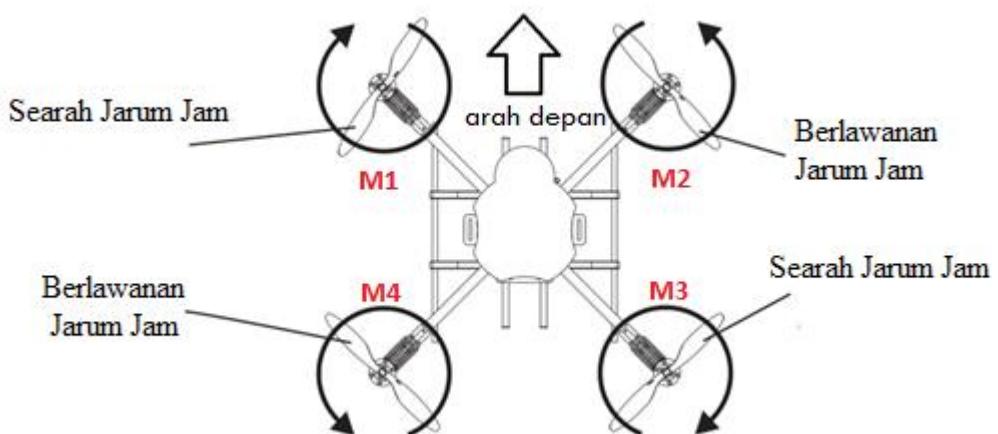
Berdasarkan beberapa penelitian di atas penulis merasa metode *SLAM* perlu dikembangkan lagi untuk diaplikasikan sebagai untuk *quadcopter Parrot AR.Drone 2.0* dipadukan dengan sensor kamera. Kombinasi ini diharapkan dapat menghasilkan sistem pemetaan menggunakan *AR.Drone*.

### 2.2 Dasar Teori

Pada bagian ini akan dibahas teori-teori pendukung yang berkaitan dengan penelitian ini, diantaranya *Unmanned Aerial Vehicle (UAV) Quadcopter*, *Simultaneous Localization And Mapping (SLAM)*, serta *Large Scale Direct Monocular Simultaneous Localization And Mapping (LSD-SLAM)*.

## 2.2.1 Unmanned Aerial Vehicle (UAV) Quadcopter

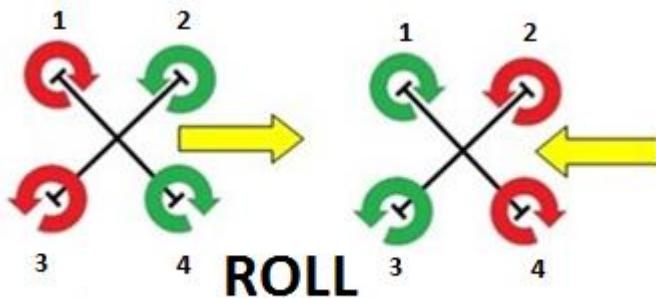
*Quadcopter* adalah robot tanpa awak berbentuk helicopter dengan empat motor dan empat baling-baling. *Quadcopter* memiliki ukuran lebih kecil dari helikopter sebenarnya sehingga sering disebut UAV *micro*. Dua baling-baling berputar searah jarum jam, dan dua lainnya berputar berlawanan arah jarum jam. Setiap motor berputar berlawanan arah terhadap motor tetangganya (M1 berlawanan dengan M2 dan M4, dst.). Motor segaris berputar searah (M1 sama dengan M3, M2 sama dengan M4). Jarak setiap motor sama antar satu dan lainnya. Gambar 2.1 Pergerakan motor *quadcopter* menunjukkan perputaran baling-baling.



Gambar 2.1 Pergerakan motor *quadcopter*

Sumber: Lawson & Logan (2017)

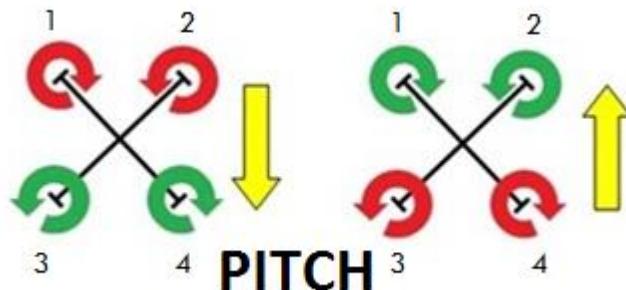
Kecepatan motor *quadcopter* dapat diatur masing-masing sehingga *quadcopter* dapat melakukan gerakan-gerakan manuver. Untuk melakukan gerakan-gerakan tersebut motor *quadcopter* berfungsi sebagai pendorong (pusher) dan penarik (puller). Untuk melakukan hover keempat motornya berputar pada kecepatan yang sama.



**Gambar 2.2 Gerakan roll quadcopter**

Sumber: Ramadhan (2015)

Gerakan *roll* merupakan arah pergerakan *quadcopter* menurut sumbu x (Romero, et al., 2014) seperti pada Gambar 2.2 Gerakan *roll quadcopter*. Ketika motor nomor 1 dan 3 menambah kecepatan dan motor nomor 2 dan 4 mengurangi kecepatan maka *quadcopter* akan bergerak ke kanan. Sebaliknya saat motor nomor 1 dan 3 mengurangi kecepatan dan motor nomor 2 dan 4 menambah kecepatan *quadcopter* akan bergerak ke kiri.

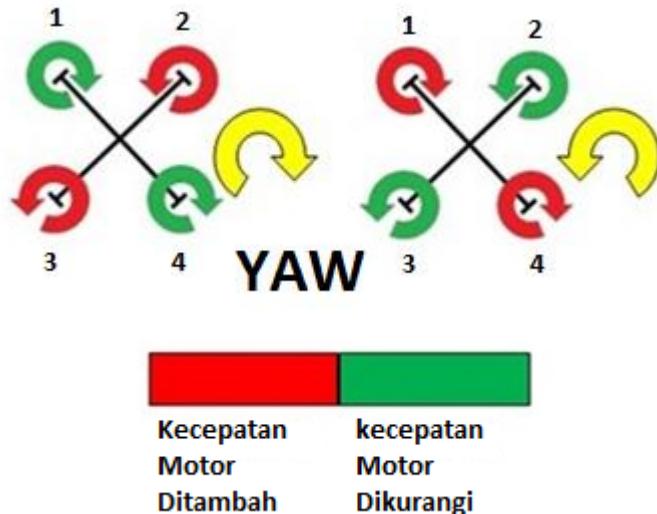


**Gambar 2.3 Gerakan pitch quadcopter**

Sumber: Ramadhan (2015)

Gerakan *pitch* merupakan pergerakan *quadcopter* menurut sumbu y (Romero, et al., 2014) seperti pada Gambar 2.3 Gerakan *pitch quadcopter*. Ketika motor

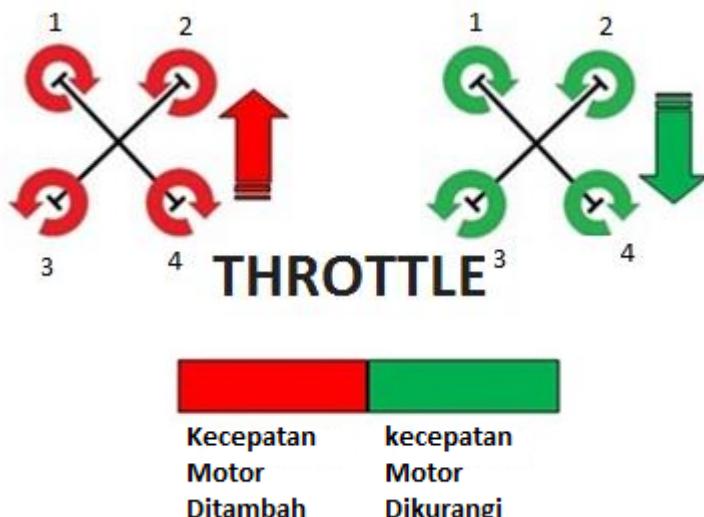
nomor 1 dan 2 menambah kecepatan dan motor nomer 3 dan 4 mengurangi kecepatan maka *quadcopter* akan bergerak ke belakang. Sebaliknya saat motor nomor 1 dan 2 mengurangi kecepatan dan motor 3 dan 4 menambah kecepatan maka *quadcopter* akan bergerak ke depan.



**Gambar 2.4 Gerakan yaw quadcopter**

Sumber: Ramadhan (2015)

Gerakan *yaw* merupakan arah pergerakan *quadcopter* menurut sumbu z (Romero, et al., 2014) seperti pada Gambar 2.4 Gerakan *yaw quadcopter*. Ketika motor nomor 1 dan 4 mengurangi kecepatan dan motor nomor 2 dan 3 menambah kecepatan maka *quadcopter* akan bergerak searah jarum jam. Sebaliknya saat motor nomor 1 dan 4 menambah kecepatan dan motor nomor 2 dan 3 mengurangi kecepatan maka *quadcopter* akan bergerak berlawanan jarum jam.



**Gambar 2.5 Gerakan throttle quadcopter**

Sumber: Ramadhan (2015)

Gerakan *throttle* merupakan pergerakan *quadcopter* keatas (naik) atau kebawah (turun) (Hernandez-Martinez, et al., 2015) seperti pada Gambar 2.5. Gerakan *throttle quadcopter*. Untuk bergerak naik maka keempat motor menambah kecepatannya. Untuk bergerak turun keempat motor mengurangi kecepatannya. Kecepatan keempat motor harus sama agar pergerakan stabil.

AR.Drone 2.0 memiliki sensor kamera depan 720p (1280 x 720 pixel) dengan lensa 93 derajat. Kamera ini mampu merekan video hingga 30 *fps* (*frame per seconds*). Sementara untuk kamera vertikal, AR.Drone menggunakan kamera QVGA (320 x 240 piksel) dengan lensa 64 derajat dan mampu merekam video hingga 60 *fps* (*frame per seconds*) (Parrot, 2018).



**Gambar 2.6 Parrot AR.Drone**

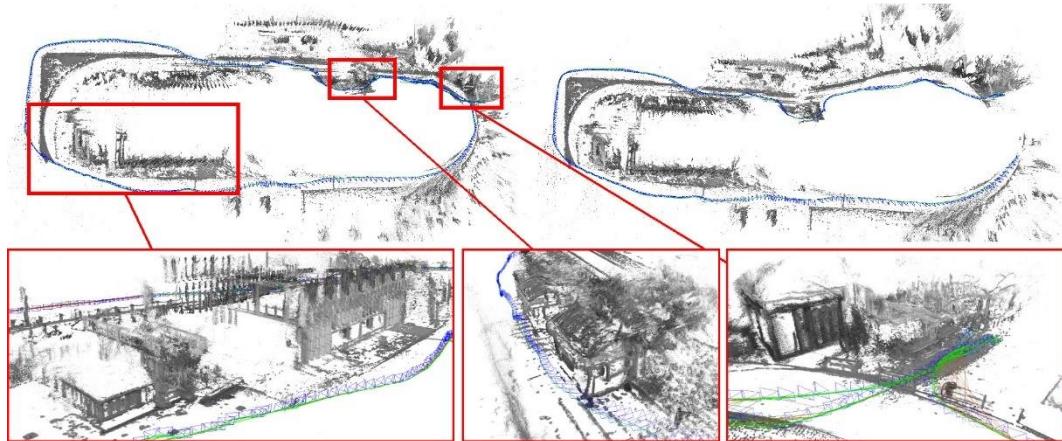
Sumber: (Parrot, 2018)

### **2.2.2 Simultaneous Localization And Mapping (SLAM)**

*Simultaneous Localization And Mapping (SLAM)* merupakan sebuah permasalahan dalam komputasi untuk membangun atau memperbaharui sebuah peta dari lingkungan yang asing (Durrant-Whyte & Bailey, 2006). Permasalahan ini muncul ketika komputasi harus berjalan dengan melacak lokasi robot pada peta tersebut. Pada *SLAM* lokasi robot ditempatkan pada sebuah peta. Dari lokasi robot tersebut robot akan mengambil kesimpulan dengan mempelajari peta (Bailey & Durrant-Whyte, 2006).

Cara kerja *visual SLAM* modern berbasis pada pelacakan suatu set poin-poin yang didapat dari *frame-frame* sebuah foto. Poin-poin yang didapat tadi kemudian digunakan untuk melacak posisi 3D nya, bersamaan dengan penggunaan perkiraan lokasi poin untuk mengkalkulasi posisi kamera yang digunakan untuk melacak poin-poin tadi. Dengan mengamati poin-poin dengan jumlah yang cukup, pemetaan dapat dilakukan untuk struktur dan juga gerakan dari obyek pemetaan. Untuk penggunaan satu buah kamera, dengan menggabungkan pengukuran poin-

poin dari beberapa *frame* foto dimungkinkan untuk mendapat model dan struktur dengan keakuratan yang tinggi (Kudan, 2016).



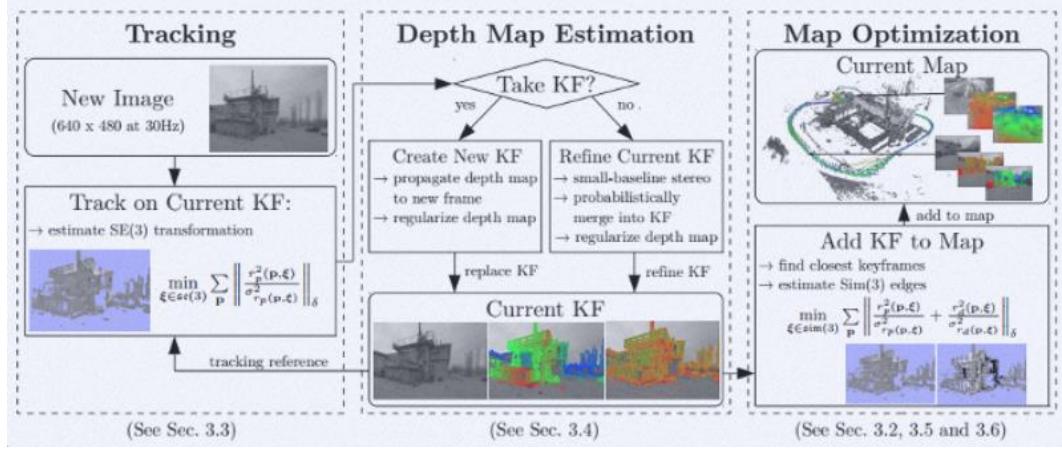
**Gambar 2.7 Peta SLAM**

**Sumber:** (Engel & Cremers, 2014)

SLAM mempunyai beberapa persyaratan kebutuhan untuk dapat dilakukan. Salah satunya adalah perangkat pengukur jarak. Perangkat ini digunakan robot untuk mengobservasi keadaan lingkungan di sekitar robot tersebut. Perangkat ini beragam, mulai dari laser, sonar hingga perangkat optik seperti kamera baik dalam format 2D maupun 3D. Perangkat-perangkat ini digunakan bergantung lokasi serta keakuratan yang ingin dicapai. Persyaratan lain adalah proses mengakuisisi data lingkungan di sekitar robot. Robot menggunakan berbagai perangkat sensor yang dimilikinya untuk menentukan posisi benda-benda di lingkungan tersebut. Proses ini membutuhkan benda-benda obyek untuk diam di tempat atau tidak bergerak. Robot tidak akan mampu untuk menentukan posisi dirinya sendiri saat obyek di sekitarnya bergerak terus-menerus. Obyek juga harus unik dan dapat dibedakan dari latar belakang lingkungan, serta harus dapat dilihat dari berbagai sudut (Maxwell, 2013).

### 2.2.3 Large Scale Direct Monocular SLAM (LSD-SLAM)

*Large Scale Direct Monocular SLAM (LSD-SLAM)* merupakan salah satu tipe *monocular SLAM* yang menggunakan satu buah lensa kamera untuk mengobservasi lingkungan. *LSD-SLAM* dikembangkan agar pembangunan peta yang konsisten dengan skala besar dari sebuah lingkungan dapat dilakukan (Engel, et al., 2014). *LSD-SLAM* bekerja dengan menggunakan intensitas dari gambar untuk pelacakan dan pembuatan peta (Engel & Cremers, 2014). Metode ini dapat digunakan untuk memetakan area dengan skala besar dan tidak memerlukan perangkat khusus. Algoritma *LSD-SLAM* mempunyai tiga komponen utama yaitu pelacakan (*tracking*), pengestimasi kedalaman peta (*depth map estimation*), serta pengoptimasi peta (*map optimisation*) seperti ditunjukkan pada Gambar 2.8 Komponen LSD-SLAM.



**Gambar 2.8 Komponen LSD-SLAM**

Sumber: (Engel & Cremers, 2014)

Foto-foto baru dari kamera dilacak secara terus menerus menggunakan metode pelacakan langsung (*direct tracking*). Posisi foto-foto pada *keyframe* saat ini di perkirakan menggunakan pose *frame* sebelumnya. Ketika kamera bergerak di luar jarak dari *keyframe* saat ini, sebuah *keyframe* baru akan diinisialisasi dari foto terbaru yang telah dilacak dengan memproyeksi poin-poin dari *keyframe* terdekat untuk menghasilkan peta kedalaman (*depth map*). *Keyframe* tersebut digunakan untuk menggantikan *keyframe* lama (Engel, et al., 2014).

## BAB 3 METODOLOGI

### 3.1 Metode Penelitian

Pada bab ini akan menjelaskan tentang metodologi penelitian yang akan digunakan dalam melakukan penelitian maupun dalam penulisan skripsi. Adapun gambaran diagram alir metodologi yang digunakan dapat dilihat pada Gambar 3.1



**Gambar 3.1 Alur Metodologi Penelitian**

Gambar 3.1 Alur Metodologi Penelitian merupakan metodologi yang digunakan. Pada bab analisis kebutuhan akan dibahas kebutuhan yang akan digunakan. Pada bab perancangan dan implementasi akan dibahas mengenai perancangan sistem yang akan dibuat. Pada bab implementasi sistem akan diimplementasikan dan apabila tidak sesuai maka akan kembali pada tahap perancangan. Apabila sudah sesuai maka akan dilanjutkan ke pengujian dan analisis untuk memperoleh hasil yang diinginkan. Terakhir akan ditarik kesimpulan dan pengambilan saran.

### 3.2 Analisis Kebutuhan

Pada bagian ini membahas mengenai analisa kebutuhan dari sistem yang dibuat. Terdapat beberapa sub bab yang dibahas, diantaranya gambaran umum sistem yang menjelaskan tentang bagaimana sistem ini dibuat, analisis kebutuhan sistem yang membahas kebutuhan-kebutuhan yang digunakan untuk penelitian dan dibagi menjadi kebutuhan pengguna, perangkat keras, komunikasi, dan perangkat

lunak. Lalu kebutuhan fungsional dan kebutuhan non fungsional. Penjelasan lebih rinci akan dibahas pada bab 4 analisis kebutuhan.

### **3.3 Perancangan Sistem**

Perancangan sistem merupakan tahapan bagaimana membangun sebuah sistem dari penelitian yang dilakukan. Tahapan ini dilakukan setelah melakukan tahapan analisis kebutuhan. Dengan adanya tahapan ini maka sistem akan dapat digambarkan secara sistematis dan terstruktur. Perancangan sistem akan dibahas per bagian sistem yang akan dibahas lebih rinci pada bab 5 perancangan dan implementasi.

### **3.4 Implementasi Sistem**

Implementasi sistem dilaksanakan sesuai dengan perancangan yang telah ditentukan sebelumnya, mulai dari analisis kebutuhan hingga perancangan sistem. Penjelasan lebih rinci akan dibahas pada bab 5 perancangan dan implementasi.

### **3.5 Pengujian dan Analisis Sistem**

Tahapan pengujian dan analisis sistem untuk menguji apakah sistem yang dibuat sudah sesuai seperti yang diharapkan penulis. Pengujian dan analisis menggunakan beberapa parameter.

1. Pengujian fungsional dari sistem. Hal ini dilakukan untuk mengetahui apakah sistem sudah berjalan sesuai dengan keinginan penulis.
2. Pengujian aplikasi yang dibuat. Hal ini dilakukan untuk mengetahui bagaimana sistem merespon terhadap *input* yang dimasukkan.
3. Pembandingan hasil nyata dengan *input* yang dimasukkan. Hal ini dilakukan untuk mengukur kesalahan yang terjadi.

### **3.6 Penarikan Kesimpulan dan Saran**

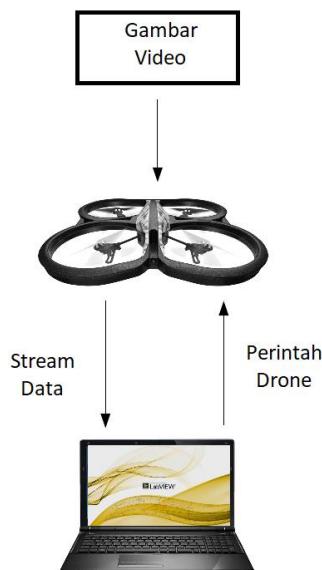
Penarikan kesimpulan merupakan tahap yang dilakukan setelah melakukan seluruh kegiatan pengujian sistem yang telah dirancang sebelumnya. Tujuan penarikan kesimpulan adalah agar penelitian ini dapat digunakan sebagai tolak ukur dan dapat dilanjutkan menjadi penelitian yang lebih baik serta tidak berhenti sampai kegiatan penulis selesai. Pengambilan saran bertujuan agar penelitian ini dapat dikembangkan menjadi penelitian yang lebih baik ke depannya.

## BAB 4 ANALISIS KEBUTUHAN

Tahapan analisis kebutuhan dibagi menjadi empat bagian yaitu gambaran umum sistem, kebutuhan sistem, kebutuhan fungsional serta kebutuhan non fungsional. Berikut merupakan uraian dari bagian-bagian tersebut.

### 4.1 Gambaran Umum Sistem

Penelitian ini akan mengimplementasikan *LSD-SLAM* (*Large-Scale Direct Monocular Simultaneous Localization and Mapping*) pada perangkat *quadcopter*. *Quadcopter* akan menangkap gambar melalui kamera terintegrasi yang kemudian akan dikirim ke laptop untuk diproses tiap *keyframe* menggunakan *direct image alignment* untuk diambil *cloudpoint*. *LSD-SLAM* menggunakan metode *direct*, yang berarti setiap informasi dari gambar akan diproses, termasuk tepi dan sudut dari sebuah obyek gambar. Gambar 4.1 Gambaran Umum Sistem merupakan gambaran umum sistem pada penelitian ini.



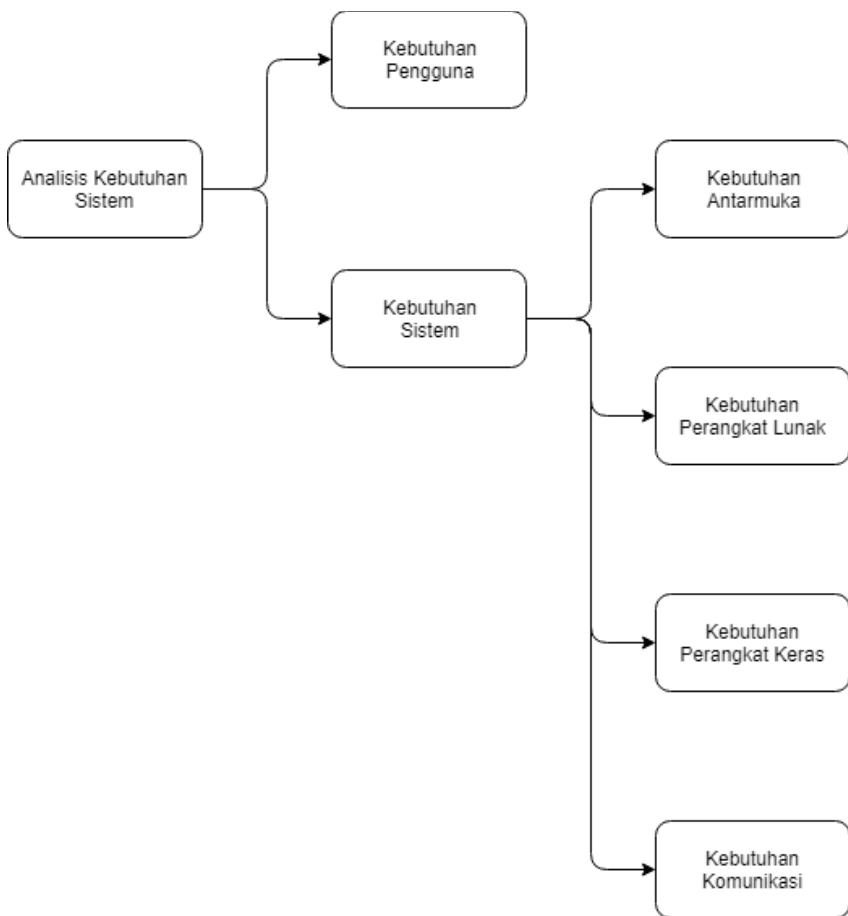
Gambar 4.1 Gambaran Umum Sistem

Pada Gambar 4.1 Gambaran Umum Sistem dijelaskan *quadcopter* mengirim data berupa video yang kemudian akan diproses *frame per frame*. Video diambil dari kamera terintegrasi *quadcopter*. Kamera yang digunakan dalam pengambilan video adalah kamera depan. Setelah *streaming* video mulai diterima oleh laptop *GCS* (*Ground Control System*), *GCS* akan mulai mengolah video tersebut kedalam *frame-frame* yang nantinya akan diolah menggunakan algoritma *LSD-SLAM*. *Frame-frame* yang diolah tersebut akan menjadi *cloudpoint* titik-titik yang nantinya akan ditampilkan sebagai model pemetaan. Sistem akan mengulang terus

proses ini hingga pengguna menghentikannya atau terjadi *loss tracking*.

## 4.2 Analisis Kebutuhan Sistem

Analisis kebutuhan sistem menggambarkan informasi yang dipakai pada perancangan sistem. Bagian ini akan menjelaskan analisis kebutuhan sistem sesuai dengan tujuan sistem agar sistem dapat berjalan sesuai keinginan.



**Gambar 4.2 Analisis Kebutuhan Sistem**

Gambar 4.2 Analisis Kebutuhan Sistem merupakan tahapan analisis kebutuhan sistem yang akan dibagi menjadi dua sub bagian, kebutuhan pengguna dan kebutuhan sistem. Bagian kebutuhan sistem akan dibagi lagi menjadi kebutuhan perangkat keras, kebutuhan perangkat lunak, kebutuhan komunikasi, dan kebutuhan antarmuka.

## 4.3 Kebutuhan Fungsional

Kebutuhan fungsional adalah kebutuhan yang menggambarkan prinsip kerja sistem yang akan dibuat dan selanjutnya menghasilkan keluaran sesuai dengan keinginan pengguna. Berikut ini adalah kebutuhan fungsional sistem.

#### **4.1.1 Mengakuisisi Citra dan Mendapatkan *Stream* Video**

Kemampuan mengakuisisi citra dan mendapatkan *stream* video merupakan kemampuan utama yang akan digunakan sebagai masukan pada LSD-SLAM.

**Tabel 4.1 Mengakuisisi Citra**

<b>Mengakuisisi Citra</b>			
<b>Tipe :</b>	Fungsional	<b>Prioritas :</b>	Tinggi
Sistem harus mampu mengakuisisi citra yang akan diteruskan sebagai video <i>stream</i> yang kemudian diolah menggunakan LSD-SLAM.			
<b>Keterangan :</b>	Menggunakan topik image_raw.		

Pada Tabel 4.1 Mengakuisisi Citra fungsi utama dalam penelitian adalah akuisisi citra yang kemudian akan digunakan sebagai video *stream* ke *Ground Control System*. Hasil video *stream* akan diproses menggunakan algoritma LSD-SLAM.

#### **4.1.2 Melakukan Navigasi di Dalam Ruangan**

Kemampuan navigasi di dalam ruangan merupakan kebutuhan utama dalam pengendalian navigasi *Ar.Drone*.

**Tabel 4.2 Navigasi Dalam Ruangan**

<b>Navigasi Dalam Ruangan</b>			
<b>Tipe :</b>	Fungsional	<b>Prioritas :</b>	Tinggi
Sistem harus mampu melakukan pergerakan serta terbang di dalam sebuah ruangan lengkap dengan berbagai obyek di dalamnya.			
<b>Keterangan :</b>	Menggunakan topik image_raw.		

Pada Tabel 4.2 Navigasi Dalam Ruangan dijelaskan bahwa *drone* harus mampu melakukan gerakan dan terbang di dalam sebuah ruangan. Kemampuan ini dibutuhkan agar *drone* dapat terbang sambil mengambil data dari kamera depan.

#### **4.1.3 Melakukan Deteksi Kesalahan Saat Mengambil Citra**

Kemampuan deteksi kesalahan merupakan kebutuhan utama dalam pengendalian pengambilan citra *Ar.Drone* yang nantinya akan digunakan saat pengolahan video *stream* menjadi *pointcloud*.

**Tabel 4.3 Melakukan Deteksi Kesalahan**

<b>Melakukan Deteksi Kesalahan</b>			
<b>Tipe :</b>	Fungsional	<b>Prioritas :</b>	Tinggi
Sistem harus mampu mendeteksi kesalahan yang terjadi karena pergerakan			

<i>drone</i> yang terlalu cepat atau karena terjadi <i>loss tracking</i> .
<b>Keterangan :</b> Menggunakan algoritma dari LSD-SLAM.

Pada Tabel 4.3 Melakukan Deteksi Kesalahan dijelaskan bahwa *drone* harus dapat mengeluarkan peringatan ketika terjadi kesalahan karena pergerakan *drone* maupun ketika terjadi *loss tracking* dari algoritma LSD-SLAM.

#### 4.1.4 Kebutuhan Komunikasi

Kebutuhan komunikasi adalah kebutuhan yang digunakan untuk menjembatani komunikasi antara *Ground Control System* dengan *Ar.Drone 2.0*.

##### 1. Wi-Fi

**Tabel 4.4 Wi-Fi**

<b>Wi-Fi</b>			
<b>Tipe :</b>	Sistem	<b>Prioritas :</b>	Tinggi
1. Mendukung komunikasi berbasis IEEE 802.11 b/g/n. 2. Wi-Fi digunakan untuk jalur komunikasi antara <i>GCS</i> dan <i>drone</i> .			
<b>Keterangan :</b>	Wi-Fi digunakan sebagai jalur komunikasi utama antara <i>GCS</i> dan <i>drone</i> .		

Pada Tabel 4.4 Wi-Fi dijelaskan tentang kebutuhan perangkat Wi-Fi adapter yang akan digunakan sebagai perangkat komunikasi antara Ar.Drone dan GCS. Wi-Fi merupakan jalur komunikasi utama untuk Ar.Drone 2.0. Data *stream* video serta kontrol *drone* dikirim melalui Wi-Fi.

#### 4.1.5 Kebutuhan Antarmuka

Kebutuhan antarmuka adalah kebutuhan untuk menjembatani pengguna sistem agar dapat menjalankan sistem secara *real time*.

##### 1. Inisialisasi

**Tabel 4.5 Inisialisasi**

<b>Inisialisasi</b>			
<b>Tipe :</b>	Sistem	<b>Prioritas :</b>	Tinggi
1. Inisialisasi koneksi antara <i>drone</i> dengan <i>Ground Control System</i> . 2. Mengatur program yang akan dijalankan.			
<b>Keterangan :</b>	Inisialisasi awal antara <i>drone</i> dan program.		

Inisialisasi Tabel 4.5 Inisialisasi digunakan untuk mengatur koneksi awal antara *drone* dengan *GCS*. Program ini memulai inisialisasi *driver* dan aplikasi pendukung untuk menjalankan *drone*.

## 2. Tampilan Kamera Depan

**Tabel 4.6 Tampilan Kamera Depan**

<b>Inisialisasi</b>			
<b>Tipe :</b>	Sistem	<b>Prioritas :</b>	Tinggi
1. Tampilan dari kamera depan. 2. Tampilan obyek-obyek yang ditangkap oleh kamera. 3. <i>Pointcloud</i> pada objek yang ditangkap kamera.			
<b>Keterangan :</b>	Tampilan dari kamera depan.		

Tampilan kamera depan Tabel 4.6 Tampilan Kamera Depan digunakan untuk menampilkan hasil video yang ditangkap oleh kamera depan. Serta menampilkan *pointcloud* pada obyek.

## 3. Tampilan LSD-SLAM

**Tabel 4.7 Tampilan LSD-SLAM**

<b>Tampilan LSD-SLAM</b>			
<b>Tipe :</b>	Sistem	<b>Prioritas :</b>	Tinggi
1. Tampilan <i>pointcloud</i> dari obyek yang ditangkap. 2. <i>Viewer</i> dari <i>LSD-SLAM</i> .			
<b>Keterangan :</b>	Tampilan <i>viewer</i> dari <i>LSD-SLAM</i> .		

Tampilan LSD-SLAM Tabel 4.7 Tampilan LSD-SLAM digunakan untuk menampilkan hasil proses dari video yang diterima. *Stream* video ini diproses dengan algoritma LSD-SLAM untuk menghasilkan *pointcloud*.

## 4. Tombol Kontrol Drone

**Tabel 4.8 Tombol Kontrol Drone**

<b>Tombol Kontrol Drone</b>			
<b>Tipe :</b>	Sistem	<b>Prioritas :</b>	Tinggi
1. Mengatur kontrol terbang <i>drone</i> . 2. Mengatur mode kamera <i>drone</i> .			
<b>Keterangan :</b>	Tombol kontrol <i>drone</i> .		

Kontrol *drone* Tabel 4.8 Tombol Kontrol Drone merupakan pengontrol terbang *drone*. Kontrol ini meliputi semua kontrol yang dibutuhkan untuk menerbangkan *drone* serta mengendalikannya ketika berada di udara.

#### 4.4 Kebutuhan Non-Fungsional

Kebutuhan Non-Fungsional merupakan kebutuhan yang menjelaskan tentang batasan-batasan kebutuhan pada perancangan sistem. Penjelasan ini akan menjadikan sistem dapat diketahui bagian-bagian yang dapat dikembangkan lebih lanjut. Berikut ini penjelasan tentang kebutuhan non-fungsional.

##### 4.3.1 Karakteristik Pengguna

Tabel 4.9 Karakteristik Pengguna

Karakteristik Pengguna			
Tipe :	Non-Fungsional	Prioritas :	Tinggi
Penelitian ini dapat digunakan dalam berbagai bidang misalnya bidang militer untuk pemetaan mata-mata, bidang arsitektur untuk pemetaan obyek-obyek yang ingin digambar, dan lain-lain.			
Keterangan :	-		

Berdasarkan Tabel 4.9 Karakteristik Pengguna sistem yang dirancang dapat digunakan oleh pengguna untuk berbagai bidang. Sistem juga dapat dikembangkan lebih lanjut untuk memetakan obyek spesifik seperti untuk bidang militer.

##### 4.3.2 Lingkungan Operasi

Tabel 4.10 Lingkungan Operasi

Lingkungan Operasi			
Tipe :	Non-Fungsional	Prioritas :	Tinggi
1. Jarak obyek yang dapat ditangkap maksimal 5 meter. 2. Diperlukan ruangan seluas 10 meter persegi agar <i>drone</i> dapat terbang dengan optimal.			
Keterangan :	-		

Berdasarkan Tabel 4.10 Lingkungan Operasi diperlukan ruangan seluas minimal 10 meter persegi agar *drone* dapat bekerja secara optimal. Jarak obyek 5 meter dikarenakan resolusi dari kamera Ar.Drone terbatas sehingga jarak lebih jauh dapat mengaburkan detail dari obyek yang ditangkap kamera.

### 4.3.3 Asumsi dan Ketergantungan

Tabel 4.11 Asumsi dan Ketergantungan

Asumsi dan Ketergantungan			
Tipe :	Non-Fungsional	Prioritas :	Tinggi
1. Sistem dapat berfungsi ketika <i>GCS</i> ( <i>Ground Control System</i> ) dan <i>drone</i> dihubungkan menggunakan Wi-Fi yang terkoneksi secara langsung. 2. Pengiriman data video <i>stream</i> hanya terjadi ketika tidak ada kesalahan seperti <i>loss tracking</i> atau terjadi kecelakaan. 3. Sistem hanya dapat berjalan ketika semua <i>driver</i> dan aplikasi pendukung seperti ardrone autonomy dan ROS terinstall.			
Keterangan :	-		

Tabel 4.11 Asumsi dan Ketergantungan menjelaskan tentang poin-poin yang harus diperhatikan dan dipenuhi agar sistem dapat berjalan optimal. Kesalahan aplikasi pendukung maupun *driver* yang belum terinstal dapat mengakibatkan sistem tidak berjalan.

### 4.3.4 Batasan Perancangan dan Implementasi

Tabel 4.12 Batasan Perancangan dan Implementasi

Batasan Perancangan dan Implementasi			
Tipe :	Non-Fungsional	Prioritas :	Tinggi
1. Jarak maksimal obyek yang dapat ditangkap adalah 5 meter. 2. Obyek atau benda di dalam ruangan harus mempunyai bentuk yang dapat dibedakan dengan latar belakang atau benda disampingnya. 3. Data navigasi yang ditampilkan antara lain kecepatan, ketinggian, serta sudut. 4. <i>Stream</i> video berbentuk <i>grayscale</i> atau hitam-putih. 5. Pergerakan <i>drone</i> tidak boleh terlalu cepat karena akan menimbulkan <i>loss tracking</i> .			
Keterangan :	-		

Tabel 4.12 Batasan Perancangan dan Implementasi *drone* tidak boleh terbang atau bergerak terlalu cepat karena dapat mengakibatkan *loss tracking* saat diproses menggunakan algoritma LSD-SLAM. Obyek yang akan dipetakan juga

harus dapat dibedakan bentuknya dengan latar belakang atau obyek disampingnya.

#### 4.3.5 Kebutuhan Pengguna

Kebutuhan pengguna merupakan kebutuhan yang diperlukan oleh pengguna untuk menjalankan atau mengoperasikan sistem.

**Tabel 4.13 Lingkungan Operasi**

Lingkungan Operasi			
Tipe :	Pengguna	Prioritas :	Tinggi
<i>Quadcopter</i> dapat terbang dalam suatu ruangan dan dapat mengambil video dari benda2 atau keadaan dalam ruang tersebut			
Keterangan :	Benda-benda dan keadaan ruang sudah diatur sebelumnya, keadaannya tidak berubah ubah.		

Pada keterangan Tabel 4.13 Lingkungan Operasi dalam pengoperasian *quadcopter* diharapkan dapat terbang dan mengambil video dari benda-benda dan keadaan dalam ruang serta mampu mengirimkan *stream* video tersebut ke *GCS* (*Ground Control System*).

**Tabel 4.14 Tampilan Antarmuka Sistem**

Tampilan Antarmuka Sistem			
Tipe :	Pengguna	Prioritas :	Tinggi
Antarmuka sistem akan menampilkan <i>pointcloud</i> dari <i>stream</i> video dan hasil dari proses <i>LSD-SLAM</i> serta memberitahukan pengguna ketika terjadi kesalahan pada sistem seperti kecelakaan atau <i>loss tracking</i> .			
<ol style="list-style-type: none"><li>1. <i>Streaming</i> video kamera depan.</li><li>2. Hasil <i>LSD-SLAM viewer</i>.</li><li>3. <i>Control PTAM drone</i>.</li><li>4. <i>Control drone</i>.</li></ol>			
Keterangan :	Antarmuka dibuat menggunakan <i>interface viewer</i> dan <i>control</i> dari <i>drone</i> .		

Pada keterangan Tabel 4.14 Tampilan Antarmuka Sistem dijelaskan bahwa pengguna dapat mengontrol *drone* dan dapat melihat hasil dari *LSD-SLAM* serta kesalahan yang mungkin terjadi.

#### 4.3.6 Kebutuhan Sistem

Kebutuhan sistem adalah kebutuhan yang dibutuhkan dalam pembangunan dan pengembangan sistem. Kebutuhan sistem dibagi menjadi kebutuhan

antarmuka, kebutuhan komunikasi, kebutuhan fungsional dan kebutuhan non fungsional.

#### 4.4 Kebutuhan Perangkat Keras

Kebutuhan perangkat keras adalah keseluruhan perangkat yang dibutuhkan untuk melakukan penelitian.

##### 4.4.1 Parrot Ar.Drone 2.0

Obyek penelitian adalah Parrot Ar.Drone 2.0. Perangkat ini memiliki kelebihan *API open source* sehingga dapat dikembangkan atau dimodifikasi sesuai kebutuhan. Perangkat ini juga memiliki berbagai sensor terintegrasi sehingga pengguna tidak perlu menambahkan sensor dasar lagi.



Gambar 4.3 Parrot Ar.Drone 2.0

Tabel 4.15 Parrot Ar.Drone 2.0

Parrot Ar.Drone 2.0			
Tipe :	Sistem	Prioritas :	Tinggi
<ol style="list-style-type: none"><li>1. Sensor-sensor bawaan <i>accelerometer</i>, <i>magnetometer</i>, <i>gyrometer</i> <i>pressure sensor</i>, <i>altitude sensor</i>.</li><li>2. Kamera depan dan bawah terintegrasi.</li><li>3. Prosesor 32 bit ARM Cortex A8.</li><li>4. RAM 1 GB.</li><li>5. Sistem Operasi Linux.</li><li>6. Rotor ECS terintegrasi.</li></ol>			
Keterangan :	Perangkat <i>quadcopter open source</i> .		

Dari Tabel 4.15 Parrot Ar.Drone 2.0 dapat disimpulkan Ar.Drone memiliki

kelebihan dalam modifikasi *API* sehingga memudahkan dalam pengembangan.

**Tabel 4.16 Spesifikasi Kamera Ar.Drone 2.0**

<b>Spesifikasi Kamera Ar.Drone 2.0</b>			
<b>Tipe :</b>	Sistem	<b>Prioritas :</b>	Tinggi
1. Kamera depan beresolusi 720p, <i>HD (High Definition)</i> 30 <i>FPS (Frame Per Second)</i> . 2. Kamera bawah beresolusi VGA (480p), 60 <i>FPS (Frame Per Second)</i> . 3. <i>Encoding</i> H264. 4. <i>Video storage on the fly</i> .			
<b>Keterangan :</b>	Spesifikasi Kamera Ar.Drone 2.0		

Dari Tabel 4.16 Spesifikasi Kamera Ar.Drone 2.0 dapat disimpulkan kamera *Ar.Drone* dapat digunakan untuk mengambil video dengan kualitas cukup baik untuk *streaming* dengan detail yang dapat dibedakan antara benda-benda yang ditangkap.

#### **4.4.2 GCS (Ground Control System)**

Dalam penelitian ini *GCS (Ground Control System)* berguna untuk memproses video dari *Ar.Drone* dengan algoritma *LSD-SLAM* sehingga akan didapatkan *pointcloud* yang akan ditampilkan pada pengguna. *GCS* juga berguna untuk mengirimkan perintah kontrol ke *Ar.Drone*.

**Tabel 4.17 Spesifikasi Komputer GCS**

<b>Spesifikasi Komputer GCS</b>			
<b>Tipe :</b>	Sistem	<b>Prioritas :</b>	Tinggi
1. RAM 4 GB atau lebih 2. Prosesor dual core atau lebih 3. Sistem operasi ubuntu 14.04			
<b>Keterangan :</b>	Spesifikasi komputer digunakan sebagai <i>ground control sistem</i> .		

Pada Tabel 4.17 Spesifikasi Komputer GCS dijelaskan tentang spesifikasi komputer yang harus dipenuhi untuk dapat menjalankan program dengan lancar. Spesifikasi dibawahnya akan menyebabkan program tidak berjalan dengan baik atau bahkan tidak dapat dijalankan sama sekali.

**Tabel 4.18 Spesifikasi Wi-Fi Adapter**

<b>Spesifikasi Wi-Fi Adapter</b>			
<b>Tipe :</b>	Sistem	<b>Prioritas :</b>	Tinggi
1. Mendukung komunikasi dengan protokol IEEE 802.11 <i>Wireless LAN</i> . 2. Mampu melakukan komunikasi dengan <i>drone</i> . 3. Dapat langsung <i>plug and play</i> pada sistem operasi Ubuntu 14.04 tanpa <i>driver</i> tambahan.			
<b>Keterangan :</b> Wi-Fi sebagai jalur komunikasi.			

Pada Tabel 4.18 Spesifikasi Wi-Fi Adapter dijelaskan adapter Wi-Fi harus dapat mendukung komunikasi dengan *Ar.Drone* serta harus *plug and play* dengan sistem operasi Ubuntu 14.04. Hal ini dimaksudkan agar pada saat aplikasi dijalankan tidak terjadi kesalahan yang dapat mengakibatkan kecelakaan. Sifat *plug and play* juga dimaksudkan agar Wi-Fi adapter bisa langsung digunakan tanpa harus menggunakan aplikasi tambahan.

## **4.5 Kebutuhan Perangkat Lunak**

Kebutuhan perangkat lunak adalah keseluruhan aplikasi serta sistem operasi yang dibutuhkan agar sistem dapat berjalan. Kebutuhan ini termasuk aplikasi penunjang serta *driver-driver* untuk menghubungkan antara *Ar.Drone* dengan *GCS*.

### **4.5.1 Sistem Operasi Ubuntu 14.04**

**Tabel 4.19 Sistem Operasi**

<b>Sistem Operasi</b>			
<b>Tipe :</b>	Sistem	<b>Prioritas :</b>	Tinggi
Ubuntu 14.04 adalah sistem operasi yang digunakan sebagai basis dari aplikasi-aplikasi yang nanti akan dijalankan.			
<b>Keterangan :</b>	Sistem operasi wajib ubuntu 14.04 update 14.04.6.		

Pada Tabel 4.19 Sistem Operasi versi Ubuntu yang digunakan adalah 14.04 dengan update 14.04.6 yang dikeluarkan pertama kali pada April 2014. Sistem operasi ini digunakan untuk mendapatkan kompatibilitas maksimal dengan *driver-driver* serta aplikasi pendukung untuk menjalankan *Ar.Drone*.

#### 4.5.2 ROS Indigo

Tabel 4.20 Sistem Operasi

ROS Indigo			
Tipe :	Sistem	Prioritas :	Tinggi
1. Paket ROS Indigo untuk Ubuntu 14.04. 2. Gazebo versi 2			
Keterangan :	ROS Indigo Ubuntu 14.04.		

Pada tabel ROS Indigo diketahui bahwa paket aplikasi yang digunakan adalah untuk Ubuntu 14.04. Paket ROS Indigo ini telah dirancang untuk kompatibel dengan ubunut 14.04 dan *driver* serta paket tambahan unutuk menjalankan *Ar.Drone*. Pada paket ROS Indigo telah tersedia Gazebo versi 2 yang dapat digunakan unutk menjalankan simulasi *drone* pada komputer. Simulasi ini berguna untuk mengetes aplikasi sebelum diimplementasikan langsung pada *drone*. Hal ini bertujuan untuk menghindari terjadinya kecelakaan pada *drone*.

#### 4.5.3 Ardrone Autonomy

Tabel 4.21 Ardrone Autonomy

Ardrone Autonomy			
Tipe :	Sistem	Prioritas :	Tinggi
Ardrone autonomy merupakan <i>driver</i> <i>Ar.Drone</i> . <i>Driver</i> ini berfungsi sebagai penghubung antara <i>API</i> pada <i>Ar.Drone</i> dan perintah-perintah yang dimasukkan oleh pengguna. <i>Driver</i> ini mengubah perintah pengguna menjadi instruksi yang dimengerti oleh <i>drone</i> .			
Keterangan :	<i>Driver Ar.Drone</i> .		

Pada Tabel 4.21 Ardrone Autonomy dijelaskan bahwa ardrone autonomy merupakan *driver* dari ardrone untuk Ubuntu 14.04 dengan ROS Indigo.

#### 4.5.4 TUM Ardrone

Tabel 4.22 TUM Ardrone

TUM Ardrone			
Tipe :	Sistem	Prioritas :	Tinggi
TUM Ardrone merupakan antarmuka untuk mengontrol gerakan dari <i>drone</i> . Paket ini digunakan untuk antarmuka dalam mengendalikan ardrone.			

<b>Keterangan :</b>	<i>Driver antarmuka Ar.Drone.</i>
---------------------	-----------------------------------

Pada Tabel 4.22 TUM Ardrone dijelaskan bahwa TUM Ardrone merupakan *driver* untuk mengontrol antarmuka dari ardrone untuk Ubuntu 14.04 dengan ROS Indigo.

## BAB 5 PERANCANGAN DAN IMPLEMENTASI

Dalam bab perancangan dan implementasi akan dijelaskan mengenai penerapan dari analisis kebutuhan sehingga bisa menjadi satu kesatuan sistem yang utuh. Dilakukan perancangan dan implementasi dari sistem.



Berdasarkan gambar ... perancangan dan implementasi sistem terdapat beberapa sub bab, yaitu perancangan dan implementasi sistem komunikasi, data latih, program pada GCS (Ground Control System), serta pergerakan quadcopter.

### 5.1 Perancangan Sistem

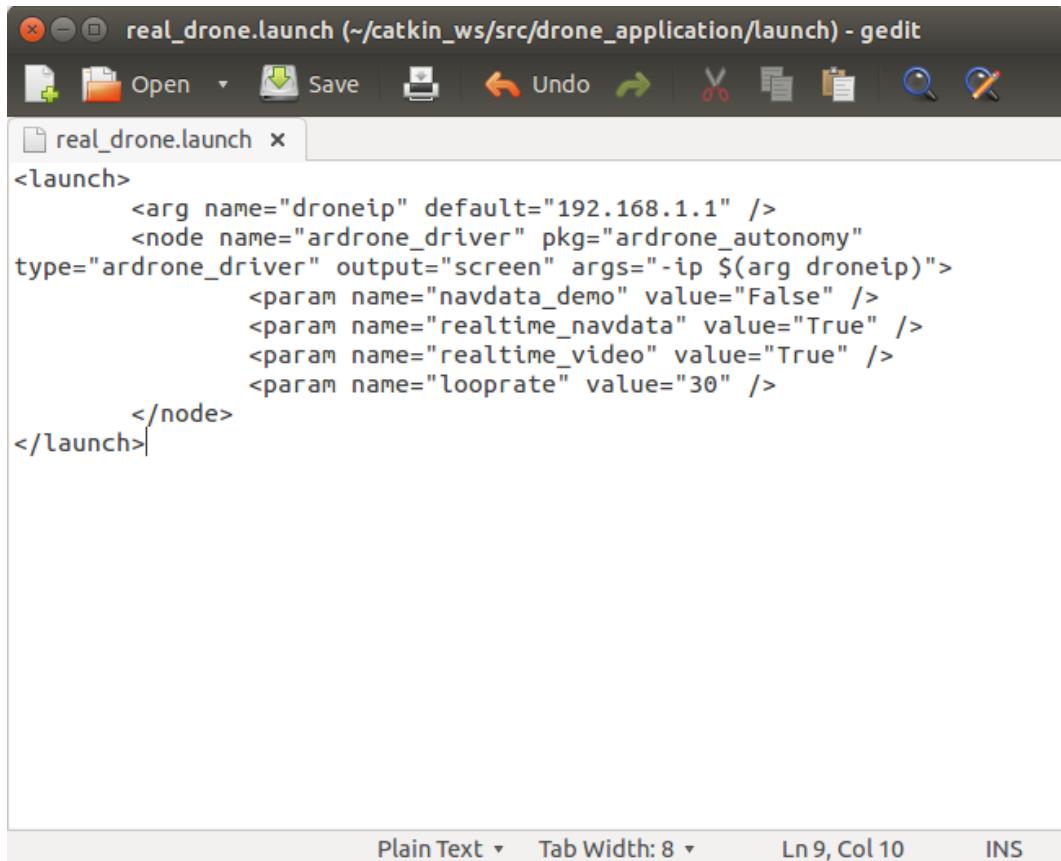
Perancangan sistem merupakan bab yang menjelaskan tentang bagaimana komponen-komponen dalam sistem didesain agar dapat menjalankan fungsinya. Pada bab ini akan dijelaskan alur yang dilewati agar setiap komponen dapat berjalan.

### 5.1.1 Sistem Komunikasi

Perancangan sistem komunikasi membahas tentang sistem komunikasi yang digunakan dalam pertukaran data pada keseluruhan sistem yang dibuat. Pada bab ini akan dijelaskan bagaimana alur pengaturan yang dijalankan agar quadcopter dan komputer GCS (Ground Control System) dapat saling terhubung dan melakukan pertukaran data. Media koneksi yang digunakan adalah Wi-Fi. Wi-Fi memungkinkan perangkat komputer dan quadcopter dapat terhubung secara nirkabel.

#### Flowchart

##### Gambar konfigurasi ip



The screenshot shows a terminal window titled "real\_drone.launch (~/catkin\_ws/src/drone\_application/launch) - gedit". The file content is an XML launch configuration for a drone. It defines a launch argument "droneip" with a default value of "192.168.1.1". It then specifies a node named "ardrone\_driver" from the package "ardrone\_autonomy". The node has type "ardrone\_driver", output "screen", and args "-ip \${arg droneip}". It includes four parameters: "navdata\_demo" (value "False"), "realtime\_navdata" (value "True"), "realtime\_video" (value "True"), and "looprate" (value "30"). The XML code is as follows:

```
<?xml version="1.0"?>
<launch>
    <arg name="droneip" default="192.168.1.1" />
    <node name="ardrone_driver" pkg="ardrone_autonomy"
type="ardrone_driver" output="screen" args="-ip ${arg droneip}">
        <param name="navdata_demo" value="False" />
        <param name="realtime_navdata" value="True" />
        <param name="realtime_video" value="True" />
        <param name="looprate" value="30" />
    </node>
</launch>
```

At the bottom of the editor, there are status indicators: "Plain Text", "Tab Width: 8", "Ln 9, Col 10", and "INS".

### 5.1.2 Sistem Navigasi

Perancangan sistem navigasi membahas tentang bagaimana pergerakan sebuah quadcopter dapat dilakukan. Pergerakan sebuah quadcopter dapat dibedakan menjadi dua yaitu pergerakan linear dan pergerakan angular. Pergerakan linear yaitu pergerakan seperti maju-mundur, ke kanan-kiri, serta naik-turun. Pergerakan angular yaitu pergerakan searah jarum jam maupun berlawanan jarum jam. Kemampuan pergerakan inilah yang menjadikan quadcopter dapat

bermanuver dalam berbagai kondisi penggunaan. Pada sub-bab ini akan dibahas alur pergerakan yang akan dilakukan quadcopter pada sistem ini.

**Flowchart maju**

**Flowchart mundur**

**Flowchart ke kanan**

**Flowchart ke kiri**

**Flowchart naik**

**Flowchart turun**

**Flowchart searah jarum jam**

**Flowchart berlawanan jarum jam**

### **5.1.3 Sistem Kamera**

Perancangan sistem kamera membahas alur konfigurasi kamera quadcopter agar dapat digunakan sebagai referensi dalam sistem LSD-SLAM. Kamera yang digunakan merupakan kamera depan yang terintegrasi dalam quadcopter. Sistem kamera dirancang agar konfigurasi hanya perlu dilakukan sekali saja. Sistem kemudian menyimpan data konfigurasi sebagai data referensi yang akan selalu digunakan pada pengujian selanjutnya.

**Flowchart sistem kamera**

### **5.1.4 Sistem LSD-SLAM**

Perancangan LSD-SLAM membahas bagaimana alur konfigurasi LSD-SLAM agar dapat digunakan pada komputer. LSD-SLAM akan bekerja setelah komputer selesai menerima gambar dari kamera quadcopter. Gambar ini akan diolah menggunakan LSD-SLAM hingga diperoleh cloud point atau titik-titik yang menggambarkan kondisi sebuah ruang.

**Flowchart LSD-SLAM**

### **5.1.5 Sistem Penampil Hasil**

Perancangan sistem penampil hasil membahas bagaimana alur konfigurasi komputer agar dapat menampilkan hasil dari pengolahan gambar oleh LSD-SLAM. Gambar yang ditampilkan merupakan cloud point atau titik-titik yang

merepresentasikan hasil pengolahan kondisi sebuah ruang. Hasil yang ditampilkan dapat dilihat dan dimanipulasi seperti digeser, diperbesar, maupun diputar.

### Flowchart penampil

## 5.2 Implementasi Sistem

Implementasi sistem merupakan sub-bab yang membahas tentang bagaimana komponen dalam sistem diimplementasikan agar sesuai dengan sub-bab perancangan sistem. Pada sub-bab ini akan dijelaskan bagaimana perancangan di atas akan diimplementasikan ke dalam setiap komponen berupa potongan-potongan kode yang nantinya dapat menjalankan fungsi sistem secara sesuai dengan yang dirancang.

### 5.2.1 Sistem Komunikasi

Implementasi sistem komunikasi membahas bagaimana pengaturan alamat yang digunakan untuk komunikasi antara komputer dan quadcopter. Alamat IP yang digunakan akan diatur dalam konfigurasi awal sehingga komputer dapat mengenali quadcopter. Quadcopter berperan sebagai host / server yang akan digunakan untuk menerima perintah yang dikirimkan dari komputer.

#### Gambar konfigurasi ip

```
○ ⊖ ⊕ /home/oktat/catkin_ws/src/drone_application/launch/real_drone.launch http://localhost:1446
* /ardrone_driver/navdata_demo: False
* /ardrone_driver/realtme_navdata: True
* /ardrone_driver/realtme_video: True
* /rosdistro: indigo
* /rosversion: 1.11.21

NODES
/
    ardrone_driver (ardrone_autonomy/ardrone_driver)

auto-starting new master
process[master]: started with pid [17073]
ROS_MASTER_URI=http://localhost:11311

setting /run_id to 6f8ee9fe-7ac1-11ea-b1c3-2c56dcbe6e8b
process[rosout-1]: started with pid [17086]
started core service [/rosout]
process[ardrone_driver-2]: started with pid [17089]
Using custom ip address 192.168.1.1
Wait authentication
Wait authentication
Wait authentication
=====+> 192.168.1.1
Getting AR.Drone version ...
```

Implementasi ini menggunakan ROS (Robot Operating System). ROS menyediakan library-library dan berbagai peralatan untuk membantu menghubungkan sistem operasi pada robot dan perangkat GCS. ROS juga menyediakan driver, sistem pertukaran data, visualisasi dan banyak fungsi lain. ROS pada sistem yang dibuat

menjembatani antara driver yang terinstall pada GCS dengan sistem operasi ardrone.

Driver yang digunakan pada GCS merupakan ardrone autonomy. Ardrone autonomy membutuhkan instalasi ROS agar dapat dicompile untuk sistem. Proses compile ini akan menghasilkan paket-paket yang nantinya akan digunakan dalam penterjemahan perintah standar dari GCS agar dapat dimengerti oleh sistem operasi dari quadcopter. Setelah semua paket dicompile dengan sempurna akan didapatkan executable node atau aplikasi yang dapat dijalankan untuk menghubungkan antara GCS dan quadcopter.

Pada versi SDK (Software Development Kit) ardrone 2.0.1 yang menjadi standar terbaru dalam driver quadcopter parrot ardrone 2.0, ardrone autonomy mempunyai beberapa fungsi dan parameter yang dapat digunakan untuk mengontrol drone. Pembacaan data dan pengiriman dapat dilakukan secara realtime maupun fixed rate (data disimpan terlebih dahulu kemudian dikirim secara bertahap). Sebagai standar dalam versi terbaru ini adalah data akan disimpan terlebih dahulu kemudian akan dikirimkan secara bertahap dengan parameter sebagai berikut realtime\_navdata=False dan looprate=50. Parameter untuk realtime\_navdata adalah boolean sehingga bernilai True atau False. Looprate merupakan parameter yang digunakan untuk menyetel banyaknya pembaruan non-realtime yang terjadi tiap detiknya, dengan standar 50hz atau 50 kali dalam satu detik.

Pada transmisi data antara drone dengan GCS terdapat pengiriman data yang dipublikasi oleh topik /ardrone/navdata. Tipe pesan yang dikirim adalah ardrone\_autonomy::Navdata dan berisi berbagai nilai untuk masing-masing fungsi sensor pada quadcopter. Data-data inilah yang nantinya digunakan sebagai panduan dalam sistem kontrol drone.

ROS master menyediakan penamaan dan layanan registrasi kepada node. Node merupakan sebuah proses yang melakukan komputasi. Node-node digabungkan secara bersamaan dan berkomunikasi dengan menggunakan topik streaming, layanan RPC dan Parameter Server. ROS Master melacak publisher dan subscriber topik dan layanan. Fungsi utama ROS Master adalah memungkinkan sebuah node untuk dapat mengenali node lainnya. Setelah node-node ini dapat mengenali node lainnya, maka node akan dapat saling berkomunikasi. ROS Master menggunakan port 11311 sebagai standar. Node-node ROS akan mengeset soket TCP secara acak. Port ini digunakan untuk pemanggilan layanan xmlrpc dari ROS Master dan node-node lain, serta port-port lain untuk beberapa topik dan layanan lain. Alokasi port-port ini terjadi secara acak dan dilakukan oleh sistem operasi sehingga perlu dilakukan pengecekan pada node-node yang sedang berjalan. IP Address standar untuk komunikasi antara GCS dan drone adalah 192.168.1.1.

Pada GCS terdapat beberapa macam aplikasi yang masing-masing ditujukan untuk fungsi yang berbeda. GCS menjembatani antara pengguna serta quadcopter dan proses-proses yang berjalan diantara keduanya. GCS pada sistem ini dipasang berbagai aplikasi dan driver pendukung untuk menghubungkan

antara perintah yang dimasukkan oleh pengguna dan sistem operasi quadcopter. Fungsi dari driver ini adalah agar quadcopter dapat mengenali perintah dari pengguna tanpa pengguna harus menggunakan instruksi khusus yang dapat langsung dimengerti oleh sistem operasi quadcopter.

\$ roscore	Perintah untuk menjalankan ROS Master
------------	---------------------------------------

\$ rosrun lsd_slam_core live_slam image:=/ardrone/front/image_raw _hz:=20 _calib:=/lokasi/nama_konfigurasi.cfg	Perintah untuk menjalankan pemrosesan SLAM
--	--

\$ rosrun lsd_slam_viewer viewer	Perintah untuk menjalankan jendela viewer untuk melihat hasil proses SLAM
----------------------------------	---

\$ rosrun record /lsd_slam/graph /lsd_slam/keyframes /lsd_slam/liveframes -o /lokasi/nama_file.bag	Perintah untuk melakukan perekaman hasil proses SLAM ke dalam sebuah file
--	---

### 5.2.2 Sistem Navigasi

Implementasi sistem navigasi membahas bagaimana komputer agar dapat mengirimkan perintah kode ke quadcopter agar dapat melakukan pergerakan.

**Kode maju**

**Kode mundur**

**Kode ke kanan**

**Kode ke kiri**

**Kode naik**

**Kode turun**

**Kode searah jarum jam**

### Kode berlawanan jarum jam

Pada perancangan pergerakan quadcopter akan dilakukan pemberian nilai pada parameter-parameter di ardronne autonomy. Pemberian nilai ini dilakukan untuk melakukan perintah pergerakan kepada quadcopter.

Pesan std\_msgs/Empty digunakan untuk mulai terbang atau takeoff bila dipublish ke /ardrone/takeoff. Drone akan mendarat apabila pesan std\_msgs/Empty dipublish ke /ardrone/land. Serta drone akan melakukan pendaratan darurat apabila pesan std\_msgs/Empty dipublish ke /ardrone/reset.

<code>rostopic pub -1 ardrone/takeoff std_msgs/Empty</code>	Memulai terbang
<code>rostopic pub -1 ardrone/land std_msgs/Empty</code>	Mendarat
<code>rostopic pub -1 ardrone/reset std_msgs/Empty</code>	Pendaratan darurat

Drone dapat terbang dan bergerak setelah takeoff apabila pesan dengan tipe geometry\_msgs::Twist di-publish ke topik cmd\_vel. Beberapa daftar perintah terbang drone antara lain sebagai berikut.

- linear.x	bergerak mundur
+ linear.x	bergerak maju
- linear.y	bergerak ke kanan
+ linear.y	bergerak ke kiri
- linear.z	bergerak ke bawah
+ linear.z	bergerak ke atas
- angular.z	berputar ke kanan
+ angular.z	berputar ke kiri

Nilai dari setiap parameter adalah antara -1.0 dan 1.0. Contohnya adalah sebagai berikut.

<code>rostopic pub -r 10 /cmd_vel geometry_msgs/Twist '{linear: {x: 1.0, y: 0.0, z: 0.0}, angular: {x: 0.0,y: 0.0,z: 0.0}'}</code>	Bergerak maju ke depan
--	------------------------

rostopic pub -r 10 /cmd_vel geometry_msgs/Twist '{linear: {x: -1.0, y: 0.0, z: 0.0}, angular: {x: 0.0,y: 0.0,z: 0.0}}'	Bergerak mundur ke belakang
rostopic pub -r 10 /cmd_vel geometry_msgs/Twist '{linear: {x: 0.0, y: 1.0, z: 0.0}, angular: {x: 0.0,y: 0.0,z: 0.0}}'	Bergerak ke arah kiri
rostopic pub -r 10 /cmd_vel geometry_msgs/Twist '{linear: {x: 0.0, y: -1.0, z: 0.0}, angular: {x: 0.0,y: 0.0,z: 0.0}}'	Bergerak ke arah kanan
rostopic pub -r 10 /cmd_vel geometry_msgs/Twist '{linear: {x: 0.0, y: 0.0, z: 1.0}, angular: {x: 0.0,y: 0.0,z: 0.0}}'	Bergerak naik ke atas
rostopic pub -r 10 /cmd_vel geometry_msgs/Twist '{linear: {x: 0.0, y: 0.0, z: -1.0}, angular: {x: 0.0,y: 0.0,z: 0.0}}'	Bergerak turun ke bawah
rostopic pub -r 10 /cmd_vel geometry_msgs/Twist '{linear: {x: 0.0, y: 0.0, z: 0.0}, angular: {x: 0.0,y: 0.0,z: 1.0}}'	Bergerak berputar ke kanan searah jarum jam
rostopic pub -r 10 /cmd_vel geometry_msgs/Twist '{linear: {x: 0.0, y: 0.0, z: 0.0}, angular: {x: 0.0,y: 0.0,z: -1.0}}'	Bergerak berputar ke kiri berlawanan arah jarum jam
rostopic pub -r 10 /cmd_vel geometry_msgs/Twist '{linear: {x: 0.0, y: 0.0, z: 0.0}, angular: {x: 0.0,y: 0.0,z: 0.0}}'	Berhenti di udara atau hover

rosservice call /ardrone/togglecam	Berganti pilihan seleksi kamera
rosrun image_view image_view image:=/ardrone/image_raw	Menampilkan kamera terpilih
rosrun image_view image_view image:=/ardrone/front/image_raw	Menampilkan kamera depan
rosrun image_view image_view image:=/ardrone/bottom/image_raw	Menampilkan kamera bawah
rostopic echo /sonar_height	Menampilkan data sensor ketinggian
rostopic echo /ardrone/navdata	Menampilkan data navigasi

### 5.2.3 Sistem Kamera

Implementasi sistem kamera membahas konfigurasi kamera quadcopter agar dapat digunakan sebagai referensi dalam sistem LSD-SLAM. Kamera yang digunakan merupakan kamera depan yang terintegrasi dalam quadcopter. Kamera ini akan digunakan untuk mengambil gambar kondisi dalam ruangan.

#### Kode konfigurasi kamera

Pada parrot ardrone 2.0 terdapat dua buah kamera, yaitu kamera depan dan kamera bawah. Kamera depan mampu menyediakan stream video dengan resolusi 640x320 pixel dalam 20 FPS (Frame Per Second) dan format codec H264. Kamera ini perlu dikalibrasi dengan menggunakan pola catur (checkerboard) agar kondisi kamera serta pencahayaan dapat dijadikan nilai standar. Kalibrasi ini akan disimpan pada parameter ROS atau melalui ardrone\_front.yaml dan ardrone\_bottom.yaml. informasi kalibrasi juga akan tersedia dan di-publish oleh topik /ardrone/camera\_info. Topik /ardrone/ akan selalu berisi stream video dan informasi dari kamera terpilih (kamera depan atau kamera bawah). Pada ardrone 2.0 fitur PIP (Picture In Picture) ditiadakan, sehingga pengguna hanya dapat memilih satu kamera saja yang aktif. Peniadaan fitur ini menjadikan kamera depan dan bawah tidak dapat aktif secara bersamaan.

Pada penggunaan ardrone autonomy, pada saat menjalankan kamera akan dibuat tiga buah topik. Topik-topik ini adalah /ardrone/image\_raw, /ardrone/front/image\_raw, dan /ardrone/bottom/image\_raw. Topik-topik ini mem-publish pesan dengan tipe image transport. Image transport digunakan untuk subscribe dan publish gambar. Image transport menyediakan dukungan untuk pengiriman gambar-gambar dalam format kompresi lebar pita rendah (low bandwidth).

**Tabel 5.1 Kode ost.yaml**

Baris	Kode ost.yaml
	image_width: 640 image_height: 360 camera_name: narrow_stereo camera_matrix: rows: 3 cols: 3 data: [647.777517, 0.000000, 360.096745, 0.000000, 643.056521, 94.558975, 0.000000, 0.000000, 1.000000] distortion_model: plumb_bob distortion_coefficients: rows: 1 cols: 5

	<pre> data: [-0.480982, 0.122341, 0.028361, -0.008703, 0.000000] rectification_matrix: rows: 3 cols: 3 data: [1.000000, 0.000000, 0.000000, 0.000000, 1.000000, 0.000000, 0.000000, 0.000000, 1.000000] projection_matrix: rows: 3 cols: 4 data: [549.940918, 0.000000, 369.061854, 0.000000, 0.000000, 613.362915, 92.528829, 0.000000, 0.000000, 0.000000, 1.000000, 0.000000]</pre>
--	--

Pada tabel ... dijelaskan bahwa hasil sensor kamera yang didapat setelah dilakukan kalibrasi menggunakan checkerboard.

**Tabel 5.2 Kode model**

Baris	Kode model kamera OpenCV
	<pre> fx fy cx cy k1 k2 p1 p2 inputWidth inputHeight "crop" / "full" / "none" / "e1 e2 e3 e4 0" outputWidth outputHeight</pre>

Pada tabel ... dijelaskan bahwa model kamera yang dipakai menggunakan model kamera OpenCV. Hasil kalibrasi dari sensor kamera kemudian dipecah dengan format tersebut agar dapat digunakan oleh library.

**Tabel 5.3 Kode hasil kalibrasi**

Baris	Kode kalibrasi kamera ardron
	<pre> 613.039384 609.869284 339.686955 165.397894 -0.648509 0.448876 0.009435 -0.011302 640 360 crop 576 320</pre>

Pada tabel ... dijelaskan hasil kalibrasi yang telah dipecah ke dalam model OpenCV. Masing-masing nilai tersebut disusun sehingga library dapat menggunakan kalibrasi yang telah dilakukan tanpa harus menggunakan kalibrasi standar.

#### **5.2.4 Sistem LSD-SLAM**

Implementasi sistem LSD-SLAM membahas bagaimana LSD-SLAM akan digunakan sebagai pengolah gambar yang ditangkap oleh quadcopter. Sistem LSD-SLAM merupakan library atau perpustakaan kode yang dapat dipanggil ketika komputer telah selesai menerima gambar. Library ini memuat algoritme yang akan mengolah gambar menjadi point cloud dari objek yang ditangkap oleh kamera.

##### **Kode konfigurasi LSD-SLAM**

**Tabel 5.4 Kode eksekusi**

Baris	Kode eksekusi lsd_slam
	rosrun lsd_slam_core live_slam /image:=<streamtopic> _calib:=<calibration_file>

Pada tabel ... diatas dijelaskan kode yang harus dijalankan pada terminal. Kode tersebut akan memanggil library dari LSD SLAM. Pada bagian image streamtopic diganti menggunakan topic stream kamera ardrone (/ardrone/front/image\_raw). Pada bagian calib dimasukkan file hasil kalibrasi yang telah dijadikan model OpenCV.

#### **5.2.5 Sistem Penampil Hasil**

Implementasi sistem penampil hasil membahas bagaimana hasil pengolahan gambar menggunakan LSD-SLAM akan ditampilkan pada layar komputer. Penampil hasil memungkinkan visualisasi dari cloud point atau titik-titik representasi dari objek dalam sebuah ruangan.

##### **Kode penampil hasil**

**Tabel 5.5 Kode eksekusi viewer**

Baris	Kode eksekusi viewer
	rosrun lsd_slam_viewer viewer

Pada tabel ... diatas dijelaskan bagaimana viewer / penampil hasil proses dari LSD SLAM ditampilkan dalam window / jendela baru. Saat kode tersebut dieksekusi jendela akan menunggu hasil proses sebuah gambar selesai kemudian akan ditampilkan titik pointcloud dari gambar tersebut. Jendela akan mengulangi proses tersebut terus menerus hingga semua gambar selesai diproses dan ditampilkan.

## BAB 6 PENGUJIAN DAN ANALISIS SISTEM

### 6.1 Simulasi di Gazebo



```
oktat@oktat-desktop: ~
oktat@oktat-desktop:~$ rosrun drone_application test_sim_lab.launch
```

Gambar 6.1 kode meluncurkan simulasi

```
/home/oktat/catkin_ws/src/drone_application/launch/test_sim_lab.launch http://localhost:41863
oktat@oktat-desktop:~$ rosrun drone_application test_sim_lab.launch
... logging to /home/oktat/.ros/log/df3ff604-b5ee-11ea-a109-0025222824ff/rosrun-4249.log
Checking log directory for disk usage. This may take awhile.
Press Ctrl-C to interrupt
Done checking log file disk usage. Usage is <1GB.

started rosrun server http://oktat-desktop:41863/

SUMMARY
=====

PARAMETERS
* /ground_truth_to_tf/frame_id: nav
* /ground_truth_to_tf/odometry_topic: ground_truth/state
* /robot_description: <?xml version="1....
* /robot_state_publisher/publish_frequency: 50.0
* /robot_state_publisher/tf_prefix:
* /rostdistro: indigo
* /rosversion: 1.11.21
* /use_sim_time: True

NODES
/
```

Gambar 6.2 alamat dan port rosrun

```
/home/oktat/catkin_ws/src/drone_application/launch/test_sim_lab.launch http://localhost:11311
auto-starting new master
process[master]: started with pid [4265]
ROS_MASTER_URI=http://localhost:11311

setting /run_id to df3ff604-b5ee-11ea-a109-0025222824ff
process[rosout-1]: started with pid [4278]
started core service [/rosout]
process[gazebo-2]: started with pid [4281]
process[gazebo_gui-3]: started with pid [4285]
process[spawn_robot-4]: started with pid [4289]
process[robot_state_publisher-5]: started with pid [4293]
process[ground_truth_to_tf-6]: started with pid [4297]
[ WARN] [1592984818.603311514]: The 'state_publisher' executable is deprecated.
Please use 'robot_state_publisher' instead
[ WARN] [1592984818.606211087]: The root link base_link has an inertia specified
in the URDF, but KDL does not support a root link with an inertia. As a workar
ound, you can add an extra dummy link to your URDF.
Gazebo multi-robot simulator, version 2.2.3
Copyright (C) 2012-2014 Open Source Robotics Foundation.
Released under the Apache 2 License.
http://gazebosim.org

Gazebo multi-robot simulator, version 2.2.3
Copyright (C) 2012-2014 Open Source Robotics Foundation.
```

Gambar 6.3 alamat ros master

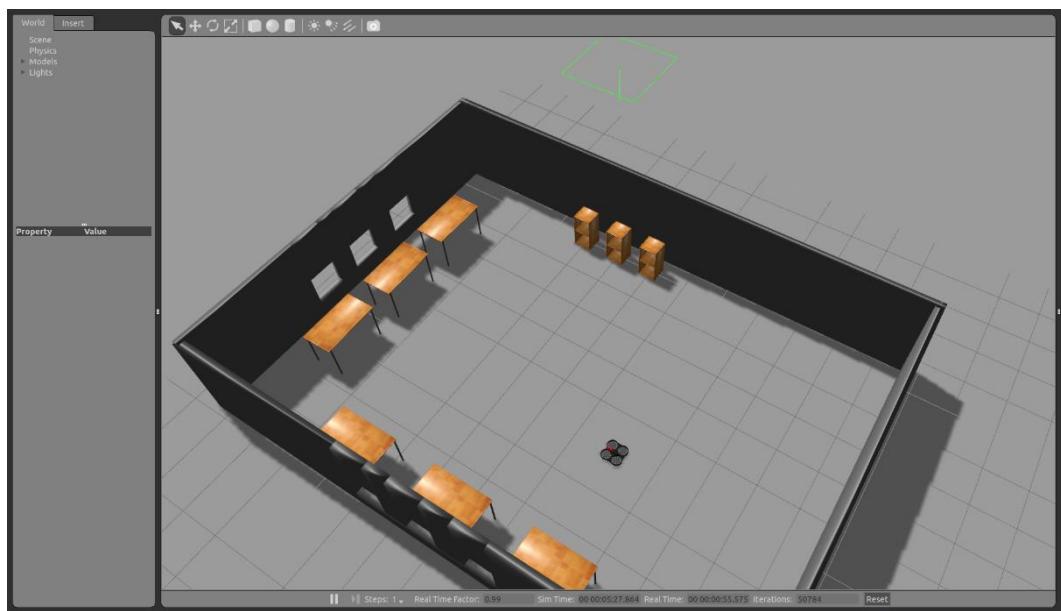
```
/home/oktat/catkin_ws/src/drone_application/launch/test_sim_lab.launch http://localhost:11311

Gazebo multi-robot simulator, version 2.2.3
Copyright (C) 2012-2014 Open Source Robotics Foundation.
Released under the Apache 2 License.
http://gazebosim.org

Msg Waiting for master.[ INFO] [1592984824.496436615]: waitForService: Service [/gazebo/set_physics_properties] has not been advertised, waiting...
[ INFO] [1592984824.512951171]: Finished loading Gazebo ROS API Plugin.
Msg Waiting for master
Msg Connected to gazebo master @ http://127.0.0.1:11345
Msg Publicized address: 192.168.1.5

Msg Connected to gazebo master @ http://127.0.0.1:11345
Msg Publicized address: 192.168.1.5
Warning [gazebo.cc:215] Waited 1seconds for namespaces.
spawn_model script started
[INFO] [WallTime: 1592984826.973262] [0.000000] Loading model xml from ros param
eter
[INFO] [WallTime: 1592984826.977401] [0.000000] Waiting for service /gazebo/spaw
n_urdf_model
Warning [gazebo.cc:215] Waited 1seconds for namespaces.
[ INFO] [1592984827.838913560, 277.103000000]: waitForService: Service [/gazebo/
set_physics_properties] is now available.
```

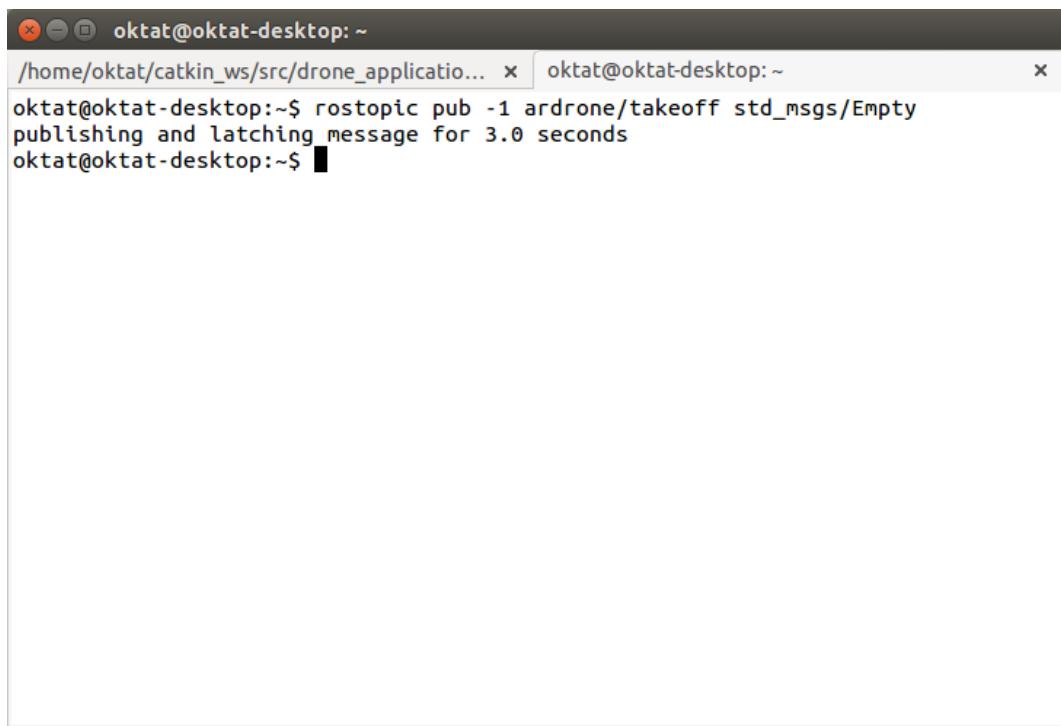
Gambar 6.4 ip ardrone



Gambar 6.5 tampilan awal simulasi lab

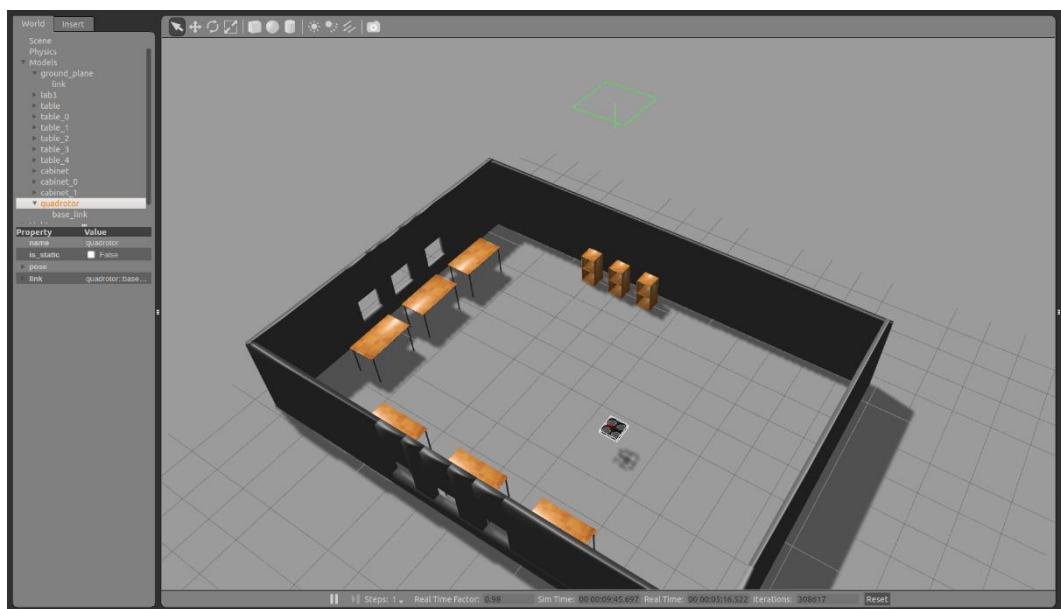
A screenshot of a terminal window titled 'oktat@oktat-desktop: ~'. The window shows a command being typed: 'rostopic pub -1 ardrone/takeoff std\_msgs/Empty'. The terminal is running on a Linux system named 'oktat'.

Gambar 6.6 perintah untuk menerbangkan ardrone

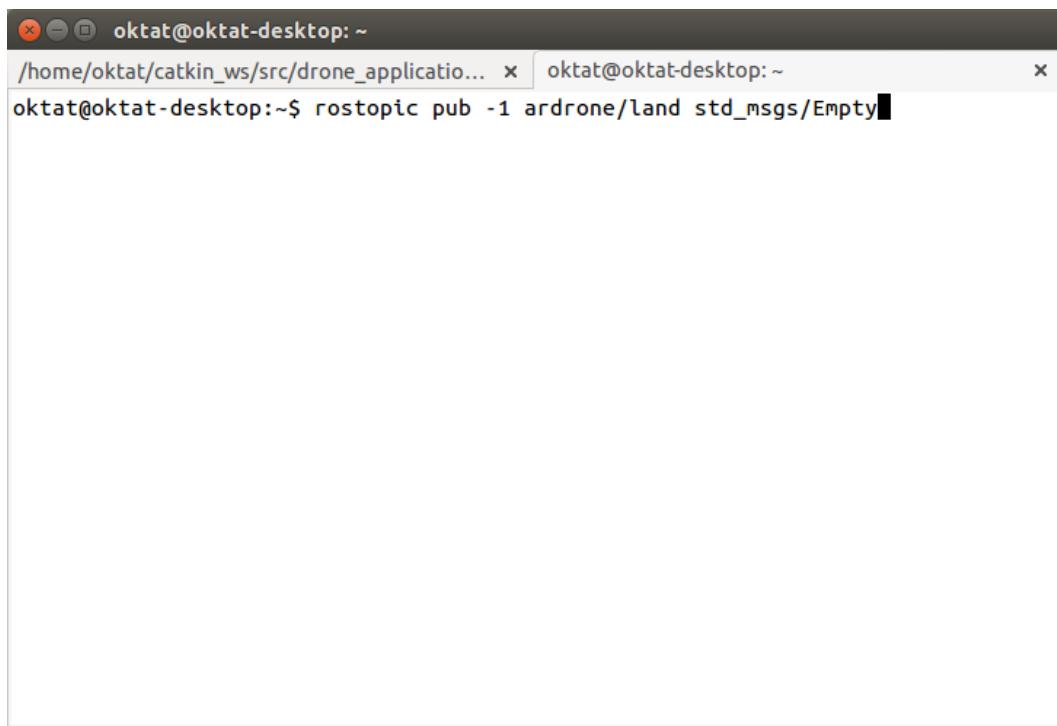


A terminal window titled 'oktat@oktat-desktop: ~' with the command '/home/oktat/catkin\_ws/src/drone\_applicatio...'. The output shows: 'oktat@oktat-desktop:~\$ rostopic pub -1 ardrone/takeoff std\_msgs/Empty publishing and latching message for 3.0 seconds oktat@oktat-desktop:~\$'. The terminal window has a dark theme.

Gambar 6.7 eksekusi terbang

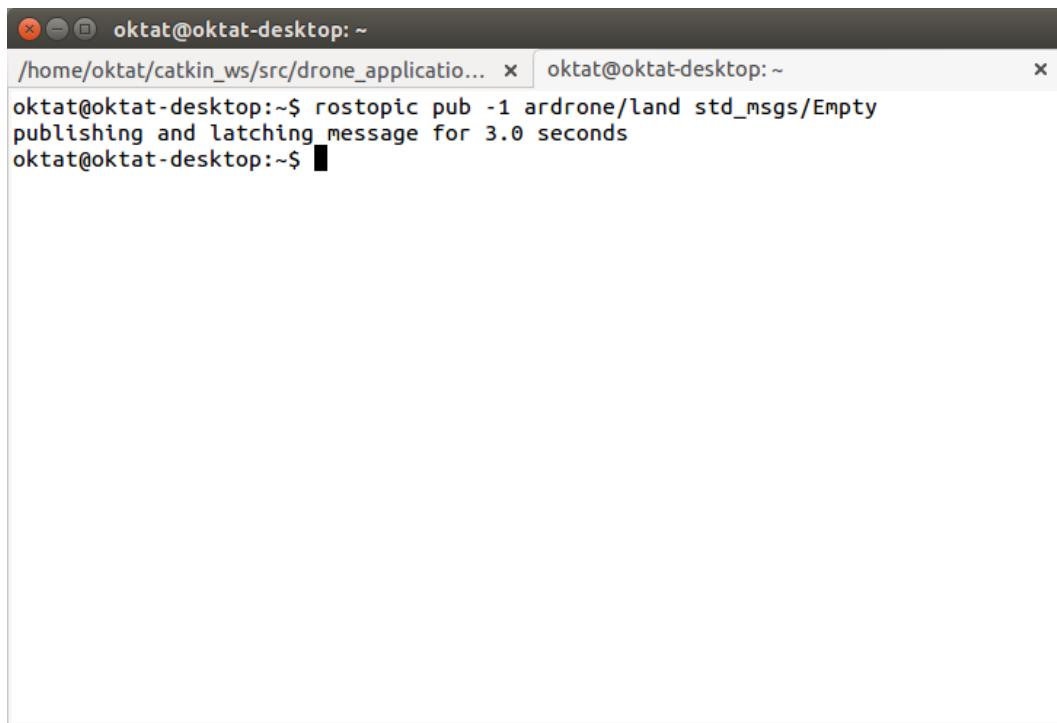


Gambar 6.8 hasil terbang



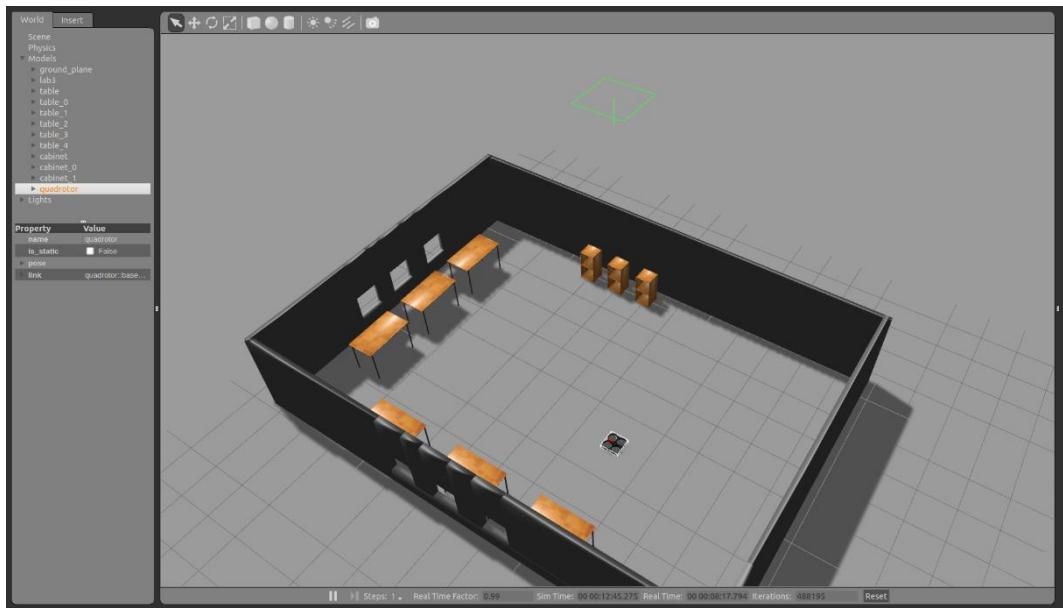
```
oktat@oktat-desktop: ~
/home/oktat/catkin_ws/src/drone_applicatio... x oktat@oktat-desktop: ~
oktat@oktat-desktop:~$ rostopic pub -1 ardronel/land std_msgs/Empty
```

Gambar 6.9 perintah untuk mendarat



```
oktat@oktat-desktop: ~
/home/oktat/catkin_ws/src/drone_applicatio... x oktat@oktat-desktop: ~
oktat@oktat-desktop:~$ rostopic pub -1 ardronel/land std_msgs/Empty
publishing and latching message for 3.0 seconds
oktat@oktat-desktop:~$
```

Gambar 6.10 eksekusi perintah mendarat



Gambar 6.11 ardrone mendarat

A screenshot of a terminal window titled "oktat@oktat-desktop: ~". The window has three tabs: "/home/oktat/catkin\_ws/src..." (active), "oktat@oktat-desktop: ~", and "oktat@oktat-desktop: ~". The active tab contains the following command and its output:

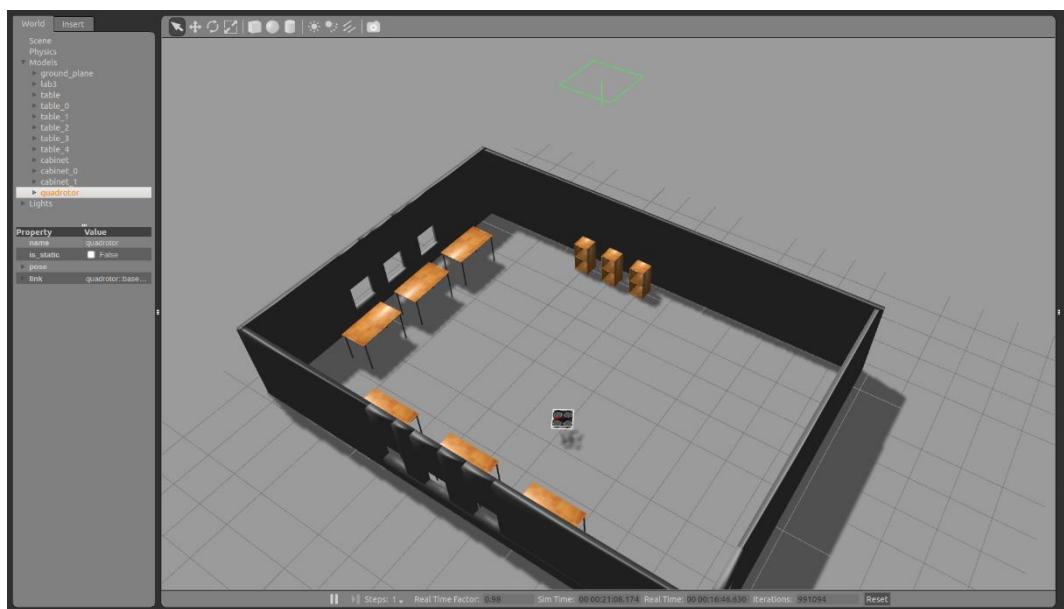
```
oktat@oktat-desktop:~$ rostopic pub -r 10 /cmd_vel geometry_msgs/Twist '{linear: {x: 0.1, y: 0.0, z: 0.0}, angular: {x: 0.0,y: 0.0,z: 0.0}}'
```

Gambar 6.12 perintah untuk maju

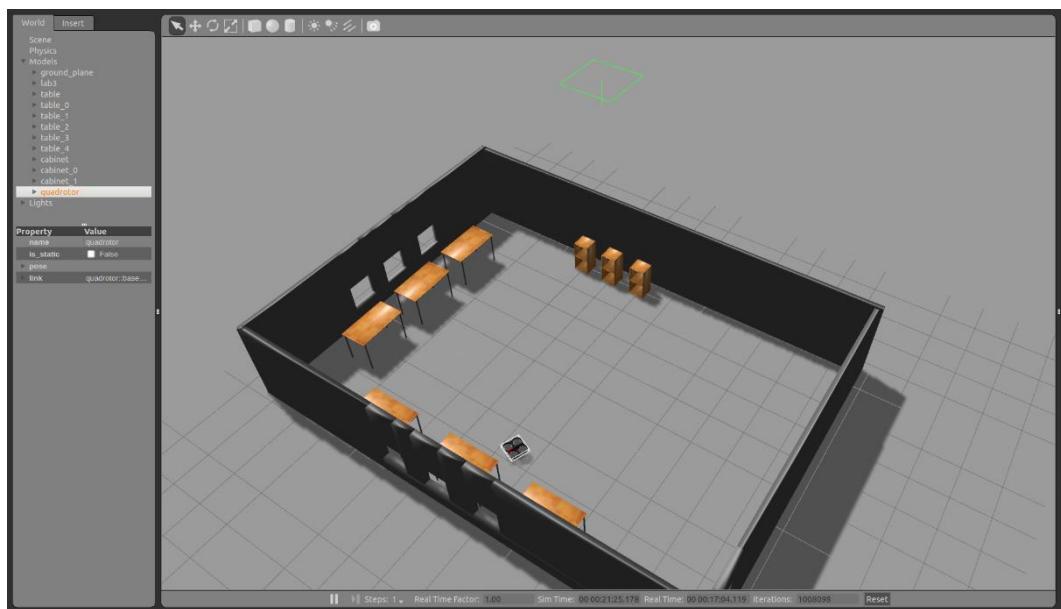


A terminal window titled 'oktat@oktat-desktop: ~' showing the command 'rostopic pub -r 10 /cmd\_vel geometry\_msgs/Twist '{linear: {x: 0.1, y: 0.0, z: 0.0}, angular: {x: 0.0,y: 0.0,z: 0.0}}' being run. The output shows the command being sent.

Gambar 6.13 eksekusi perintah untuk maju



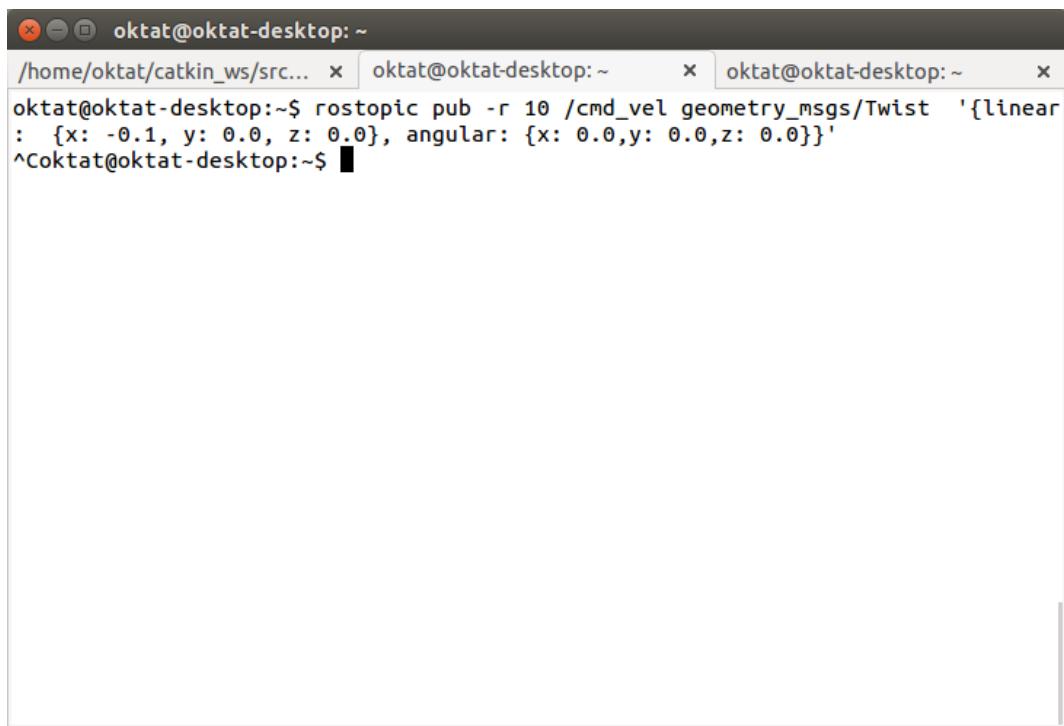
Gambar 6.14 hasil perintah maju 1



Gambar 6.15 hasil perintah maju 2

A screenshot of a terminal window titled 'oktat@oktat-desktop: ~'. It contains a single command: 'rostopic pub -r 10 /cmd\_vel geometry\_msgs/Twist '{linear: {x: -0.1, y: 0.0, z: 0.0}, angular: {x: 0.0,y: 0.0,z: 0.0}}''. The terminal is running on a Linux system named 'oktat'.

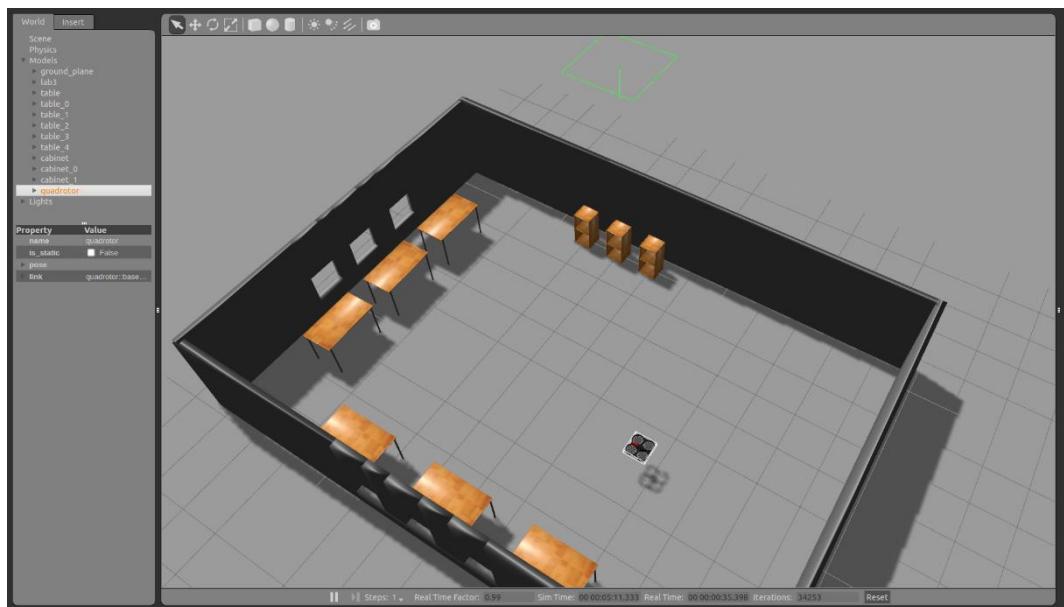
Gambar 6.16 perintah untuk mundur



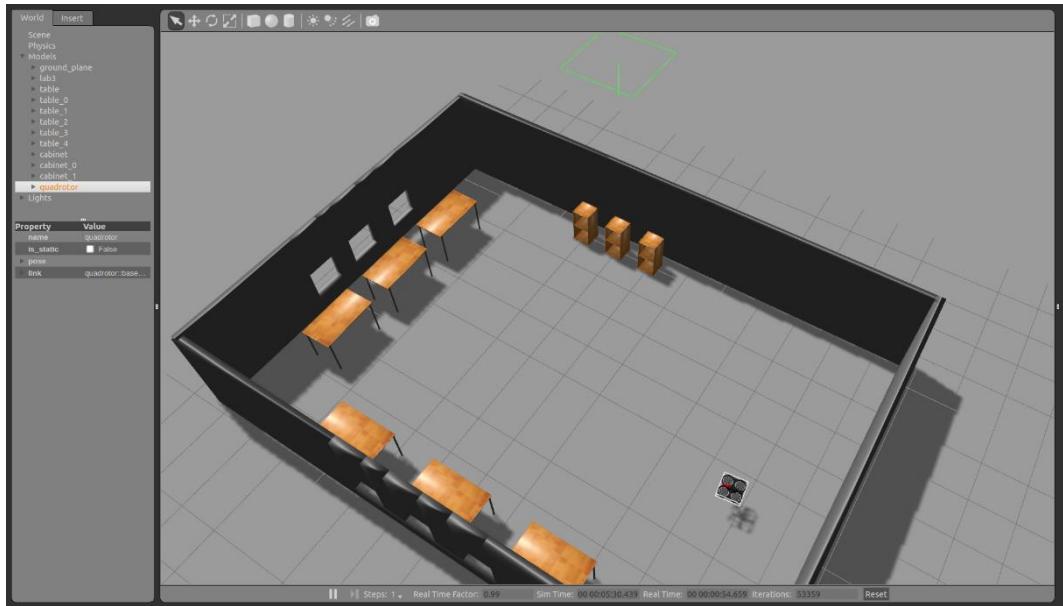
A terminal window titled 'oktat@oktat-desktop: ~' showing the execution of a ROS command. The command is 'rostopic pub -r 10 /cmd\_vel geometry\_msgs/Twist '{linear: {x: -0.1, y: 0.0, z: 0.0}, angular: {x: 0.0,y: 0.0,z: 0.0}}''. The output shows the command being sent at a rate of 10 Hz.

```
oktat@oktat-desktop:~$ rostopic pub -r 10 /cmd_vel geometry_msgs/Twist '{linear: {x: -0.1, y: 0.0, z: 0.0}, angular: {x: 0.0,y: 0.0,z: 0.0}}'
^Coktat@oktat-desktop:~$
```

Gambar 6.17 eksekusi perintah untuk mundur



Gambar 6.18 hasil perintah untuk mundur 1

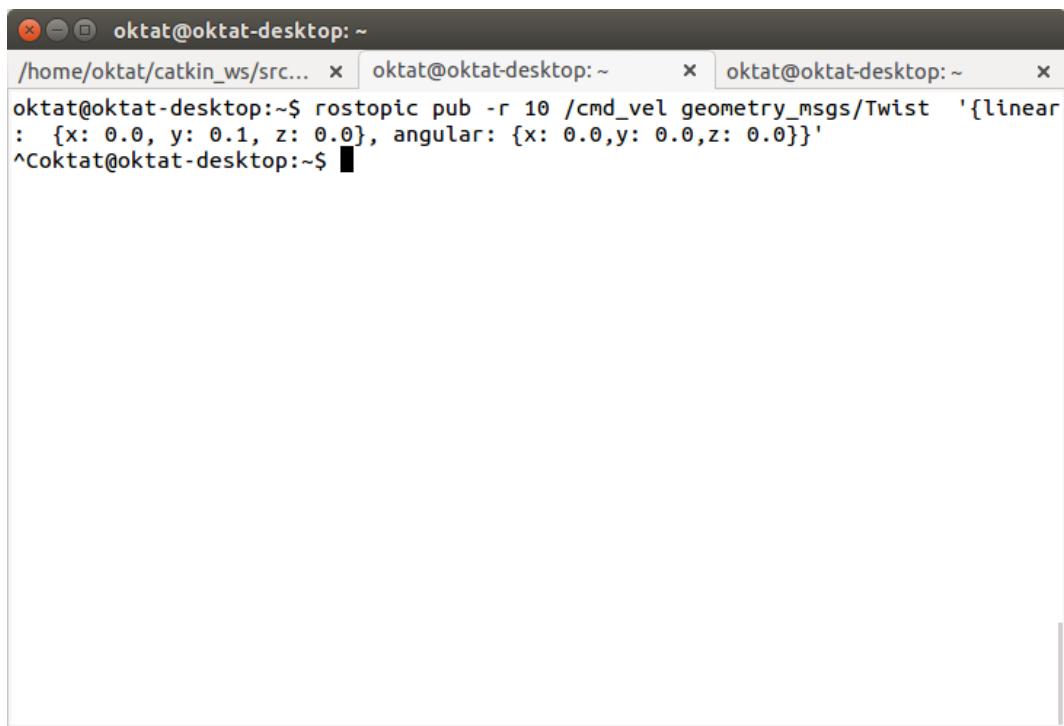


Gambar 6.19 hasil perintah untuk mundur 2

```
oktat@oktat-desktop: ~
/home/oktat/catkin_ws/src... x oktat@oktat-desktop: ~ x oktat@oktat-desktop: ~ x
oktat@oktat-desktop:~$ rostopic pub -r 10 /cmd_vel geometry_msgs/Twist '{linear: {x: 0.0, y: 0.1, z: 0.0}, angular: {x: 0.0,y: 0.0,z: 0.0}}'
```

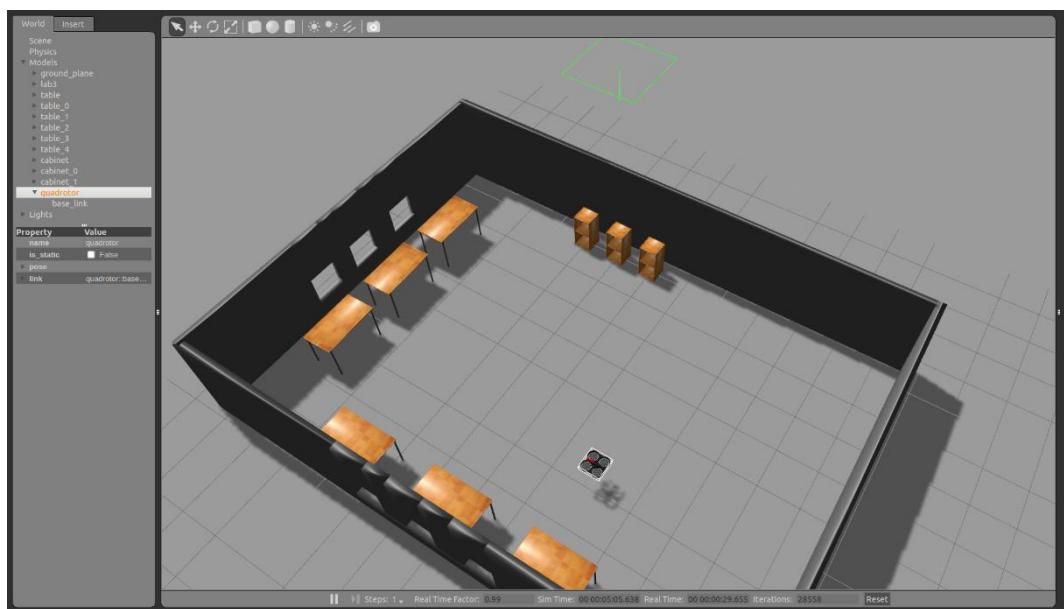
A screenshot of a terminal window titled 'oktat@oktat-desktop: ~'. It contains a single command: 'rostopic pub -r 10 /cmd\_vel geometry\_msgs/Twist '{linear: {x: 0.0, y: 0.1, z: 0.0}, angular: {x: 0.0,y: 0.0,z: 0.0}}''. This command publishes a Twist message at a rate of 10 Hz on the '/cmd\_vel' topic, specifying linear velocity in the y-direction (0.1) and zero values for other axes.

Gambar 6.20 perintah untuk bergerak ke kiri

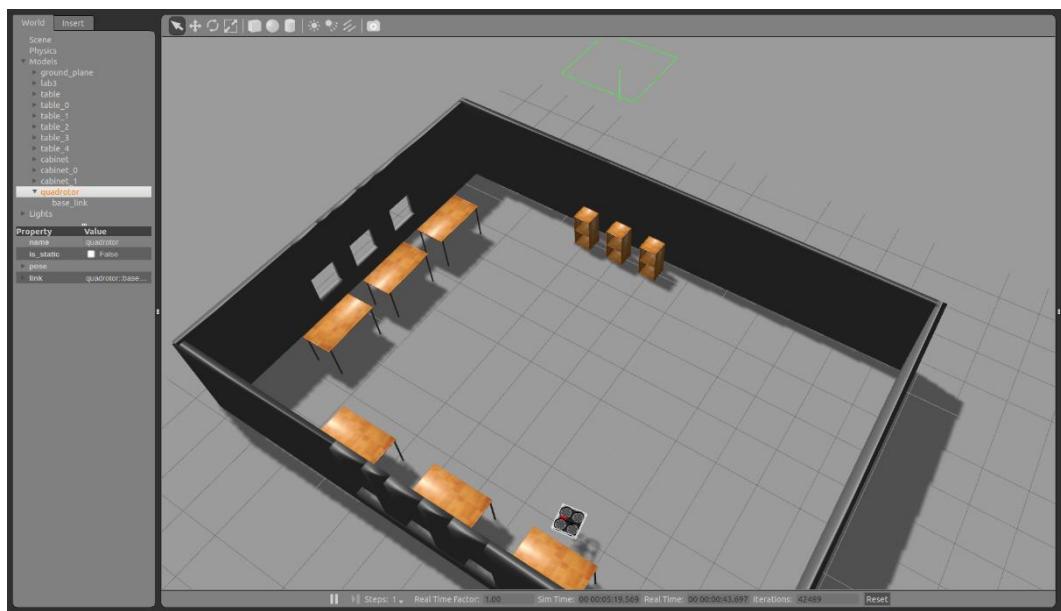


A terminal window titled 'oktat@oktat-desktop: ~' showing the execution of a ROS command. The command is 'rostopic pub -r 10 /cmd\_vel geometry\_msgs/Twist '{linear: {x: 0.0, y: 0.1, z: 0.0}, angular: {x: 0.0,y: 0.0,z: 0.0}}''. The output shows the command being sent and then ends with '^C'.

Gambar 6.21 eksekusi perintah untuk bergerak ke kiri



Gambar 6.22 hasil perintah untuk bergerak ke kiri 1

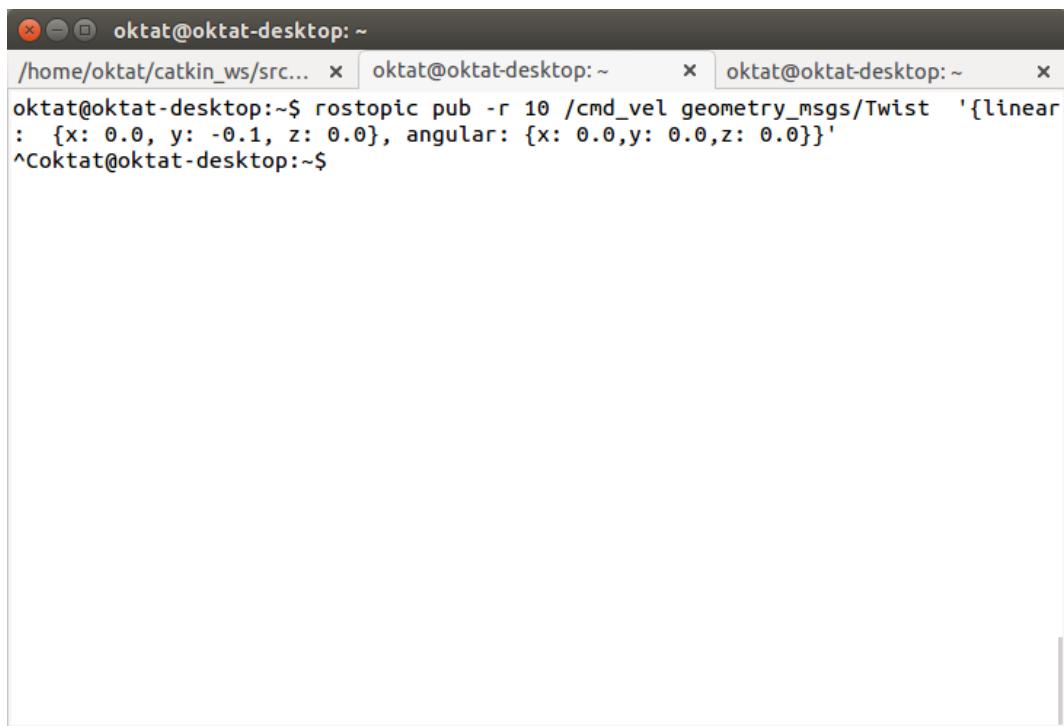


Gambar 6.23 hasil perintah untuk bergerak ke kiri 2

```
oktat@oktat-desktop: ~
/home/oktat/catkin_ws/src... x oktat@oktat-desktop: ~ x oktat@oktat-desktop: ~ x
oktat@oktat-desktop:~$ rostopic pub -r 10 /cmd_vel geometry_msgs/Twist '{linear: {x: 0.0, y: -0.1, z: 0.0}, angular: {x: 0.0,y: 0.0,z: 0.0}}'
```

A screenshot of a terminal window titled 'oktat@oktat-desktop: ~'. The terminal is running a ROS command to publish a Twist message at a rate of 10 Hz on the '/cmd\_vel' topic. The message specifies linear velocity in the y-direction (-0.1) and angular velocity around the y-axis (0.0).

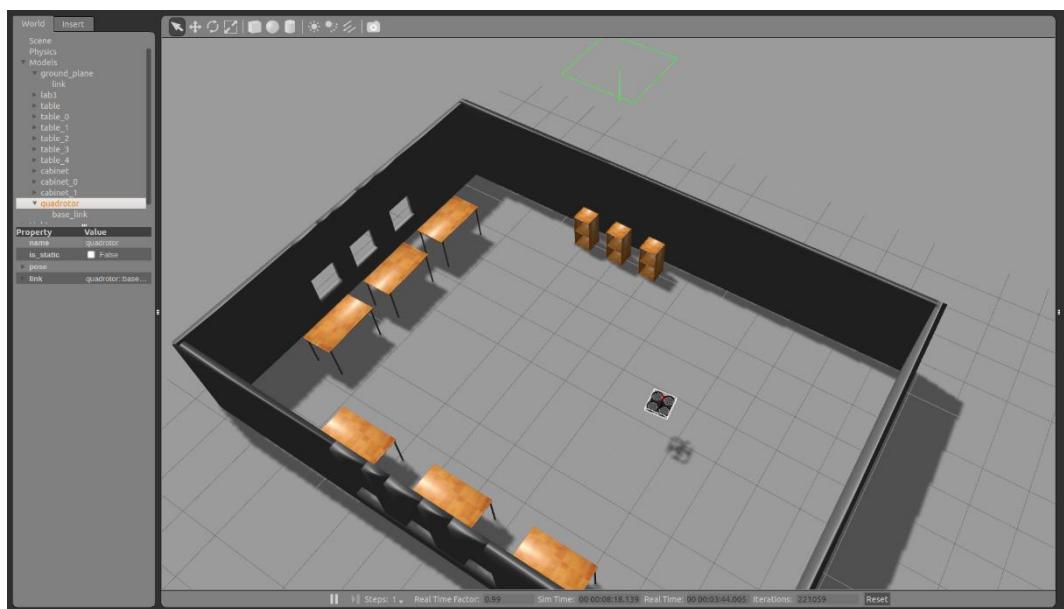
Gambar 6.24 perintah untuk bergerak ke kanan



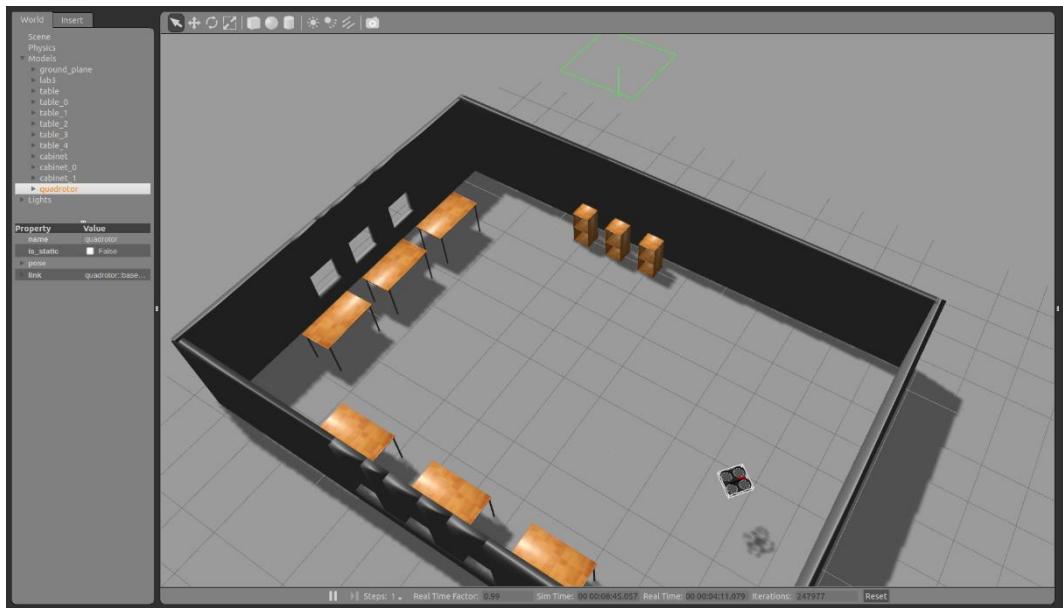
A terminal window titled 'oktat@oktat-desktop: ~' containing the following command and its output:

```
/home/oktat/catkin_ws/src... x oktat@oktat-desktop: ~ x oktat@oktat-desktop: ~ x
oktat@oktat-desktop:~$ rostopic pub -r 10 /cmd_vel geometry_msgs/Twist '{linear: {x: 0.0, y: -0.1, z: 0.0}, angular: {x: 0.0,y: 0.0,z: 0.0}}'
^Coktat@oktat-desktop:~$
```

Gambar 6.25 eksekusi perintah untuk bergerak ke kanan



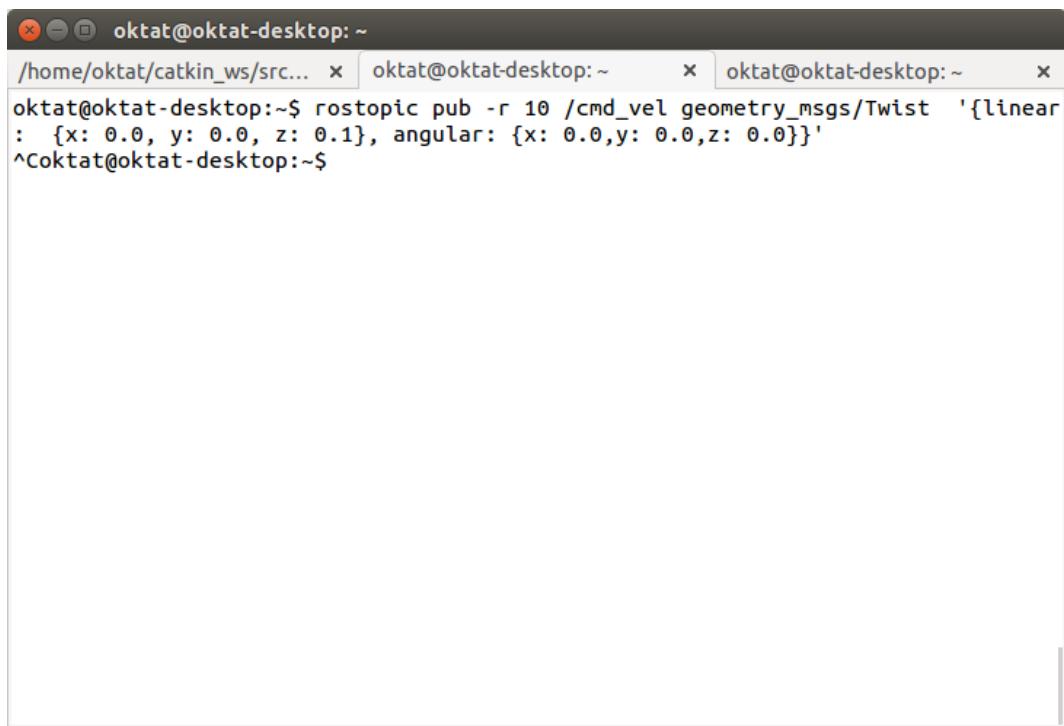
Gambar 6.26 hasil perintah untuk bergerak ke kanan



Gambar 6.27 hasil perintah untuk bergerak ke kanan

A screenshot of a terminal window titled 'oktat@oktat-desktop: ~'. It contains a single command: 'rostopic pub -r 10 /cmd\_vel geometry\_msgs/Twist '{linear: {x: 0.0, y: 0.0, z: 0.1}, angular: {x: 0.0,y: 0.0,z: 0.0}}''. The terminal is part of a larger window with multiple tabs.

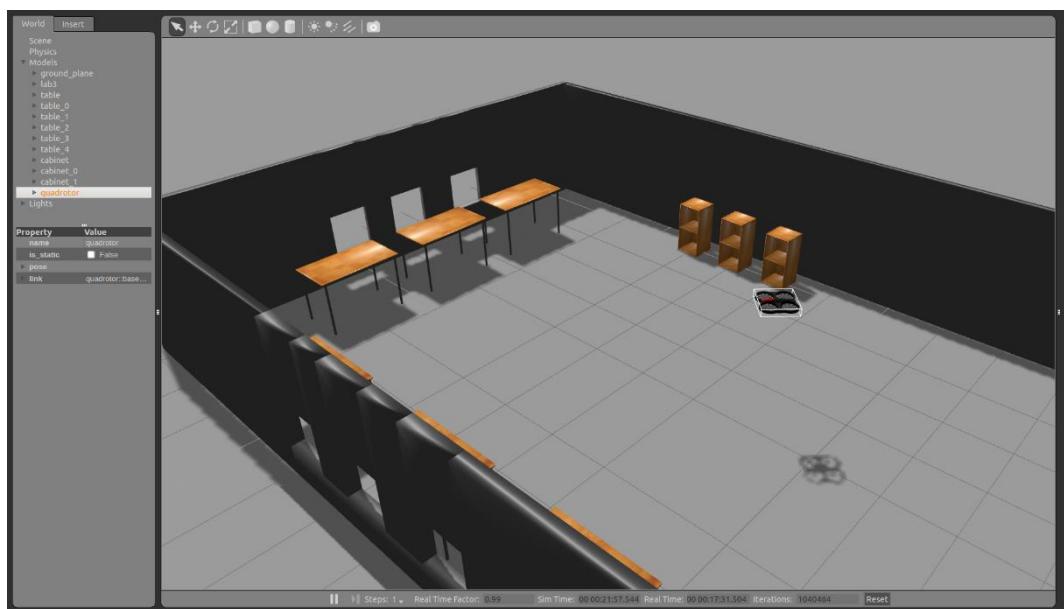
Gambar 6.28 perintah untuk bergerak ke naik



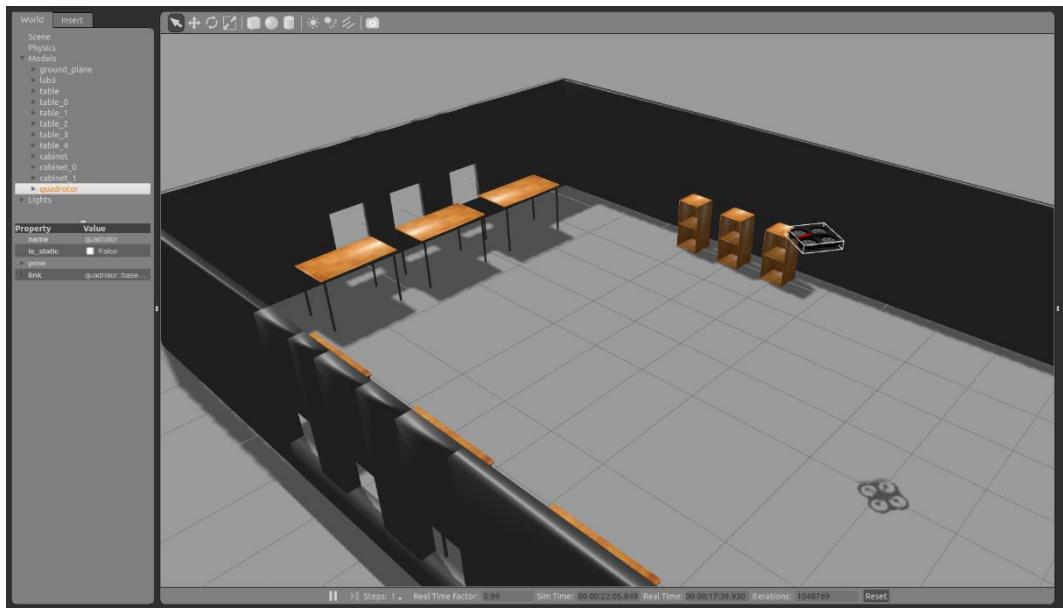
A terminal window titled "oktat@oktat-desktop: ~" containing the following command and its output:

```
/home/oktat/catkin_ws/src... x oktat@oktat-desktop: ~ x oktat@oktat-desktop: ~ x
oktat@oktat-desktop:~$ rostopic pub -r 10 /cmd_vel geometry_msgs/Twist '{linear: {x: 0.0, y: 0.0, z: 0.1}, angular: {x: 0.0,y: 0.0,z: 0.0}}'
^Coktat@oktat-desktop:~$
```

Gambar 6.29 eksekusi perintah untuk bergerak ke naik



Gambar 6.30 hasil perintah untuk bergerak ke naik



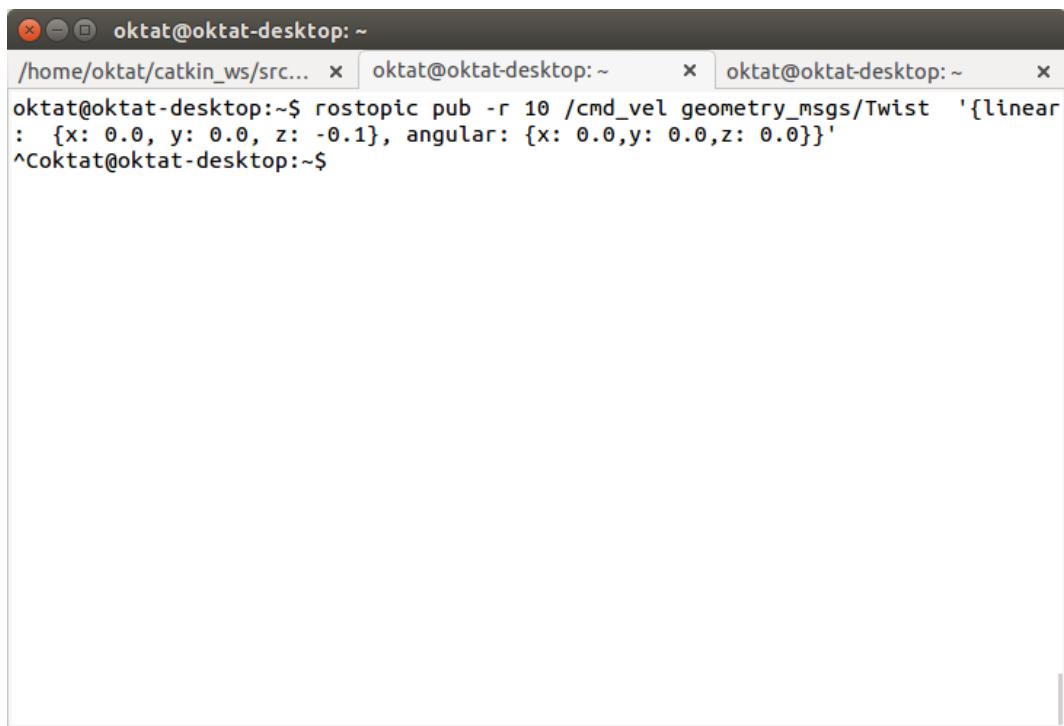
Gambar 6.31 hasil perintah untuk bergerak ke naik 2

A screenshot of a terminal window titled "oktat@oktat-desktop: ~". The terminal is running on a Linux system. The user has opened two tabs, both showing the same command. The command is: 

```
rostopic pub -r 10 /cmd_vel geometry_msgs/Twist '{linear: {x: 0.0, y: 0.0, z: -0.1}, angular: {x: 0.0,y: 0.0,z: 0.0}}'
```

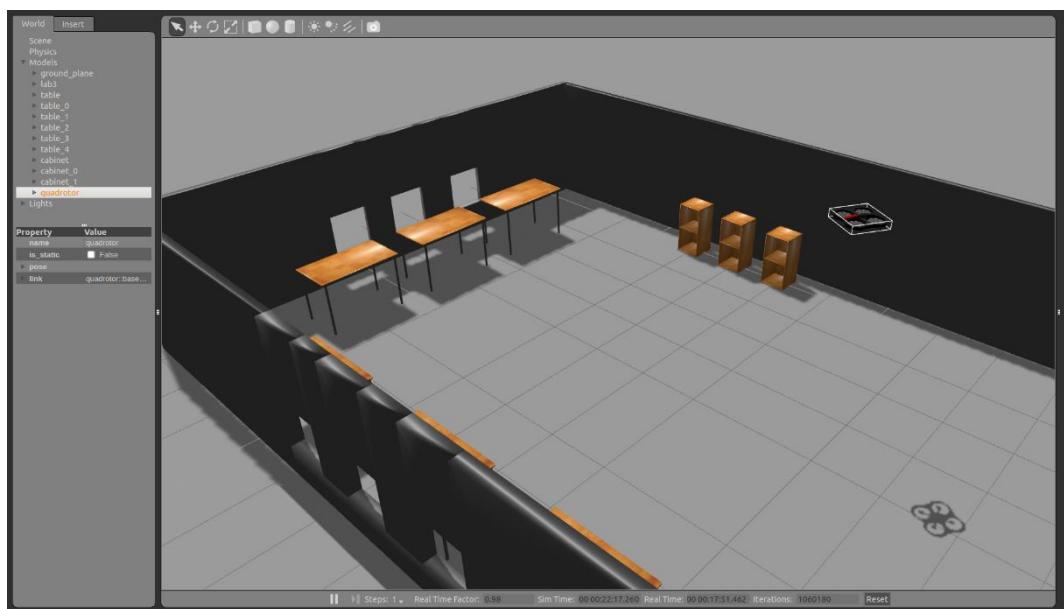
 This command publishes a Twist message at a rate of 10 Hz on the "/cmd\_vel" topic, specifying linear velocity in the z-axis (-0.1) and zero angular velocity.

Gambar 6.32 perintah untuk bergerak turun

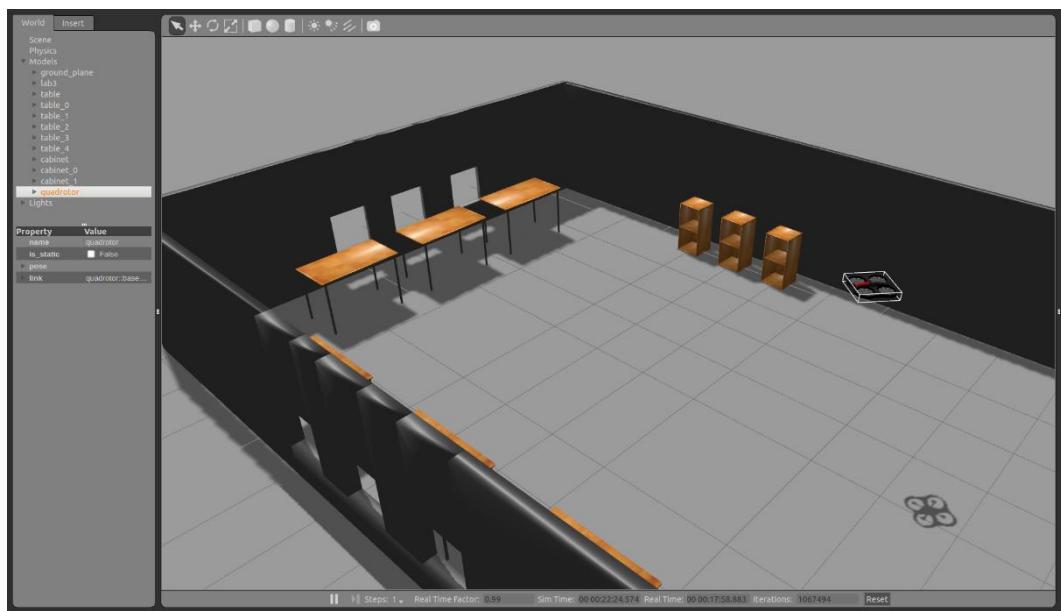


```
oktat@oktat-desktop: ~
/home/oktat/catkin_ws/src... x oktat@oktat-desktop: ~ x oktat@oktat-desktop: ~ x
oktat@oktat-desktop:~$ rostopic pub -r 10 /cmd_vel geometry_msgs/Twist '{linear: {x: 0.0, y: 0.0, z: -0.1}, angular: {x: 0.0,y: 0.0,z: 0.0}}'
^Coktat@oktat-desktop:~$
```

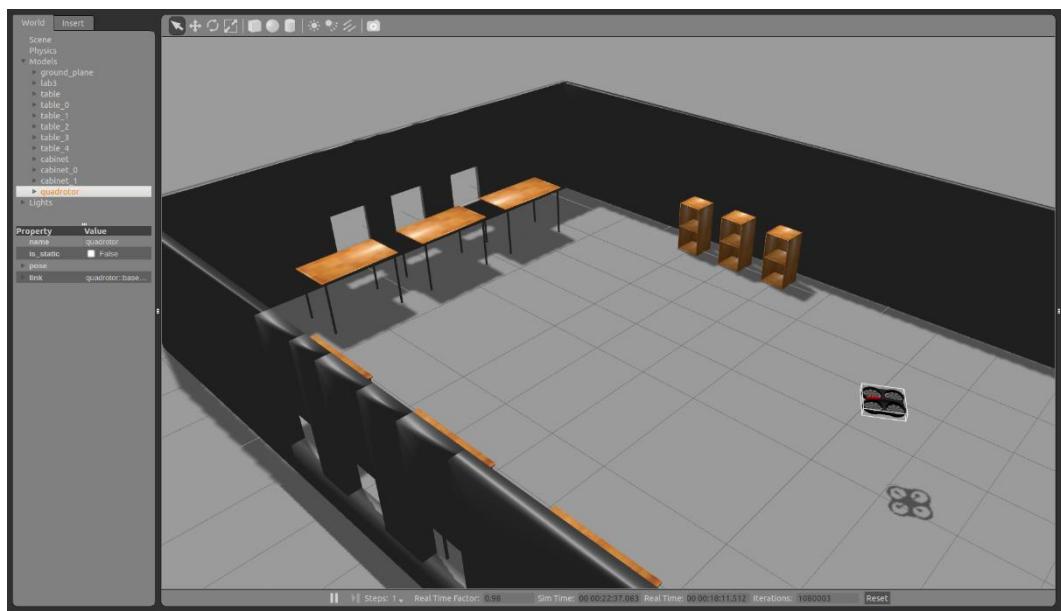
Gambar 6.33 eksekusi perintah untuk bergerak turun



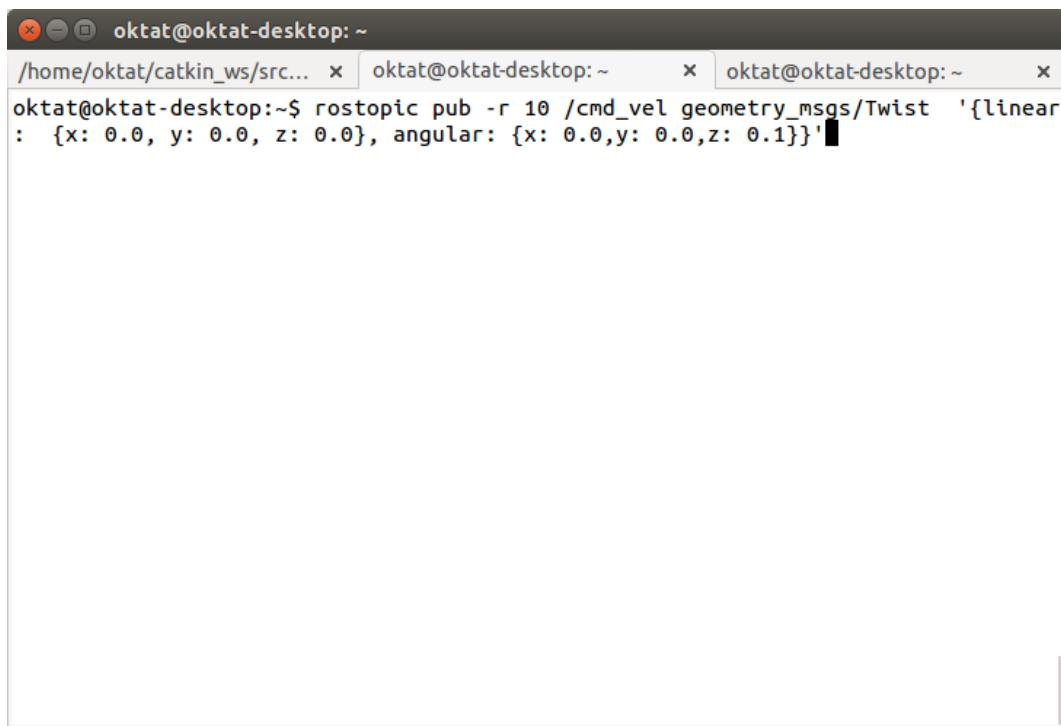
Gambar 6.34 hasil perintah untuk bergerak turun



Gambar 6.35 hasil perintah untuk bergerak turun

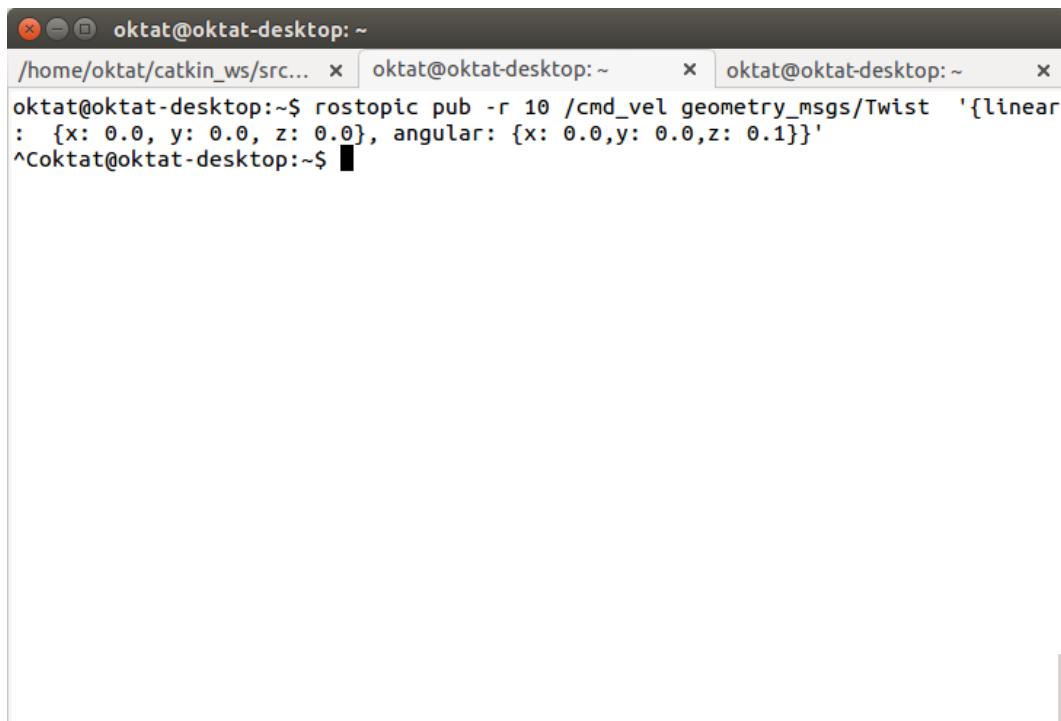


Gambar 6.36 hasil perintah untuk bergerak turun



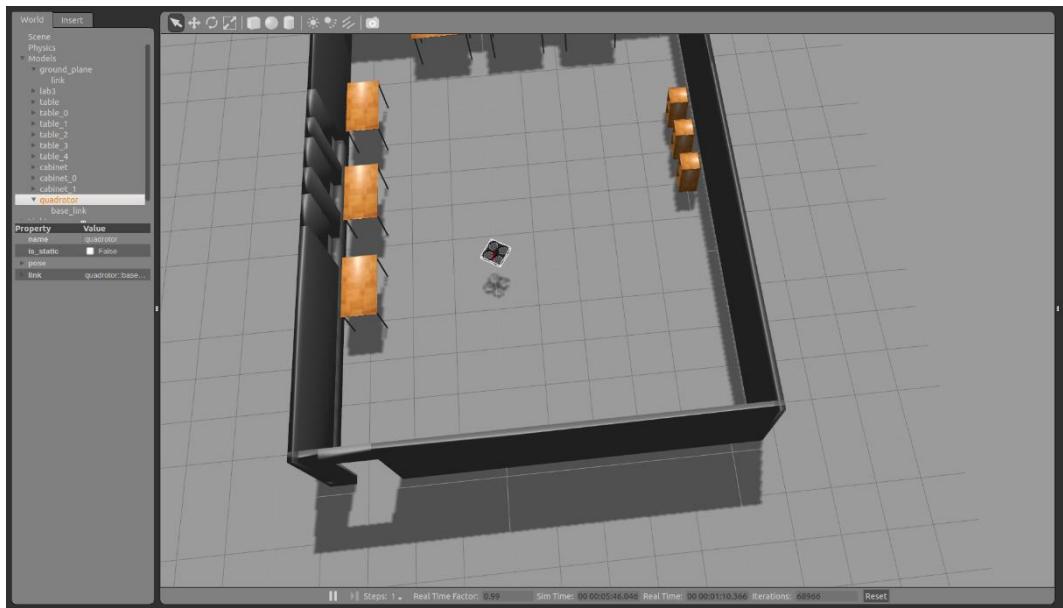
```
oktat@oktat-desktop: ~
/home/oktat/catkin_ws/src... x oktat@oktat-desktop: ~ x oktat@oktat-desktop: ~ x
oktat@oktat-desktop:~$ rostopic pub -r 10 /cmd_vel geometry_msgs/Twist '{linear
: {x: 0.0, y: 0.0, z: 0.0}, angular: {x: 0.0,y: 0.0,z: 0.1}}'
```

Gambar 6.37 perintah untuk bergerak berlawanan arah jarum jam

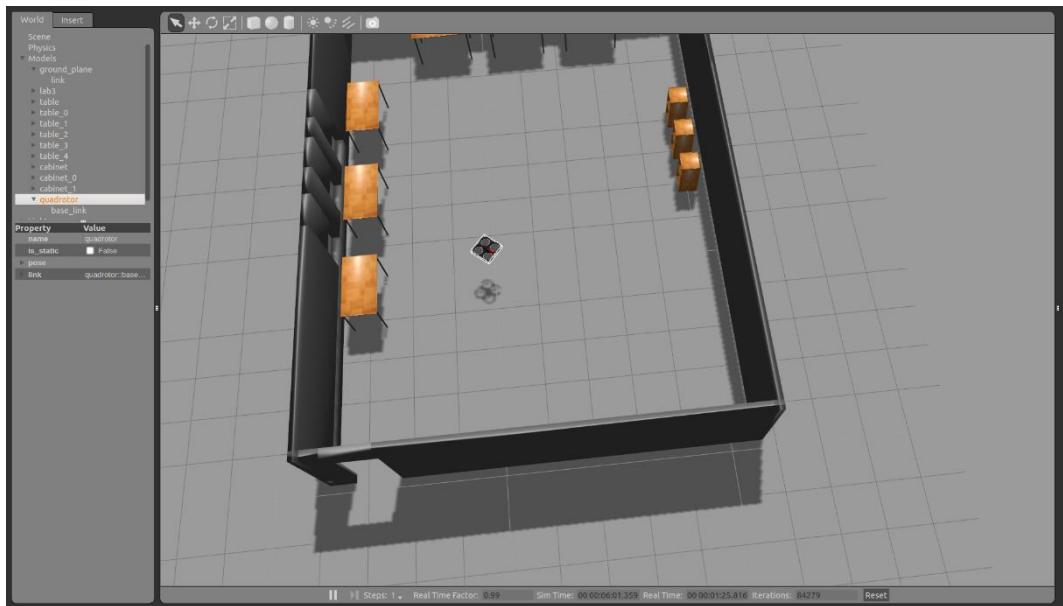


```
oktat@oktat-desktop: ~
/home/oktat/catkin_ws/src... x oktat@oktat-desktop: ~ x oktat@oktat-desktop: ~ x
oktat@oktat-desktop:~$ rostopic pub -r 10 /cmd_vel geometry_msgs/Twist '{linear
: {x: 0.0, y: 0.0, z: 0.0}, angular: {x: 0.0,y: 0.0,z: 0.1}}'
^Coktat@oktat-desktop:~$
```

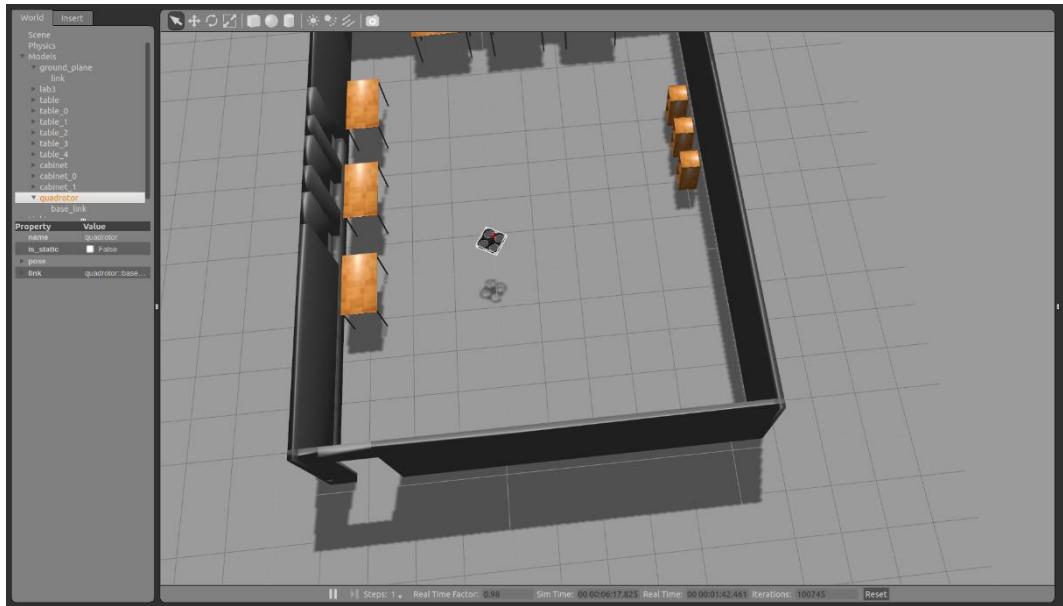
Gambar 6.38 eksekusi perintah untuk bergerak berlawanan arah jarum jam



Gambar 6.39 hasil perintah untuk bergerak berlawanan arah jarum jam 1



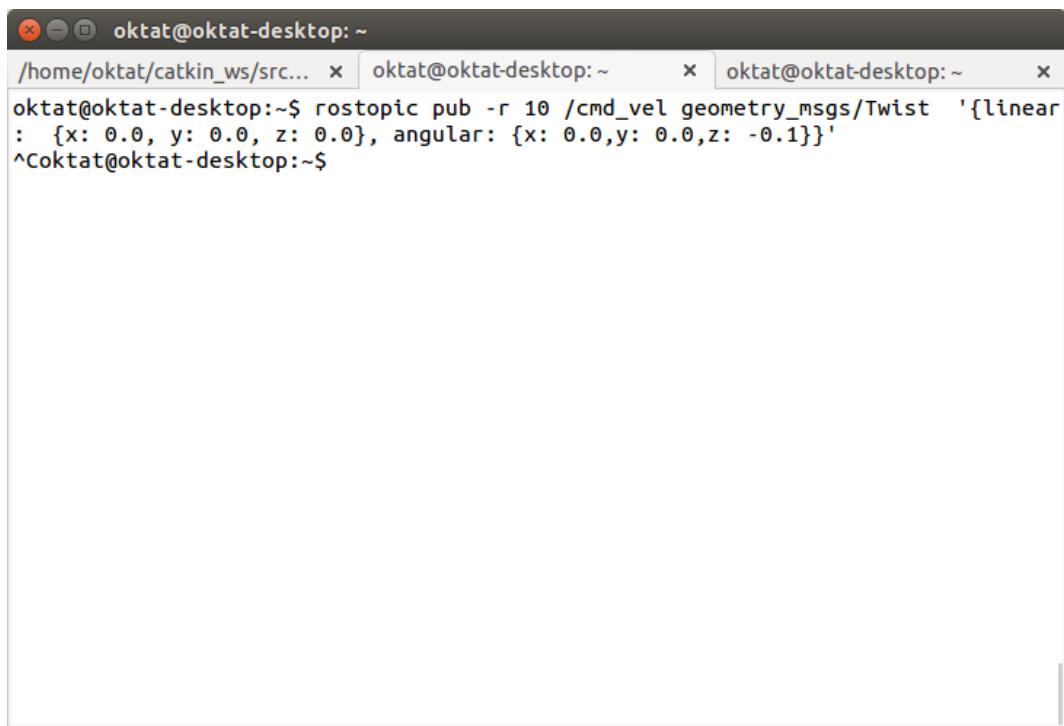
Gambar 6.40 hasil perintah untuk bergerak berlawanan arah jarum jam 2



Gambar 6.41 hasil perintah untuk bergerak berlawanan arah jarum jam 3

A screenshot of a terminal window titled "oktat@oktat-desktop: ~". The terminal shows a command being typed: "rostopic pub -r 10 /cmd\_vel geometry\_msgs/Twist '{linear: {x: 0.0, y: 0.0, z: 0.0}, angular: {x: 0.0,y: 0.0,z: -0.1}}'". The terminal window has three tabs open, but only the current tab is active.

Gambar 6.42 perintah untuk bergerak searah jarum jam



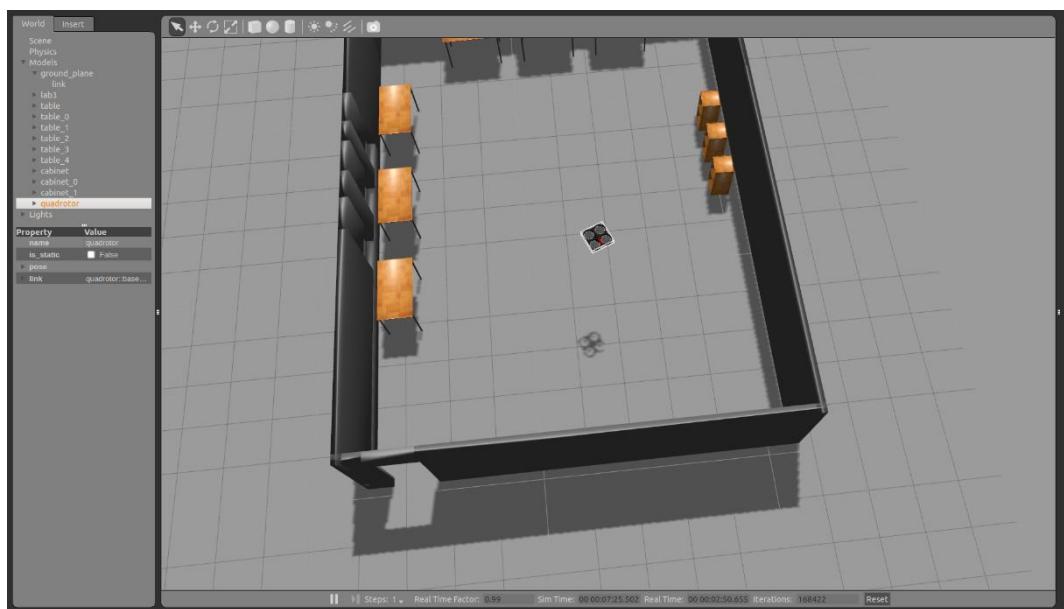
```
oktat@oktat-desktop: ~
```

```
/home/oktat/catkin_ws/src... x oktat@oktat-desktop: ~ x oktat@oktat-desktop: ~ x
```

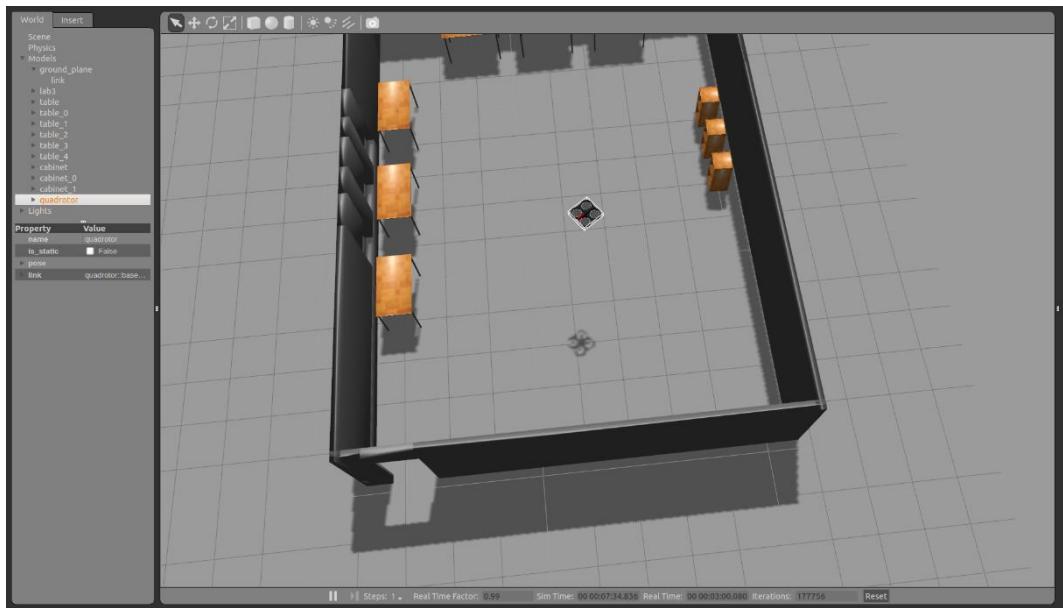
```
oktat@oktat-desktop:~$ rostopic pub -r 10 /cmd_vel geometry_msgs/Twist '{linear: {x: 0.0, y: 0.0, z: 0.0}, angular: {x: 0.0,y: 0.0,z: -0.1}}'
```

```
^Coktat@oktat-desktop:~$
```

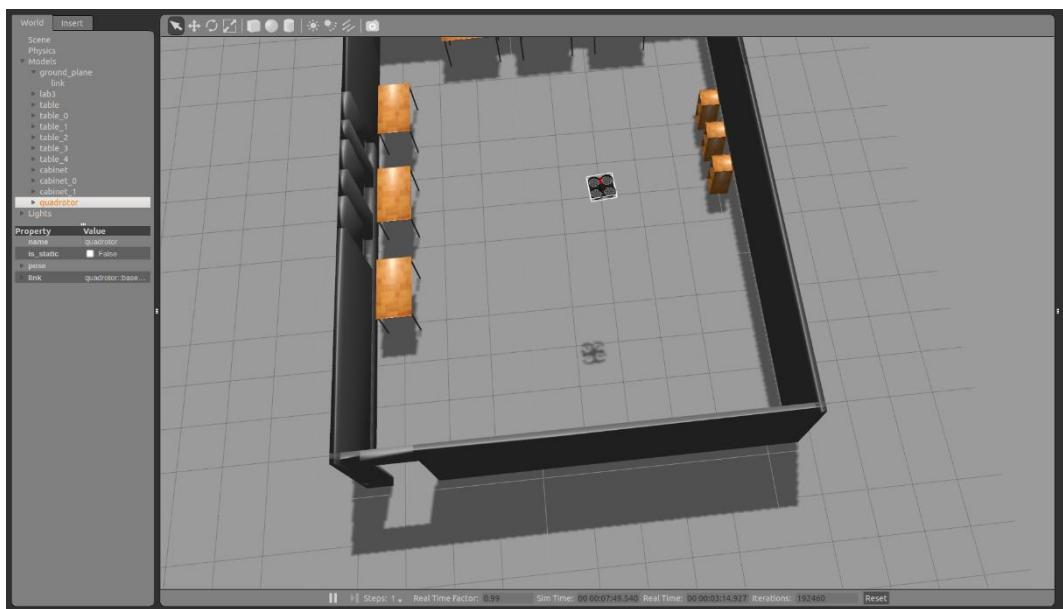
Gambar 6.43 eksekusi perintah untuk bergerak searah jarum jam



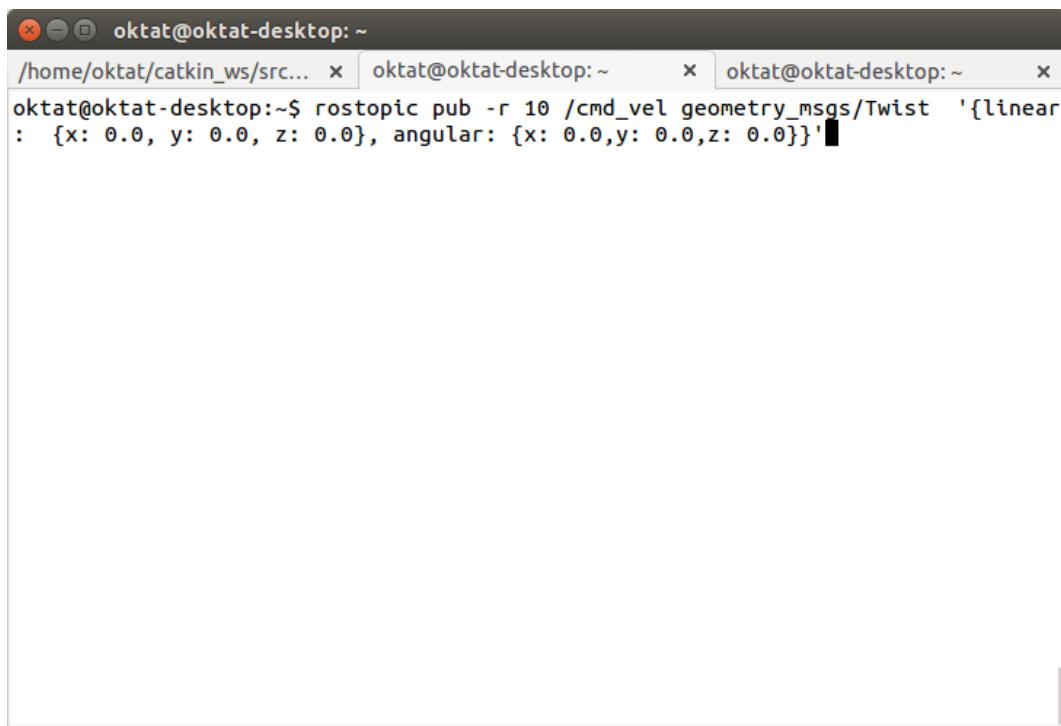
Gambar 6.44 hasil perintah untuk bergerak searah jarum jam 1



Gambar 6.45 hasil perintah untuk bergerak searah jarum jam 2

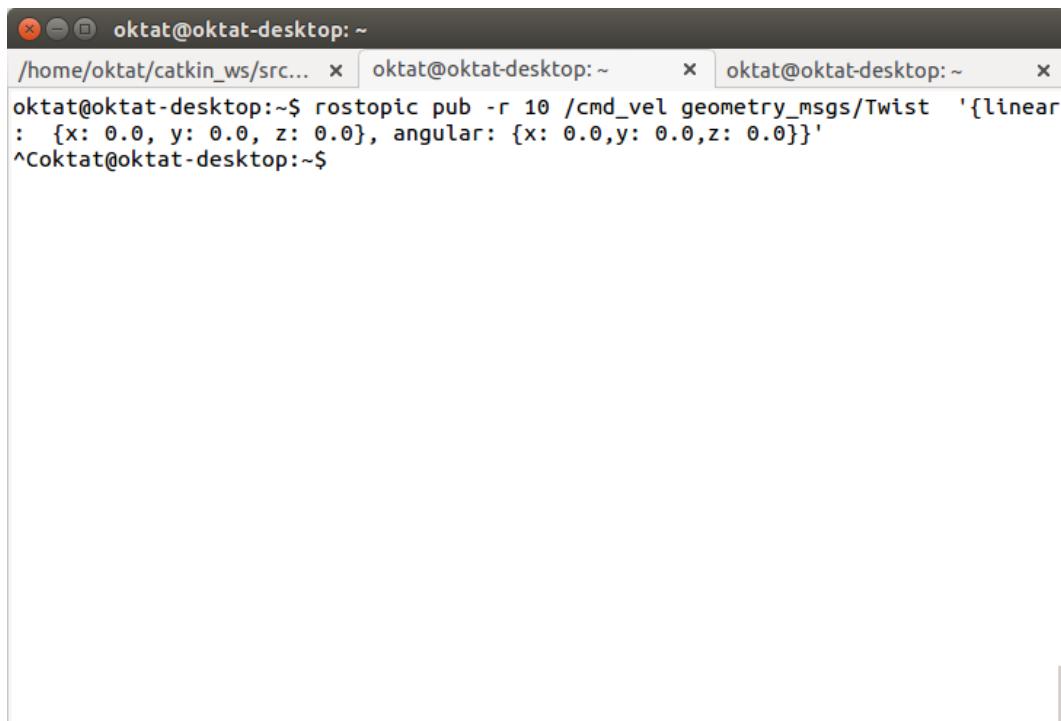


Gambar 6.46 hasil perintah untuk bergerak searah jarum jam 3



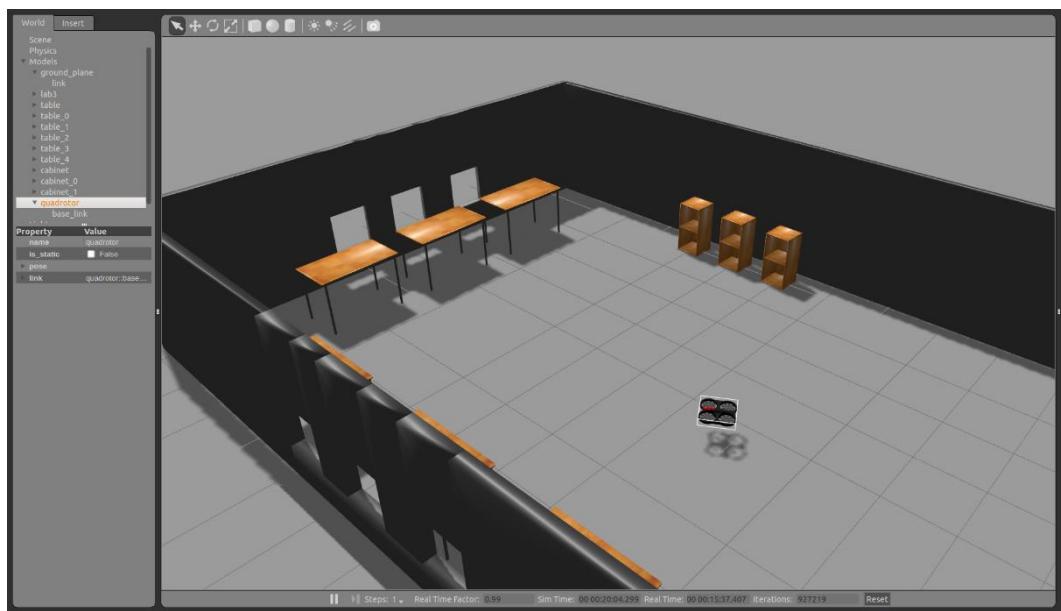
```
oktat@oktat-desktop:~$ rostopic pub -r 10 /cmd_vel geometry_msgs/Twist '{linear: {x: 0.0, y: 0.0, z: 0.0}, angular: {x: 0.0,y: 0.0,z: 0.0}}'
```

Gambar 6.47 perintah untuk diam di udara

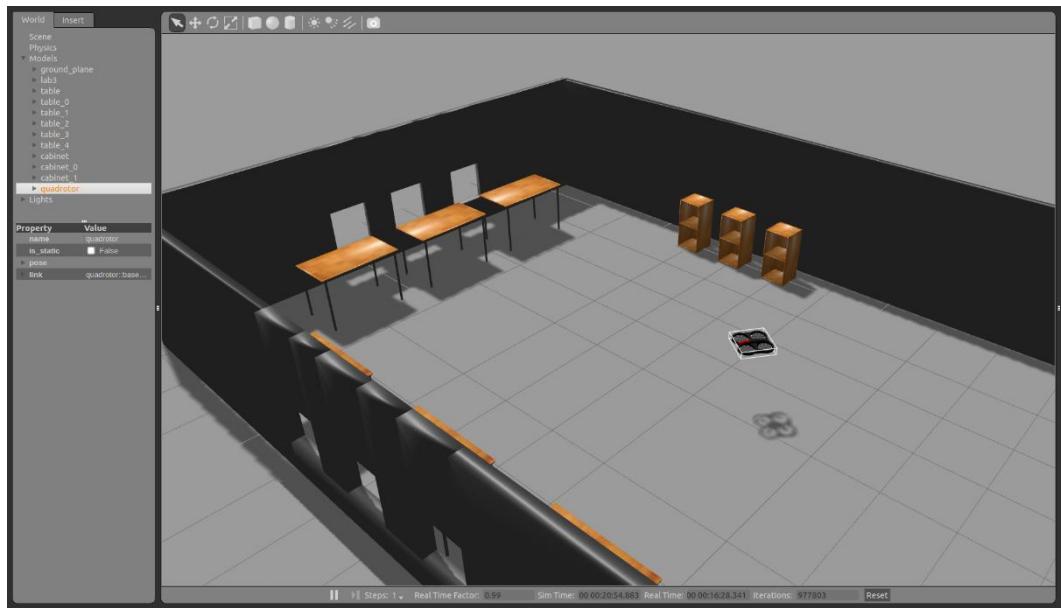


```
oktat@oktat-desktop:~$ rostopic pub -r 10 /cmd_vel geometry_msgs/Twist '{linear: {x: 0.0, y: 0.0, z: 0.0}, angular: {x: 0.0,y: 0.0,z: 0.0}}'  
^Coktat@oktat-desktop:~$
```

Gambar 6.48 eksekusi perintah untuk diam di udara



Gambar 6.49 hasil perintah untuk diam di udara 1

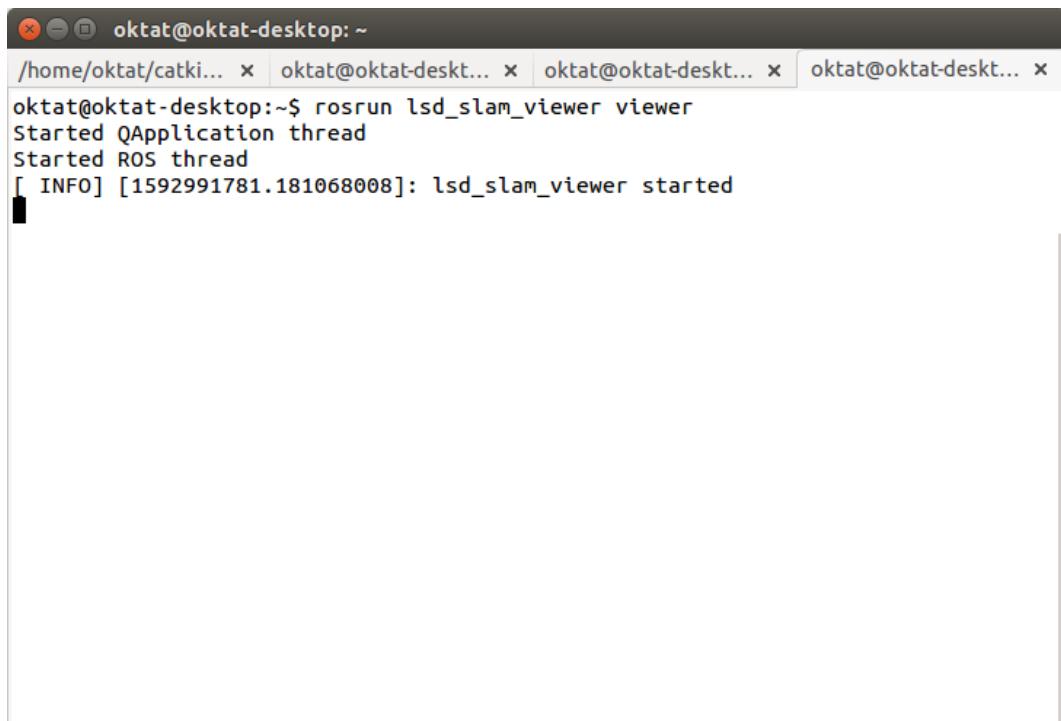


Gambar 6.50 hasil perintah untuk diam di udara 2



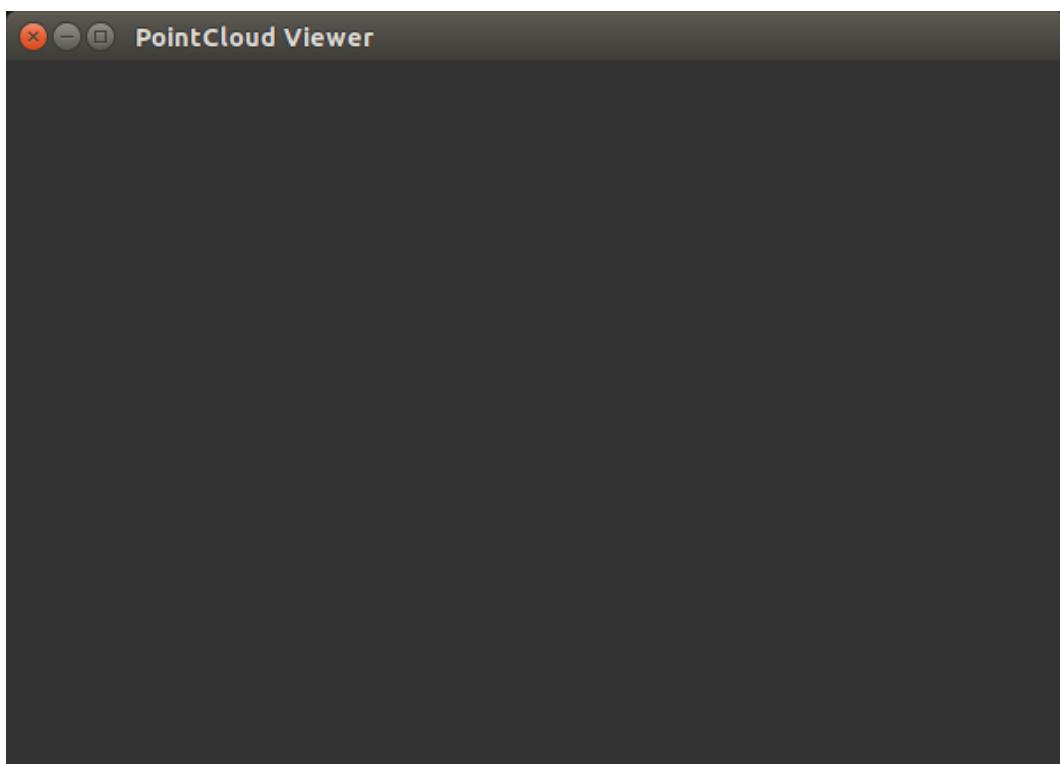
```
oktat@oktat-desktop: ~
/home/oktat/catki... x oktat@oktat-deskt... x oktat@oktat-deskt... x oktat@oktat-deskt... x
oktat@oktat-desktop:~$ rosrun lsd_slam_viewer viewer
```

Gambar 6.51 perintah untuk menjalankan viewer slam



```
oktat@oktat-desktop: ~
/home/oktat/catki... x oktat@oktat-deskt... x oktat@oktat-deskt... x oktat@oktat-deskt... x
oktat@oktat-desktop:~$ rosrun lsd_slam_viewer viewer
Started QApplication thread
Started ROS thread
[ INFO] [1592991781.181068008]: lsd_slam_viewer started
```

Gambar 6.52 eksekusi perintah untuk menjalankan viewer slam



Gambar 6.53 hasil jendela perintah untuk menjalankan viewer slam

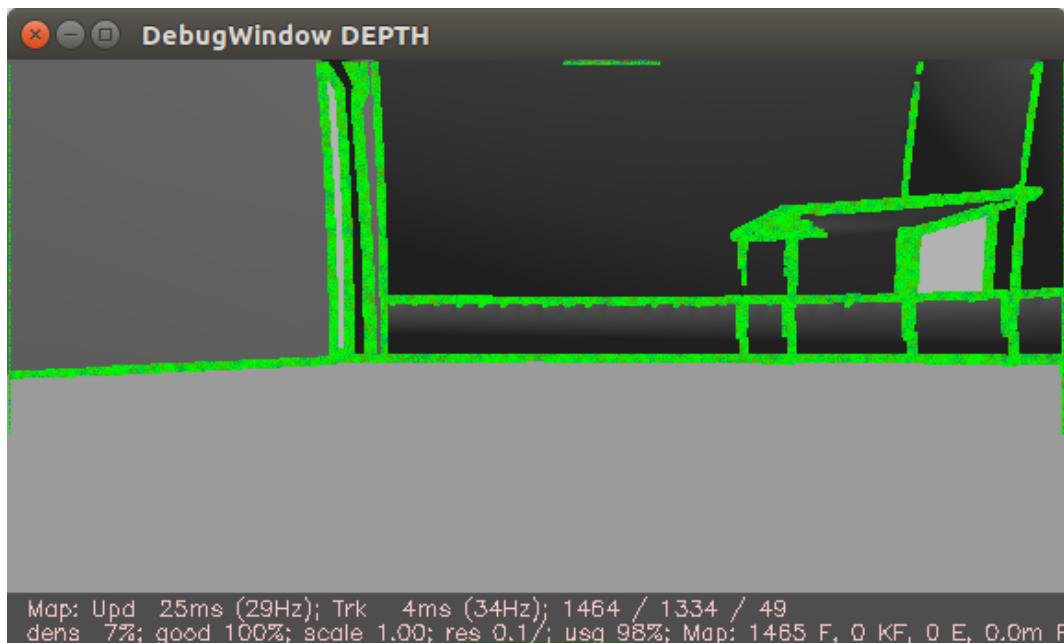
A screenshot of a terminal window titled "oktat@oktat-desktop: ~". The window contains the following command:

```
oktat@oktat-desktop:~$ rosrun lsd_slam_core live_slam image:=~/ardrone/front/image_raw _calib:=/home/oktat/catkin_ws/cameraconfig/ardrone_front_97_1.cfg
```

Gambar 6.54 perintah untuk menjalankan slam

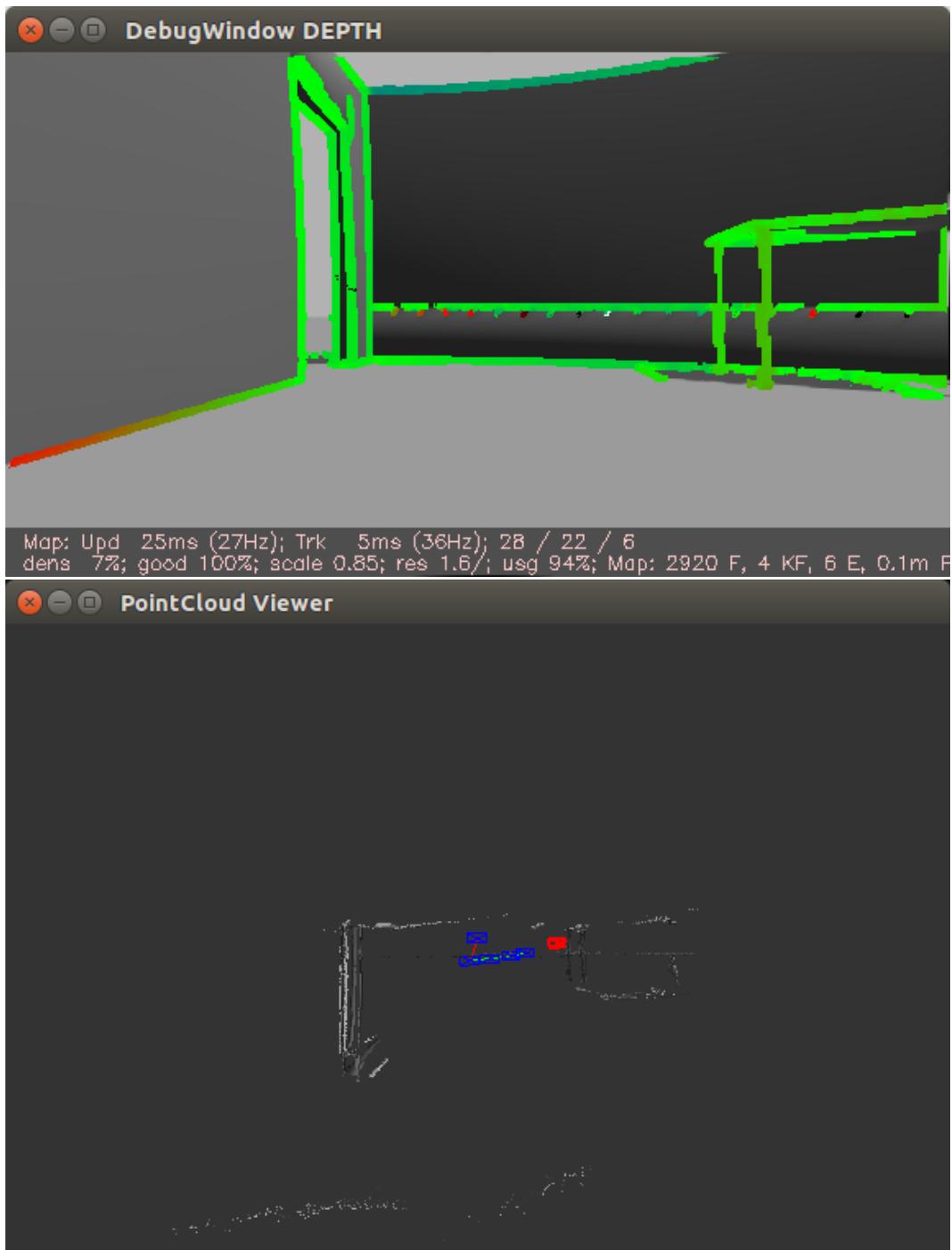
```
oktat@oktat-desktop: ~$ roslaunch lsd_slam_core live_slam image:=~/ardrone/front/image_raw _calib:=~/home/oktat/catkin_ws/cameraconfig/ardrone_front_97_1.cfg  
Reading Calibration from file /home/oktat/catkin_ws/cameraconfig/ardrone_front_97_1.cfg ... found!  
found OpenCV camera model, building rectifier.  
Input resolution: 640 360  
In: 613.039368 609.869263 339.686951 165.397888 -0.648509 0.448876 0.009435 -0.011302  
Out: Crop  
Output resolution: 576 320  
Started mapping thread!  
Started optimization thread  
Started constraint search thread!  
Doing Random initialization!  
started image display thread!  
Done Random initialization!
```

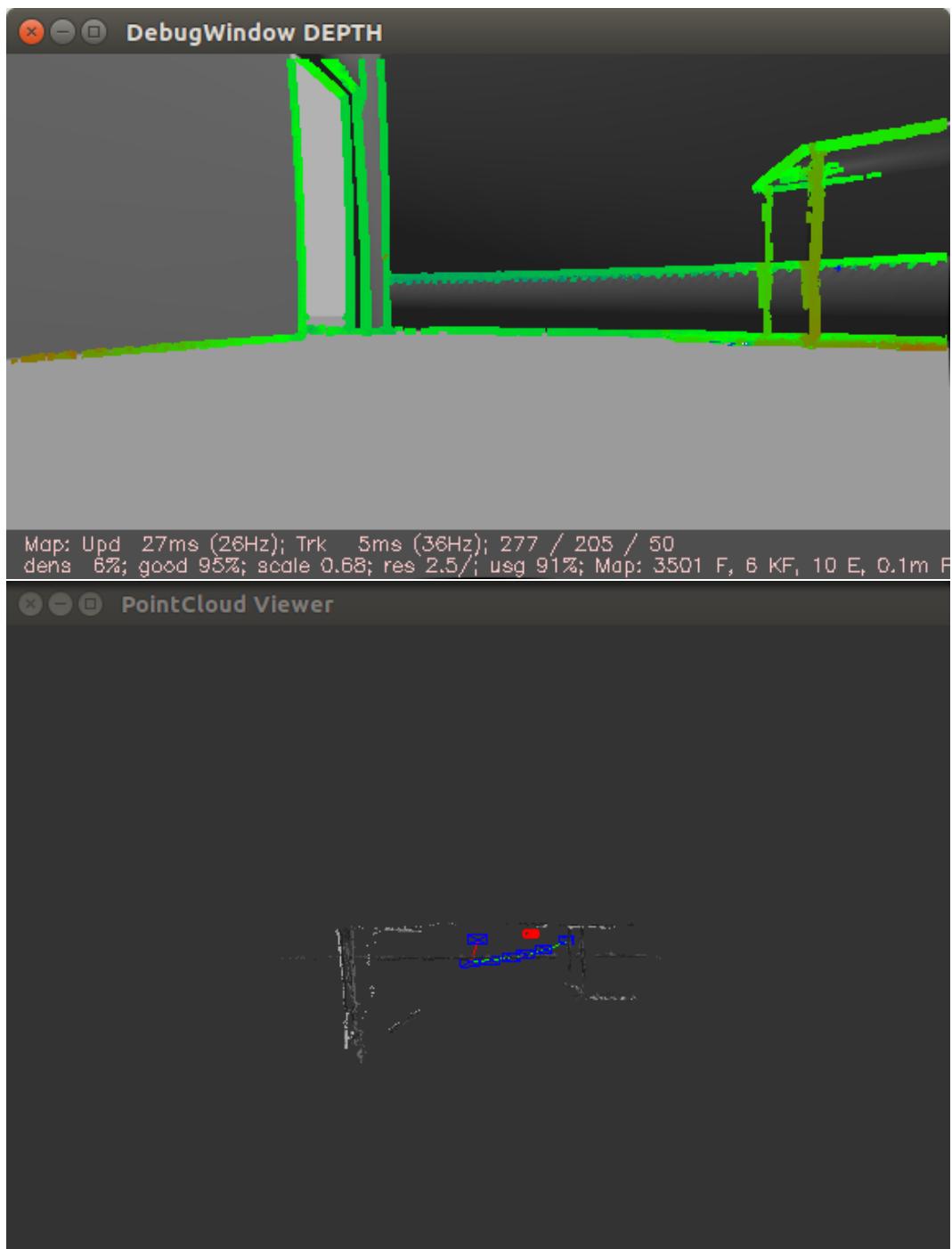
Gambar 6.55 eksekusi perintah untuk menjalankan slam

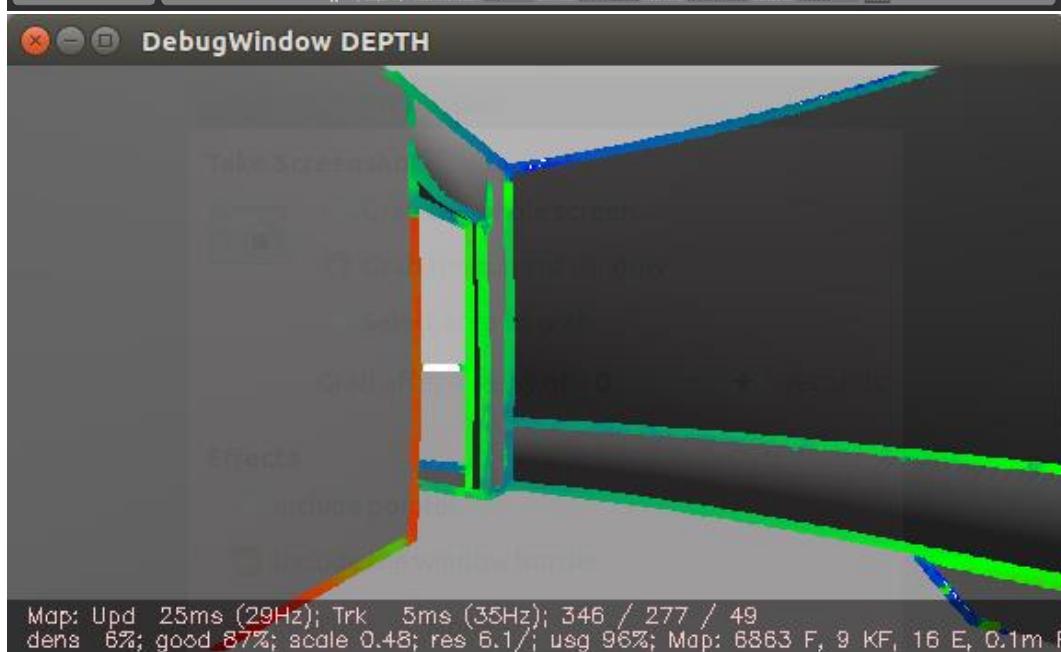
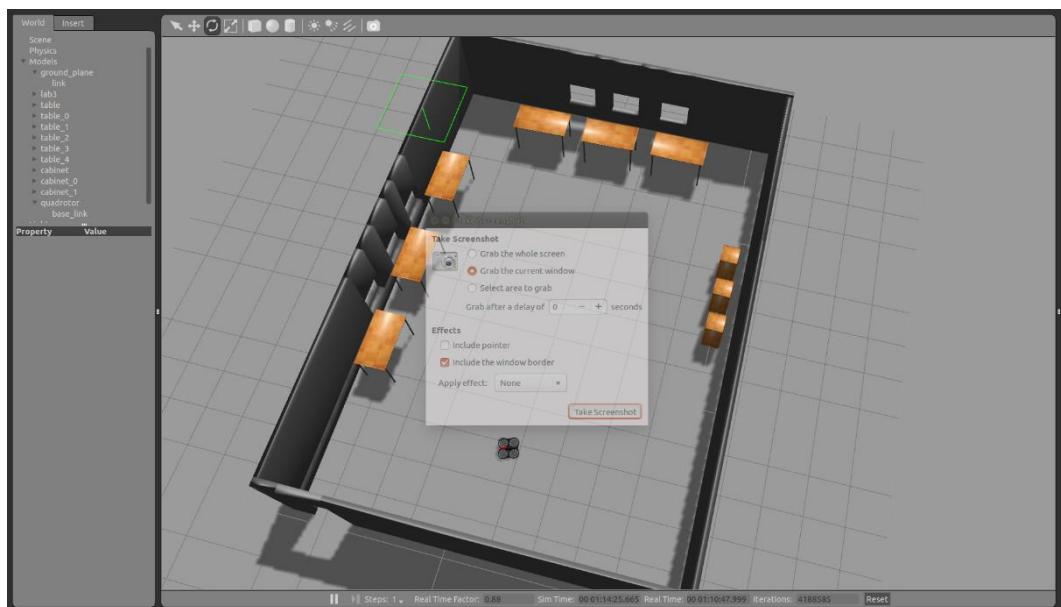


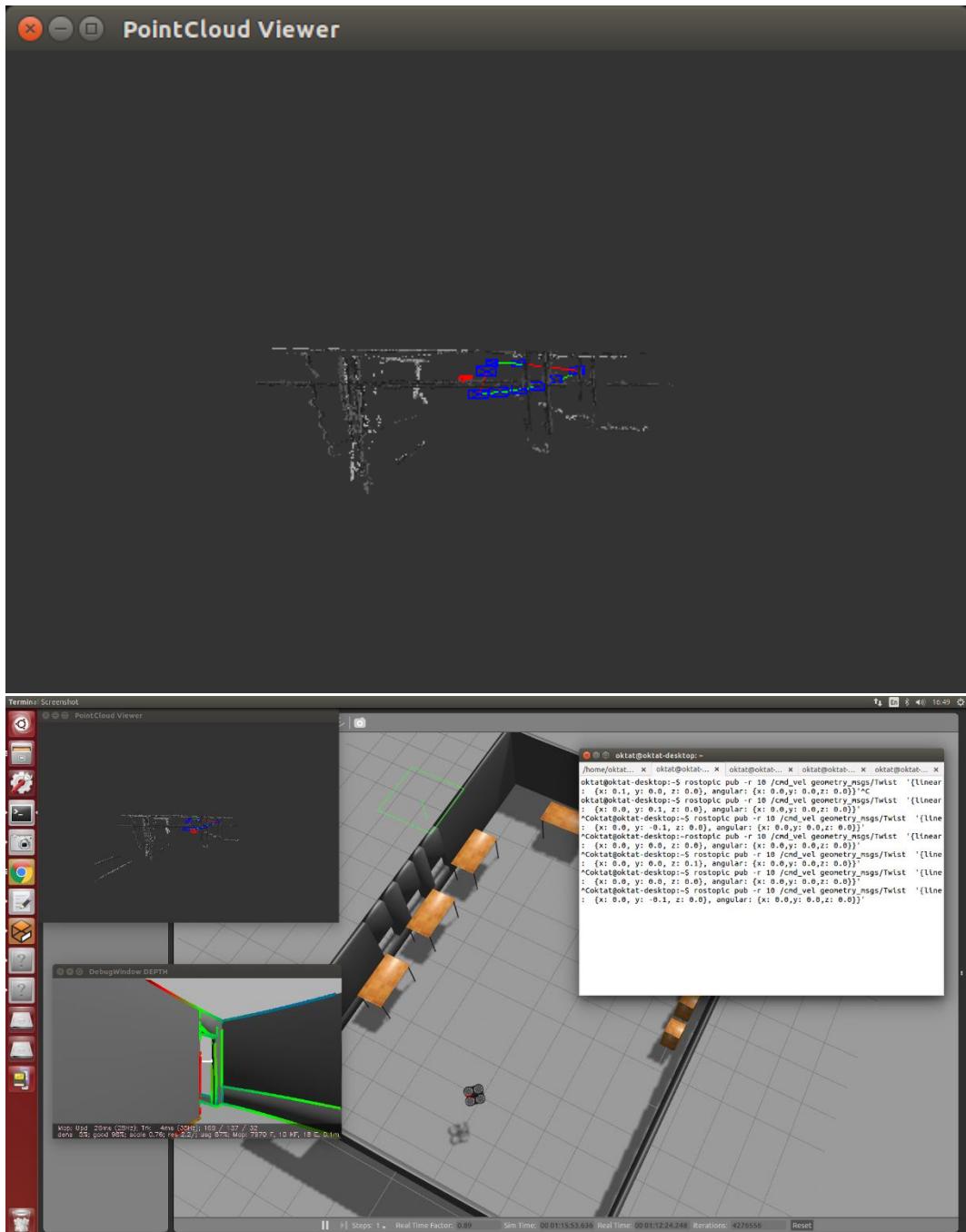
Gambar 6.56 hasil perintah untuk menjalankan slam

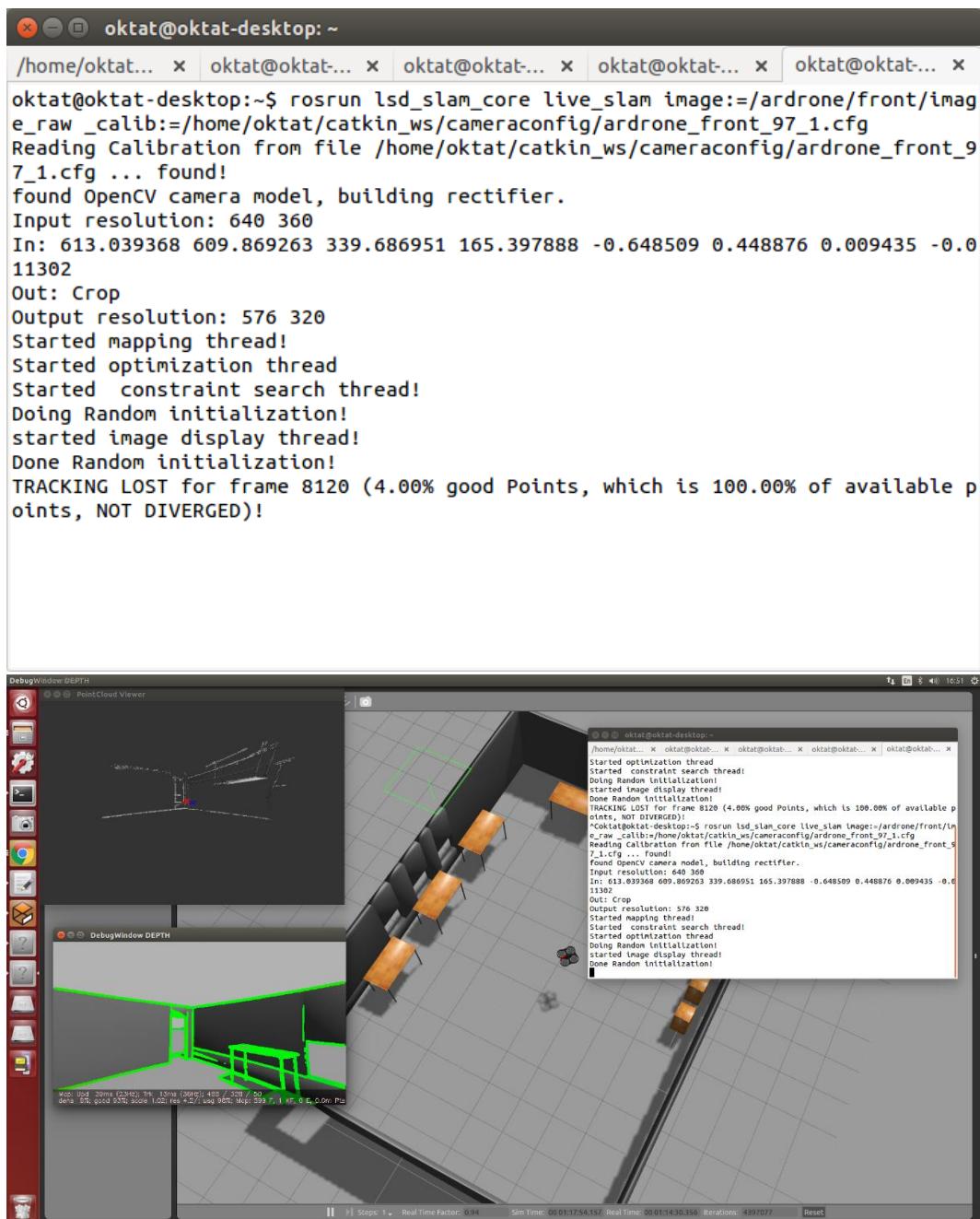
## 6.2 Hasil LSD-SLAM

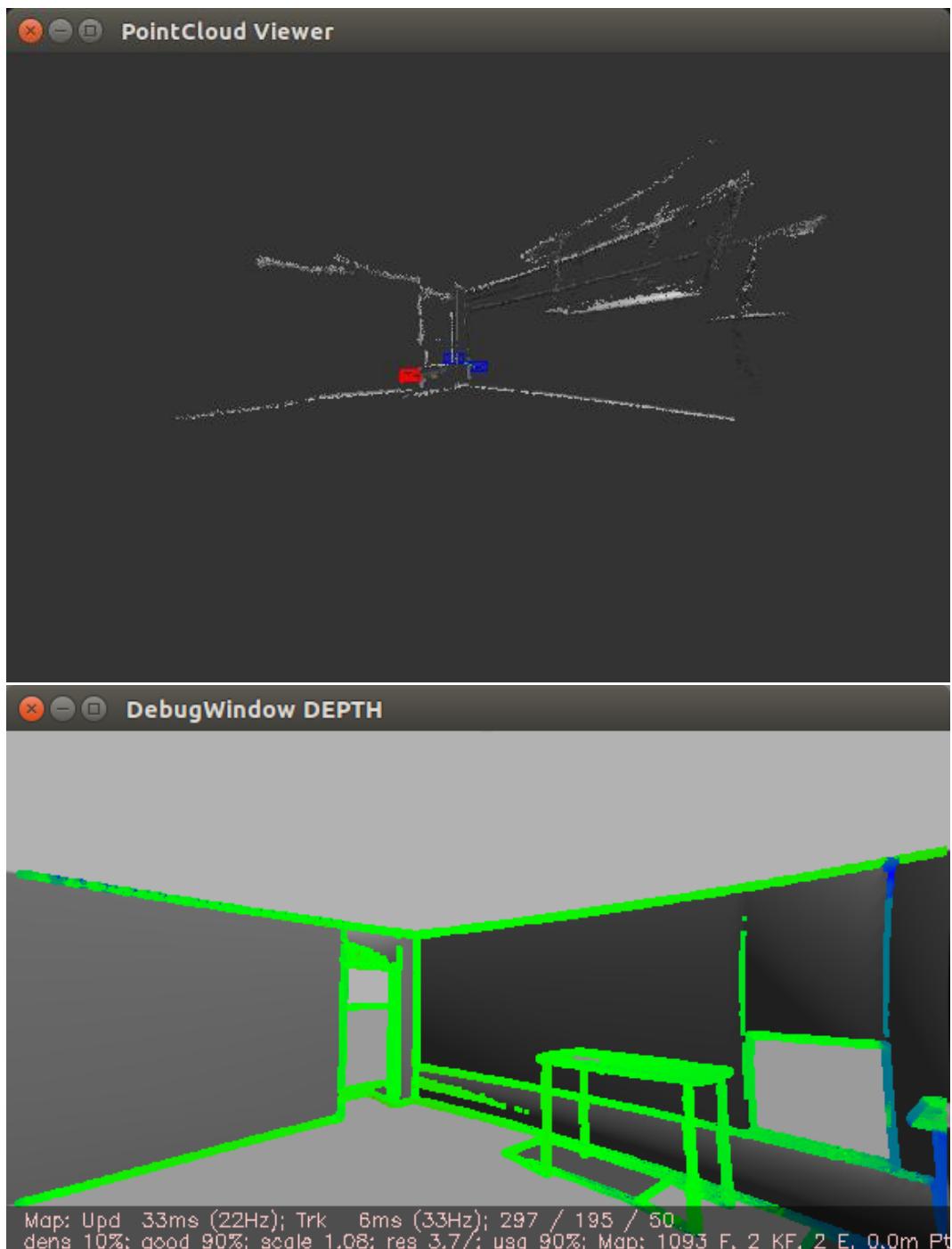


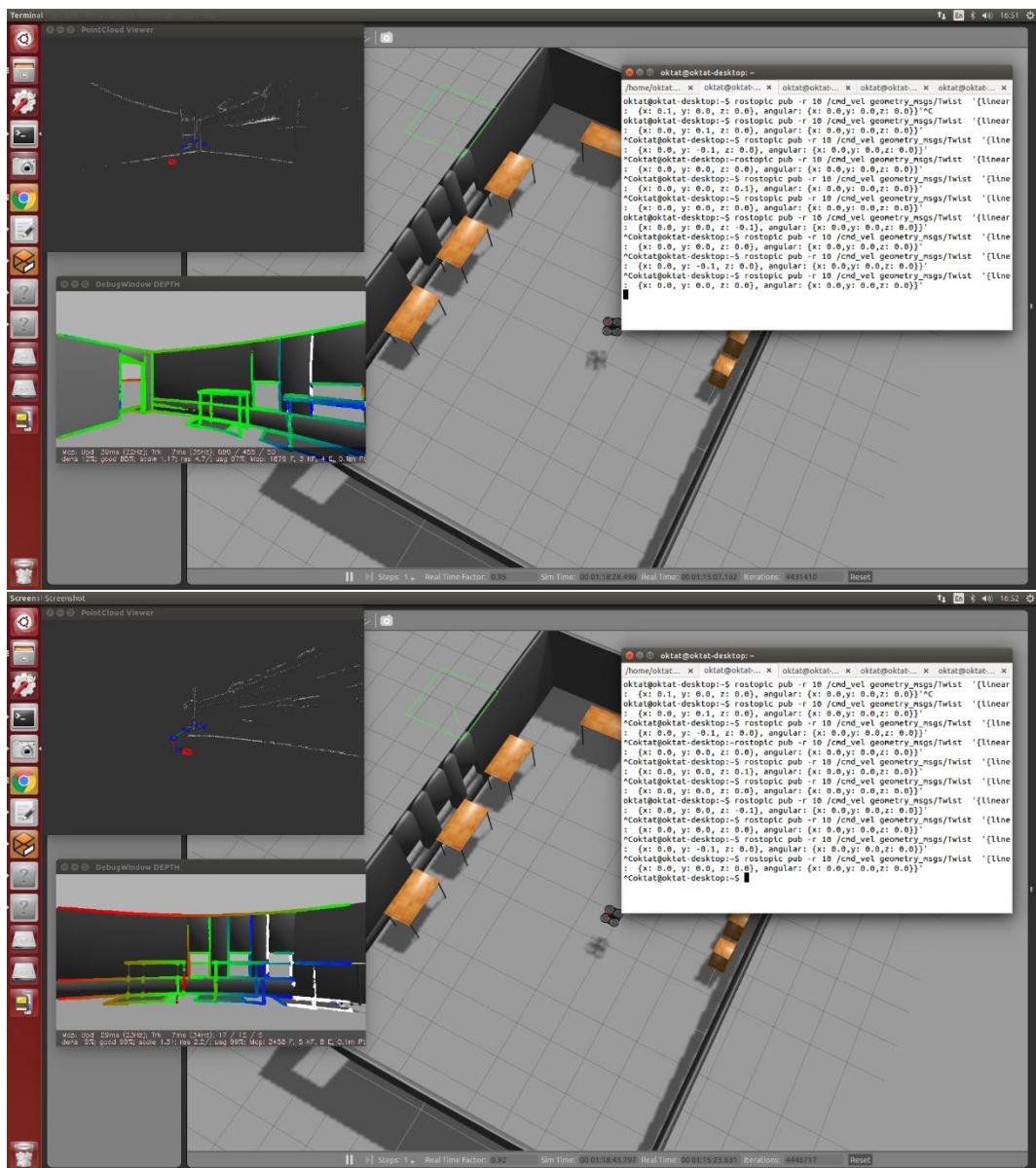


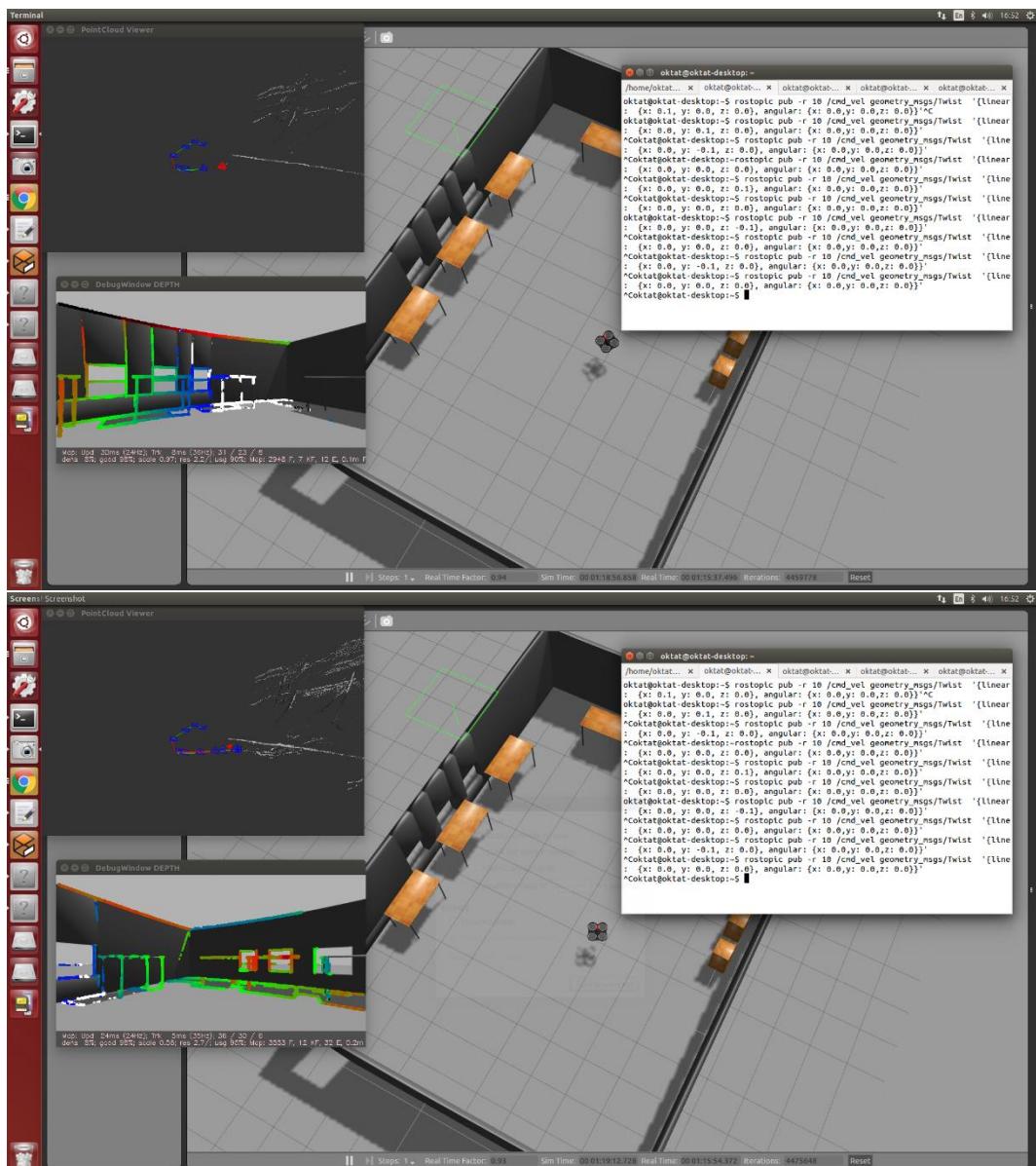


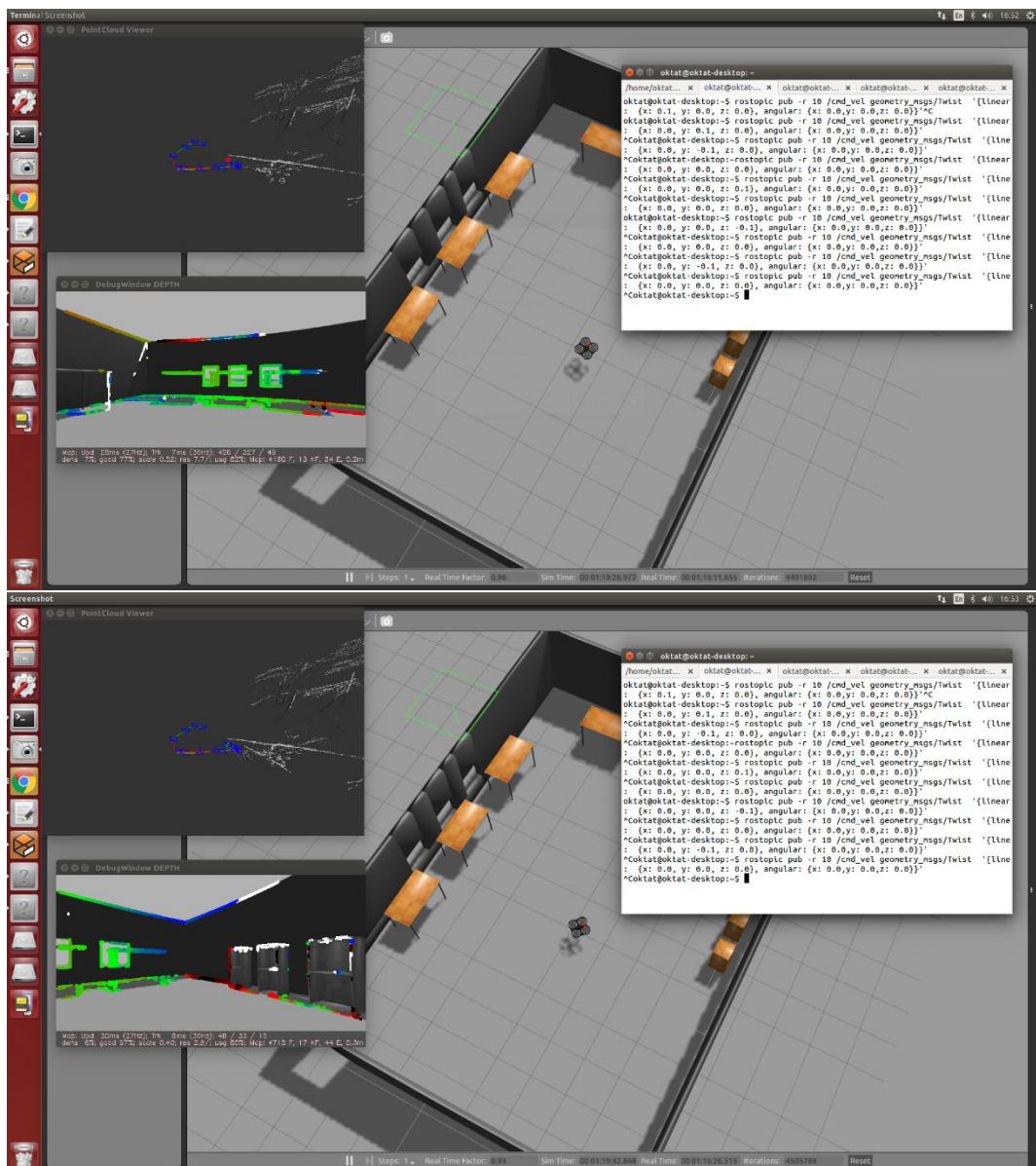


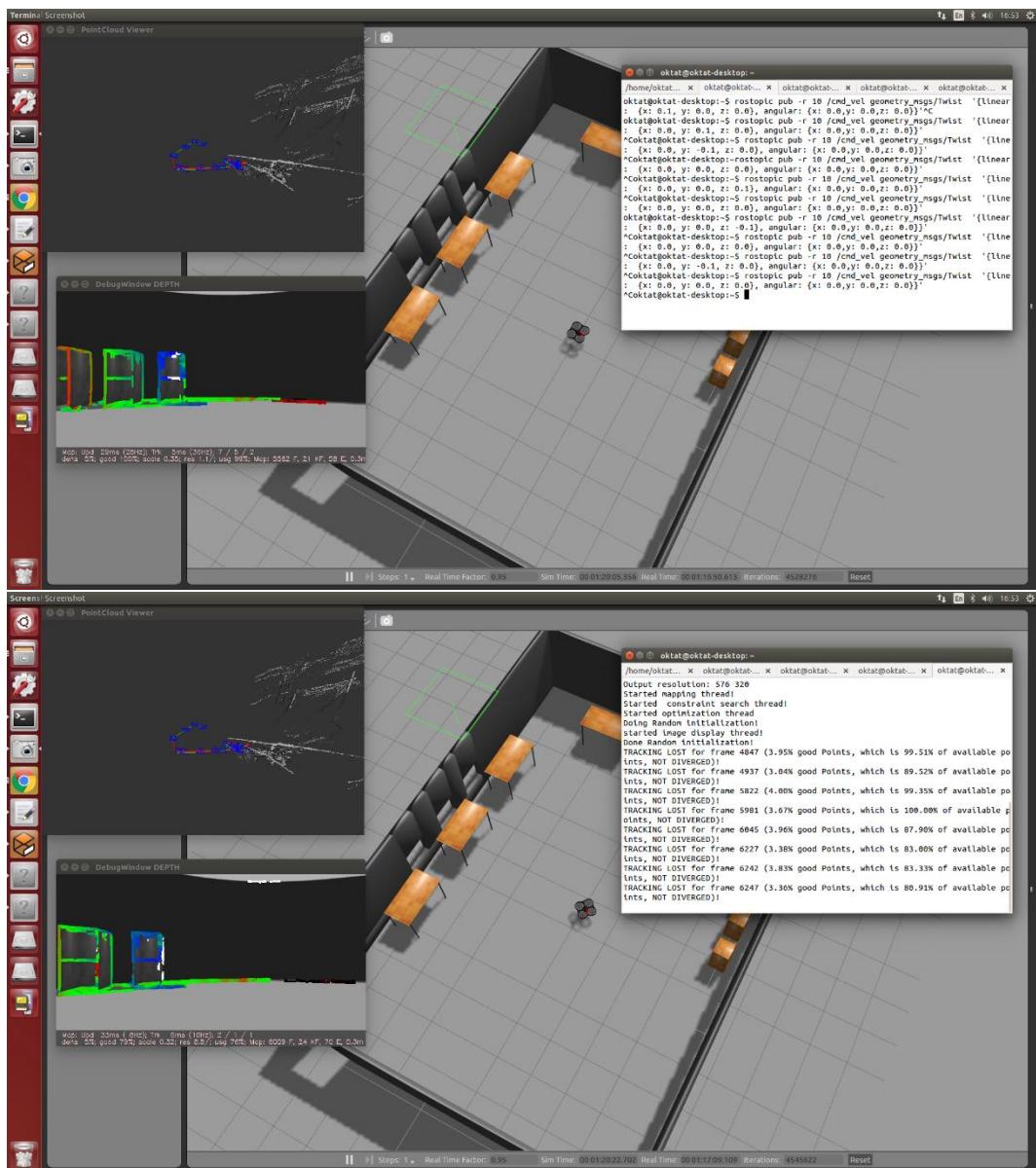


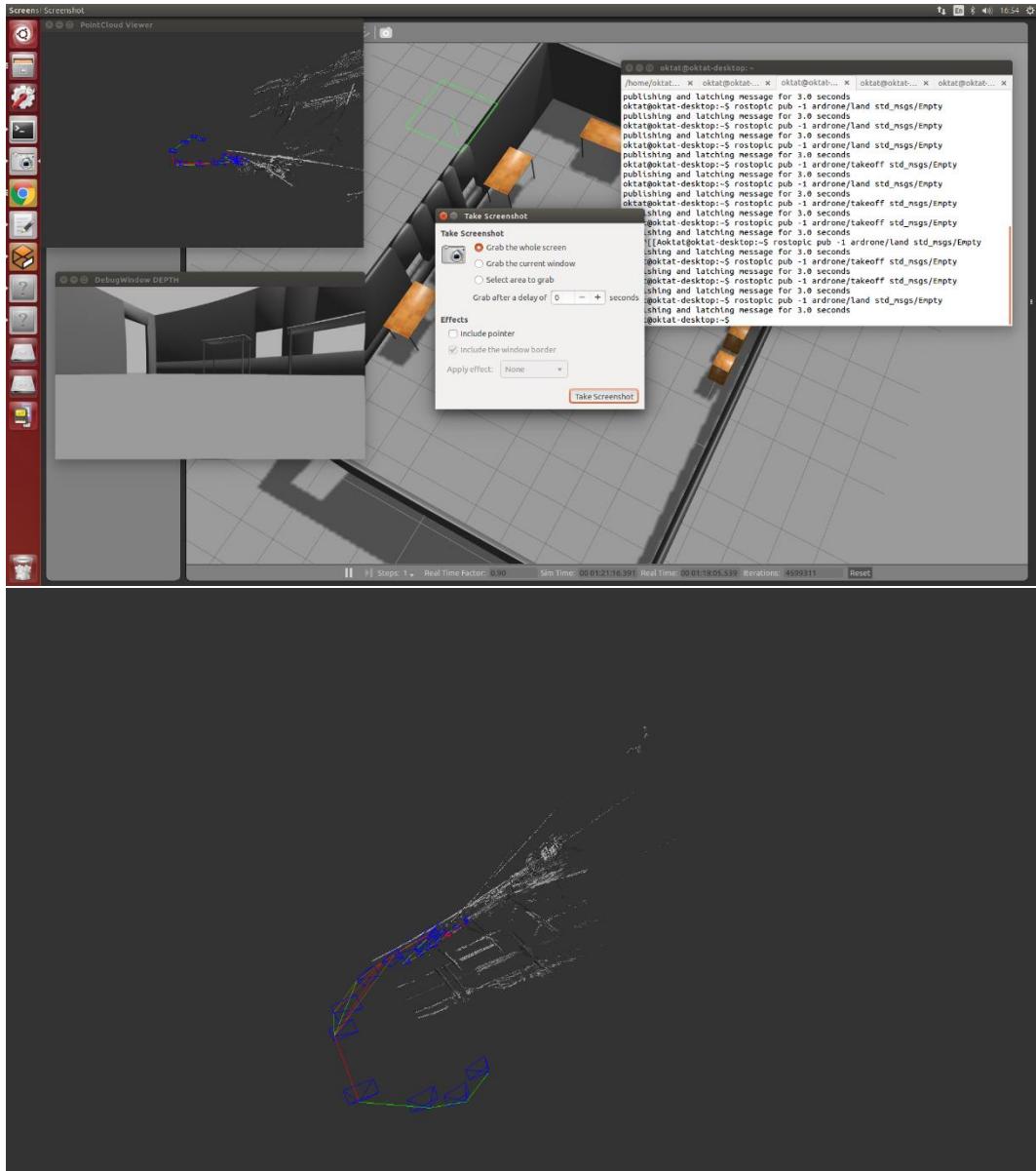


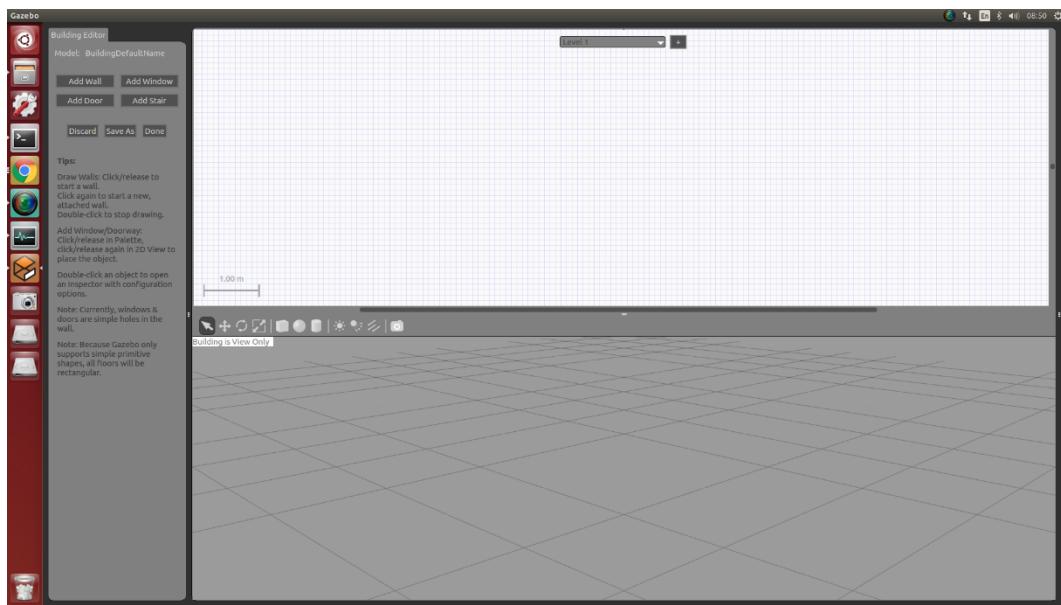








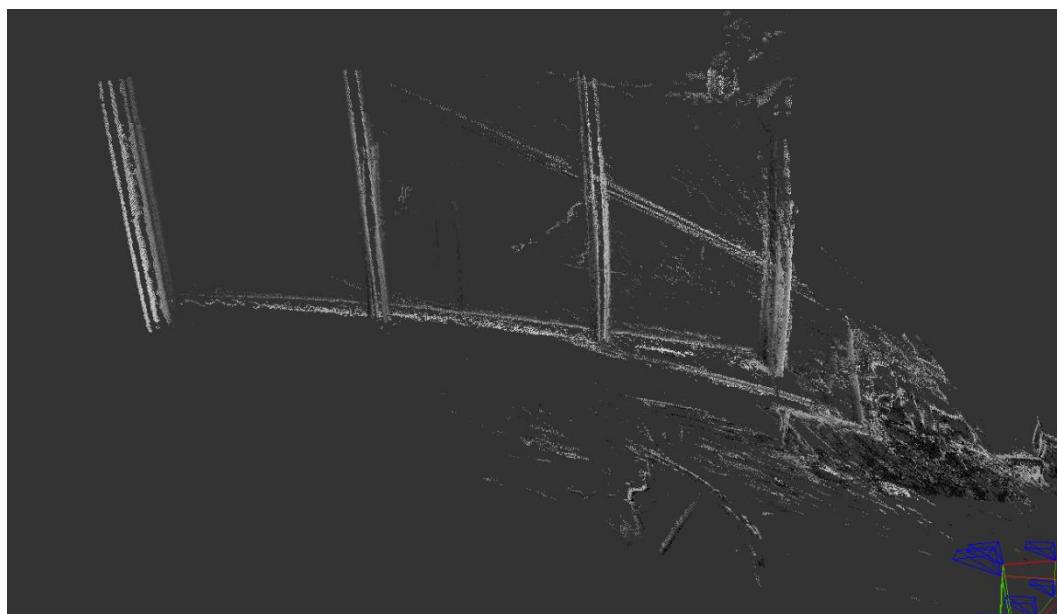




### 6.3 Hasil Pengujian LSD-SLAM



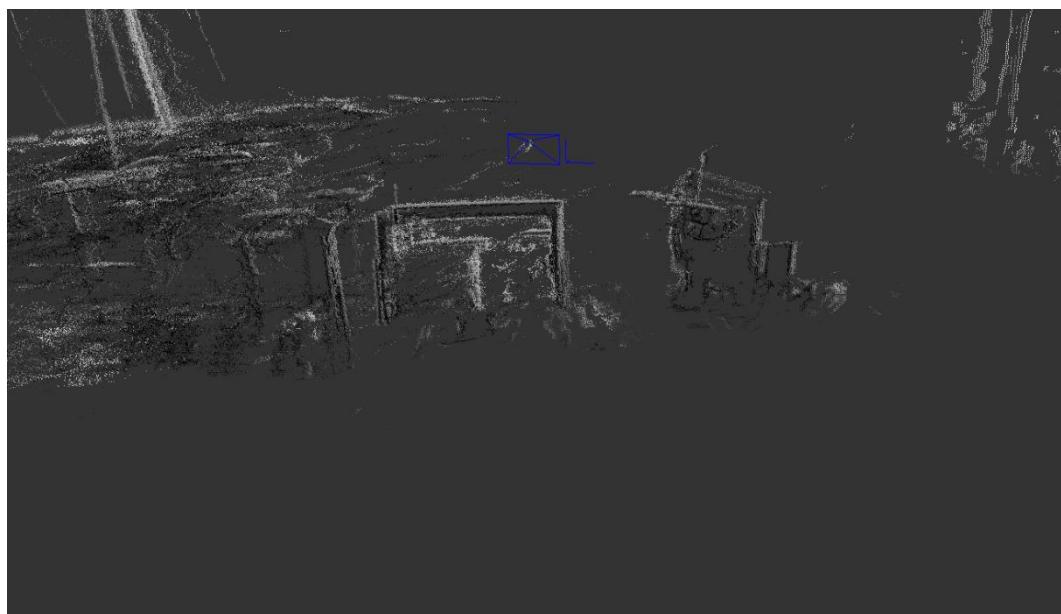
Gambar 6.57 Kondisi sebenarnya



Gambar 6.58 Hasil pengujian proses LSD-SLAM



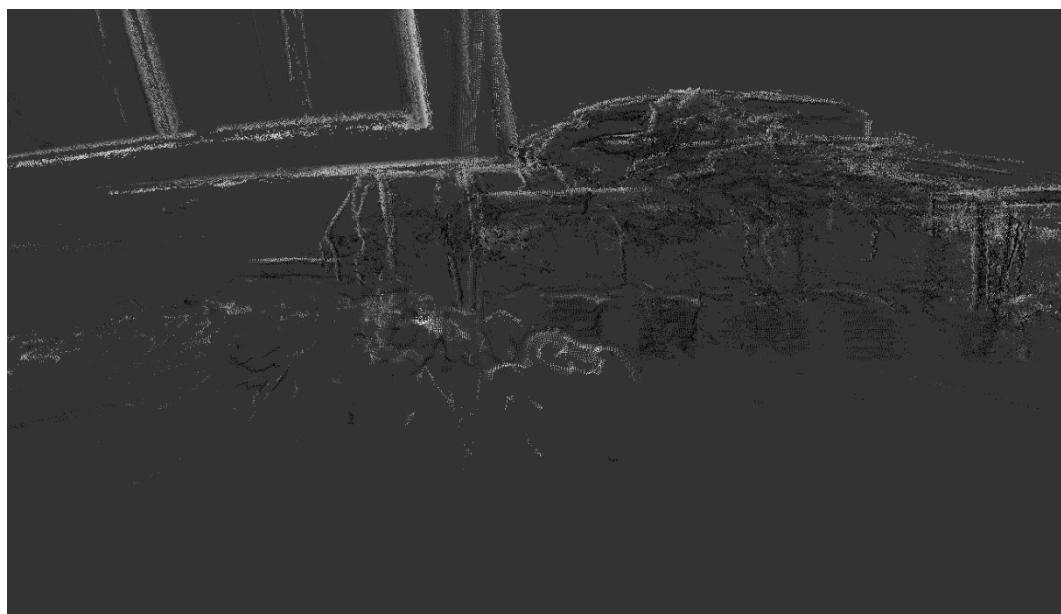
Gambar 6.59 Kondisi sebenarnya



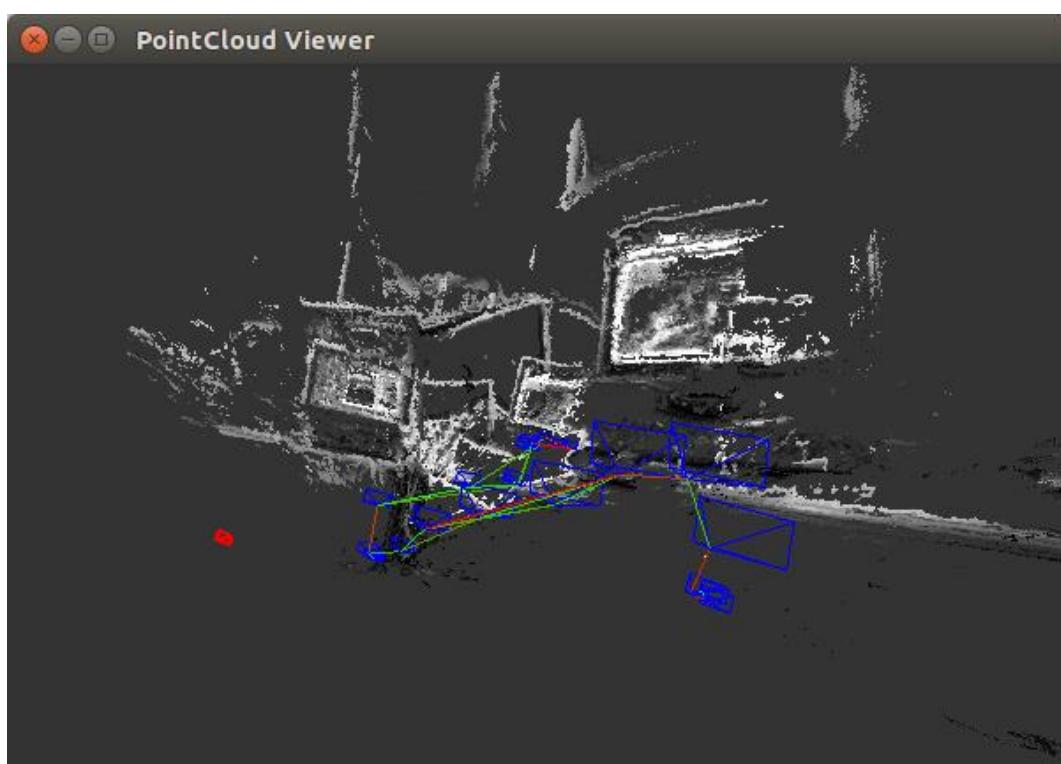
Gambar 6.60 Hasil pengujian proses LSD-SLAM



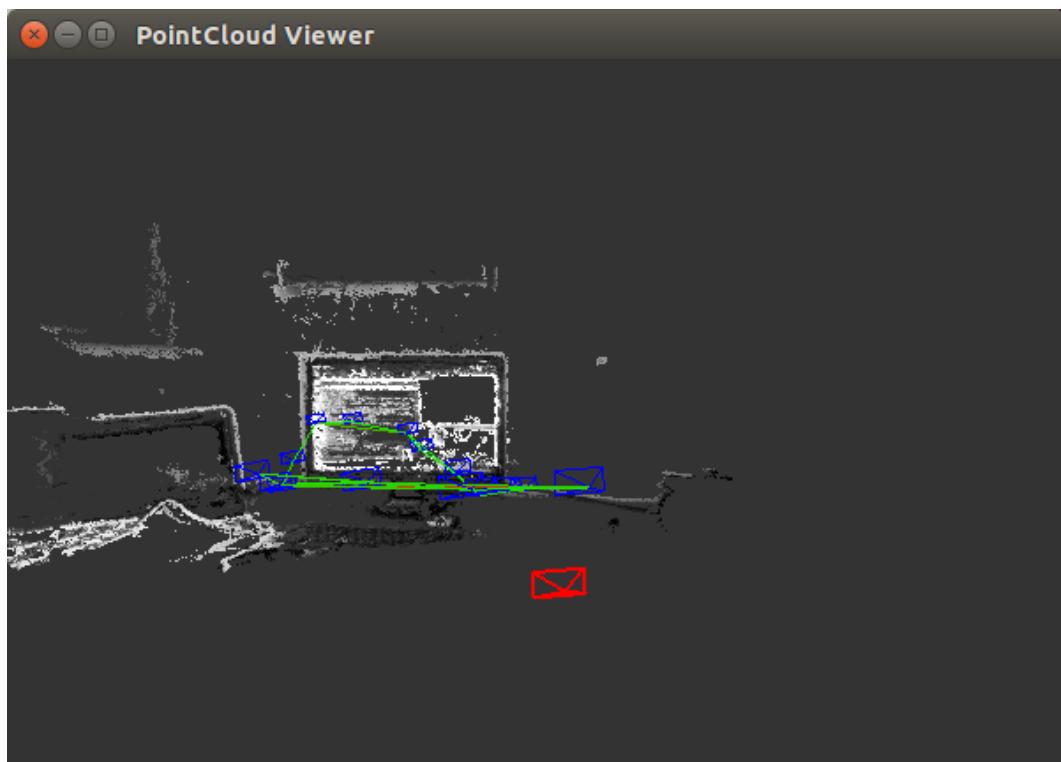
Gambar 6.61 kondisi sebenarnya



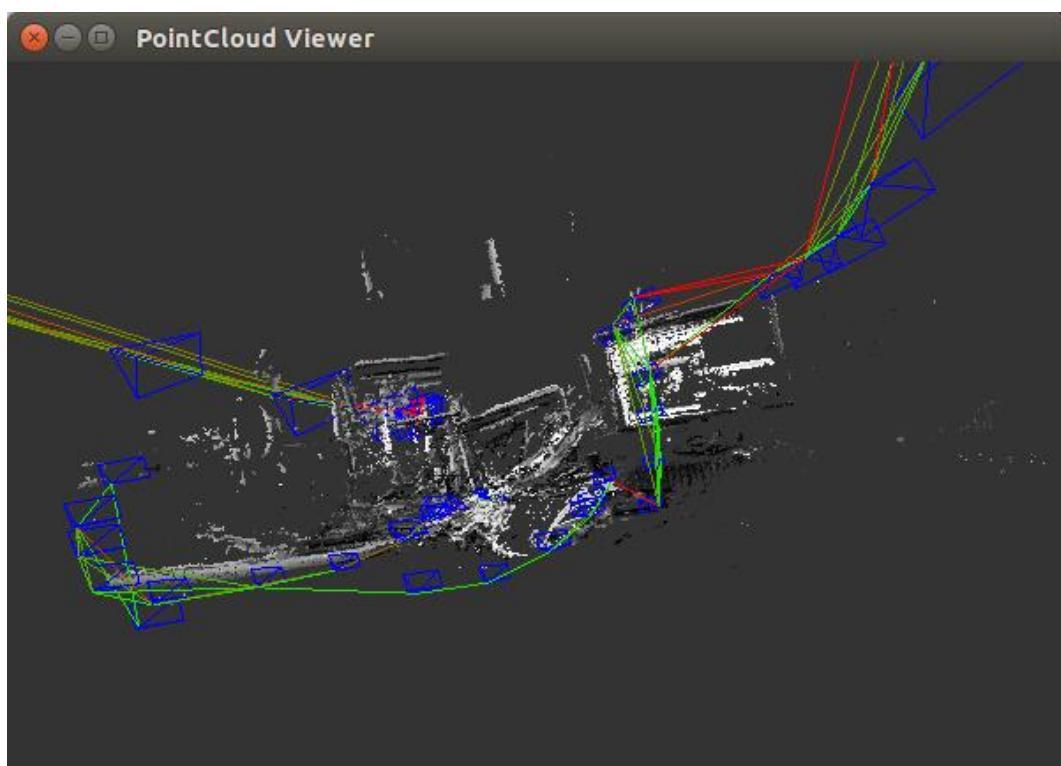
Gambar 6.62 Hasil pengujian proses LSD-SLAM



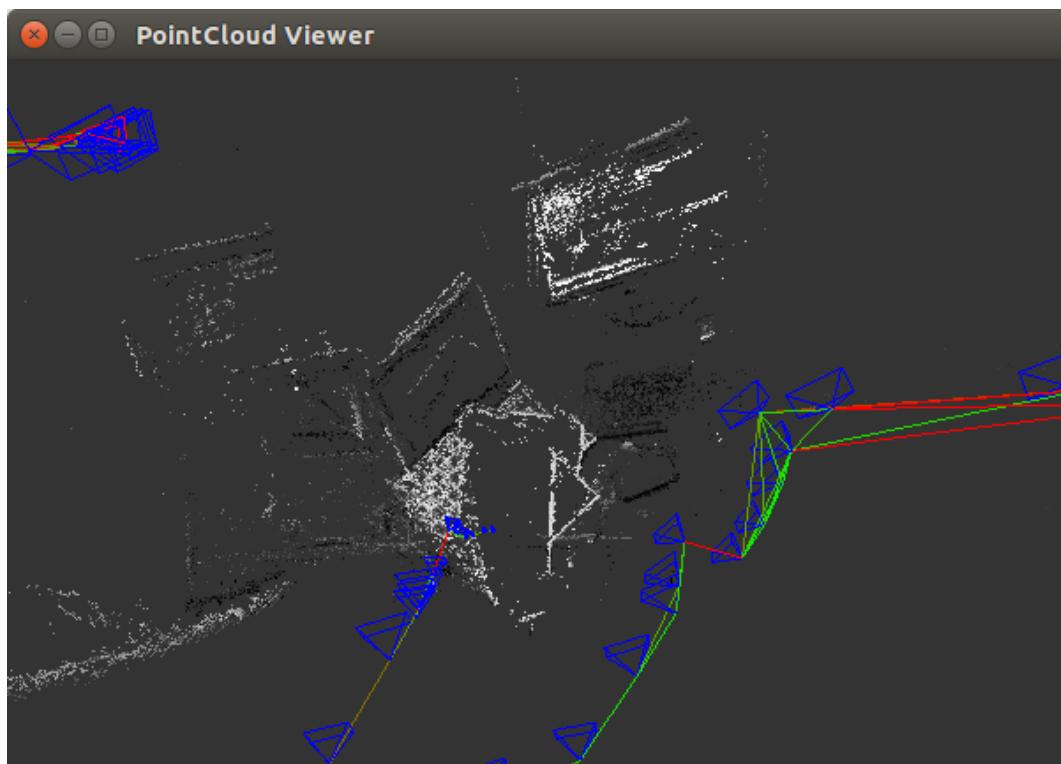
Gambar 6.63 Hasil pengujian proses LSD-SLAM



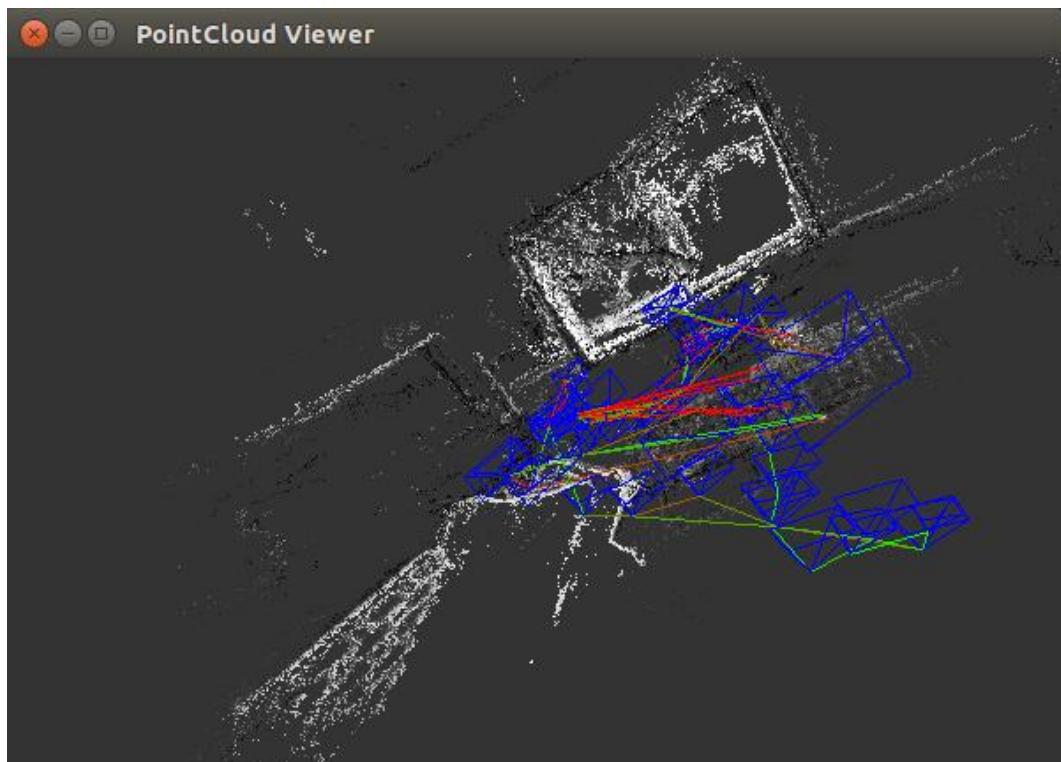
Gambar 6.64 Hasil pengujian proses LSD-SLAM



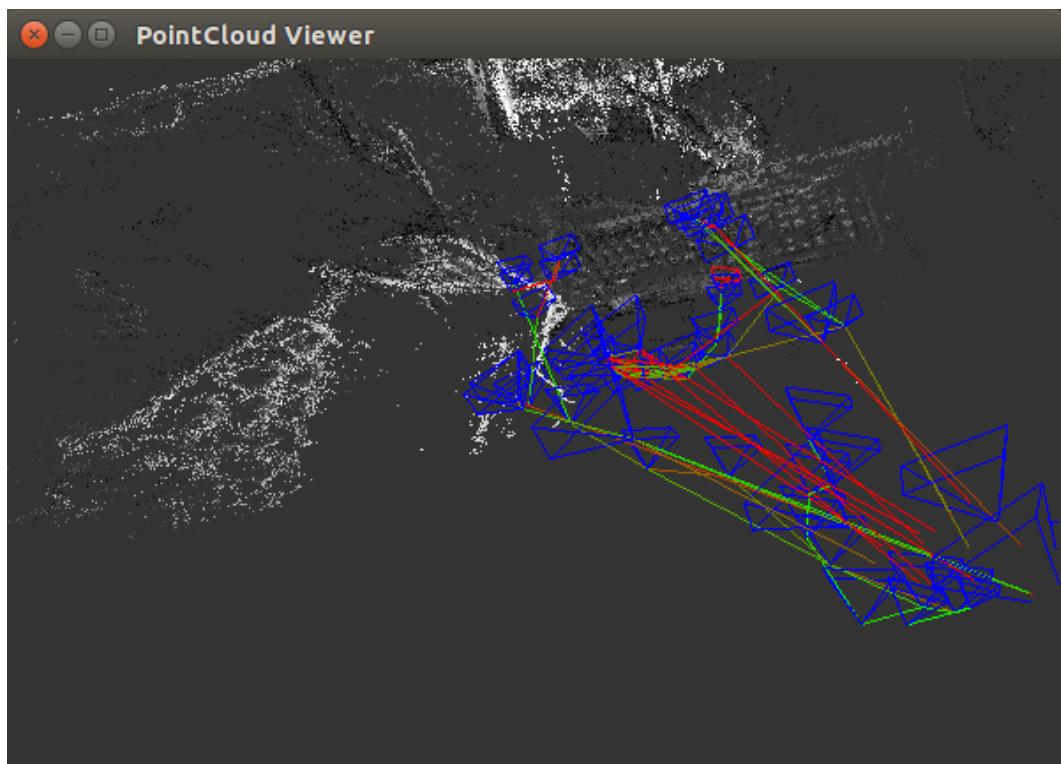
Gambar 6.65 Hasil pengujian proses LSD-SLAM



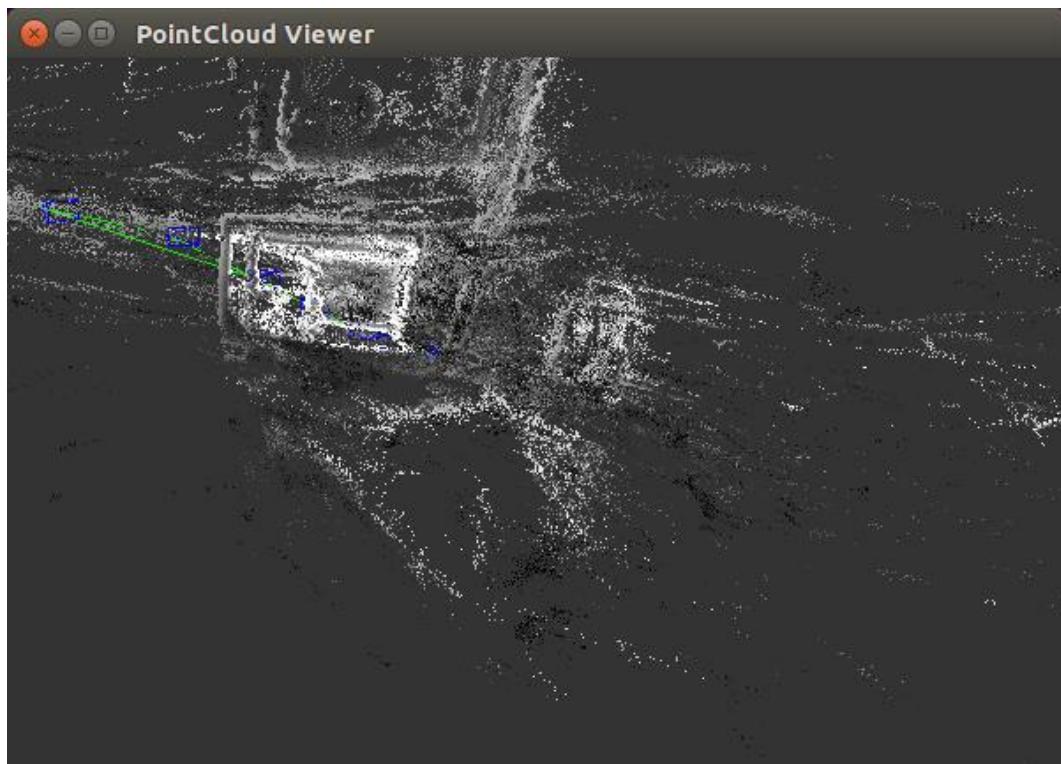
Gambar 6.66 Hasil pengujian proses LSD-SLAM



Gambar 6.67 Hasil pengujian proses LSD-SLAM



Gambar 6.68 Hasil pengujian proses LSD-SLAM



Gambar 6.69 Hasil pengujian proses LSD-SLAM

## DAFTAR PUSTAKA

- Bailey, T. & Durrant-Whyte, H., 2006. Simultaneous localization and mapping (SLAM): part II. Dalam: *IEEE Robotics & Automation Magazine*. 13. s.l.:IEEE, pp. 108-117.
- Buyval, A., Afanasyev, I. & Magid, E., 2017. *Comparative Analysis of ROS-based Monocular SLAM Methods for Indoor Navigation*. Nice, France, Ninth International Conference on Machine Vision.
- Durrant-Whyte, H. & Bailey, T., 2006. Simultaneous localization and mapping: part I. Dalam: *IEEE Robotics & Automation Magazine* 13. s.l.:IEEE, pp. 99-110.
- Engel, J. & Cremers, D., 2014. *LSD-SLAM: Large-Scale Direct Monocular SLAM*. [Online] Available at: <https://vision.in.tum.de/research/vslam/lسدslam> [Diakses 18 Januari 2019].
- Engel, J., Schops, T. & Cremers, D., 2014. *LSD-SLAM: Large-Scale Direct Monocular SLAM*. [Online] Available at: [https://vision.in.tum.de/\\_media/spezial/bib/engel14eccv.pdf](https://vision.in.tum.de/_media/spezial/bib/engel14eccv.pdf) [Diakses 18 Januari 2019].
- Hernandez-Martinez, E. et al., 2015. *Trajectory Tracking of a Quadcopter UAV with Optimal Translational Control*. Ciudad, IFAC.
- Kudan, 2016. *An Introduction to Simultaneous Localisation and Mapping*. [Online] Available at: <https://www.kudan.eu/kudan-news/an-introduction-to-slam/> [Diakses 18 Januari 2019].
- Maxwell, R., 2013. *Robotic Mapping: Simultaneous Localization and Mapping (SLAM)*. [Online] Available at: <https://www.gislounge.com/robotic-mapping-simultaneous-localization-and-mapping-slam/> [Diakses 18 Januari 2019].
- Mur-Artal, R. & D. Tardos, J., 2015. *Probabilistic Semi-Dense Mapping from Highly Accurate Feature-Based Monocular SLAM*, Spain: Instituto de Investigacion en Ingenieria de Aragon (I3A), Universidad de Zaragoza.
- Omega, D., 2017. *How GPS Drone Navigation Works*. [Online] Available at: <http://www.droneomega.com/gps-drone-navigation-works/> [Diakses 7 September 2017].
- Parrot, 2018. *PARROT AR.DRONE 2.0*. [Online] Available at: <https://www.parrot.com/global/drones/parrot-ardrone-20-elite-edition#parrot-ardrone-20-elite-edition-details> [Diakses 18 Januari 2019].
- Romero, L. E., Pozo, D. F. & Rosales, J. A., 2014. *Quadcopter stabilization by using PID controllers*. Quito, IEEE.

Yap, M., Bonardi, A., Larsen, P. & Howell, A., 2016. *LSD-SLAM About Large Scale Direct Monocular SLAM*. [Online] Available at: <https://www.doc.ic.ac.uk/~ab9515/lssd slam.html> [Diakses 18 Januari 2019].