IMPLEMENTASI *VISUAL SIMULTANEOUS LOCALISATION AND MAPPING (VSLAM)* MENGGUNAKAN *SINGLE CAMERA*

SKRIPSI

Disusun oleh:

BINASHIR ROFI'AH

1112530

PROGRAM STUDI TEKNIK ELEKTRO S-1

KONSENTRASI TEKNIK KOMPUTER

FAKULTAS TEKNOLOGI INDUSTRI

INSTITUT TEKNOLOGI NASIONAL MALANG

2015

[illegible]

LEMBAR PERSETUJUAN

IMPLEMENTASI *VISUAL SIMULTANEOUS LOCALISATION AND MAPPING (VSLAM)* MENGGUNAKAN *SINGLE CAMERA*

SKRIPSI

Disusun dan diajukan untuk melengkapi serta memenuhi persyaratan
guna mencapai gelar Sarjana Teknik

Disusun oleh :
BINASHIR ROFI'AH
NIM 1112530

Diperiksa dan Disetujui,

Dosen Pembimbing I,	Dosen Pembimbing II,
Dr. Eng. Aryuanto Soetedjo, ST, MT	**Bima Aulia Firmandani, ST**
NIP P. 1030800417	

Mengetahui,
Ketua Program Studi Teknik Elektro S-1,

M. Ibrahim Ashari, ST, MT
NIP P. 1030100358

PROGRAM STUDI TEKNIK ELEKTRO S-1
KONSENTRASI TEKNIK KOMPUTER
FAKULTAS TEKNOLOGI INDUSTRI
INSTITUT TEKNOLOGI NASIONAL MALANG
2015

PERKUMPULAN PENGELOLA PENDIDIKAN UMUM DAN TEKNOLOGI NASIONAL MALANG
INSTITUT TEKNOLOGI NASIONAL MALANG
FAKULTAS TEKNOLOGI INDUSTRI
FAKULTAS TEKNIK SIPIL DAN PERENCANAAN
PROGRAM PASCASARJANA MAGISTER TEKNIK
BNI (PERSERO) MALANG
BANK NIAGA MALANG
Kampus I : Jl. Bendungan Sigura-gura No. 2 Telp. (0341) 551431 (Hunting), Fax. (0341) 553015 Malang 65145
Kampus II : Jl. Raya Karanglo, Km 2 Telp. (0341) 417636 Fax. (0341) 417634 Malang

BERITA ACARA UJIAN SKRIPSI
FAKULTAS TEKNOLOGI INDUSTRI

NAMA : BINASHIR ROFI'AH
NIM : 1112530
JURUSAN : Teknik Elektro S-1
KONSENTRASI : Teknik Komputer
MASA BIMBINGAN : SEMESTER GANJIL 2014/2015
JUDUL SKRIPSI : **IMPLEMENTASI *VISUAL SIMULTANEOUS LOCALISATION AND MAPPING (VSLAM)* MENGGUNAKAN *SINGLE CAMERA***

Dipertahankan di Hadapan Majelis Penguji Skripsi Jenjang Strata Satu (S-1) pada:

Hari : Sabtu
Tanggal : 21 Februari 2015
Dengan Nilai : **82 (A)**

PANITIA UJIAN SKRIPSI

Ketua Majelis Penguji,

M. Ibrahim Ashari, ST, MT
NIP P. 1030100358

Sekretaris Majelis Penguji,

Dr. Eng. I Komang Somawirata, ST, MT
NIP P. 1030100361

ANGGOTA PENGUJI

Dosen Penguji I,

Irmalia Suryani Faradisa, ST, MT
NIP P. 1030100365

Dosen Penguji II,

Ir. Yusuf Ismail Nakhoda, MT
NIP Y. 1018800189

PERKUMPULAN PENGELOLA PENDIDIKAN UMUM DAN TEKNOLOGI NASIONAL MALANG
INSTITUT TEKNOLOGI NASIONAL MALANG
FAKULTAS TEKNOLOGI INDUSTRI
FAKULTAS TEKNIK SIPIL DAN PERENCANAAN
PROGRAM PASCASARJANA MAGISTER TEKNIK
PERSERO) MALANG
NIAGA MALANG
Kampus I : Jl. Bendungan Sigura-gura No. 2 Telp. (0341) 551431 (Hunting), Fax. (0341) 553015 Malang 65145
Kampus II : Jl. Raya Karanglo, Km 2 Telp. (0341) 417636 Fax. (0341) 417634 Malang

SURAT PERNYATAAN ORISINALITAS

Yang bertandatangan di bawah ini:

Nama	: Binashir Rofi'ah
NIM	: 1112530
Program Studi	: Teknik Elektro S-1
Konsentrasi	: Teknik Komputer

Dengan ini menyatakan bahwa Skripsi yang saya buat adalah hasil karya sendiri, bukan merupakan plagiasi/penjiplakan dari karya orang lain. Dalam Skripsi ini tidak memuat karya orang lain, kecuali dicantumkan sumbernya sesuai dengan ketentuan yang berlaku.

Demikian surat pernyataan ini saya buat untuk dipergunakan sebagaimana mestinya, dan apabila di kemudian hari ada pelanggaran atas surat pernyataan ini, saya bersedia menerima sangsinya.

Malang, 3 Maret 2015
Yang Membuat Pernyataan,

Binashir Rofi'ah
NIM 1112530

LEMBAR PERSEMBAHAN

"... Sesungguhnya sesudah kesulitan ada kemudahan. Maka apabila kamu telah selesai (urusan dunia) maka bersungguh-sungguhlah (dalam beribadah), dan hanya kepada Tuhanmulah kamu berharap."

(QS. Al Insyirah: 6-8)

Skripsi ini penulis persembahkan
untuk Allah, orang tua, adik, teman dan ITN.
Terima kasih selalu ada di sisi ini.

IMPLEMENTASI *VISUAL SIMULTANEOUS LOCALISATION AND MAPPING (VSLAM)* MENGGUNAKAN *SINGLE CAMERA*

Binashir Rofi'ah, NIM 1112530
Dosen Pembimbing: Dr. Eng. Aryuanto Soetedjo, ST, MT dan Bima Aulia F., ST

Konsentrasi Teknik Komputer, Program Studi Teknik Elektro S-1
Fakultas Teknologi Industri, Institut Teknologi Nasional Malang
Jl. Raya Karanglo Km. 2 Malang
e-mail: link.of.rofi@gmail.com

ABSTRAK

Dalam rangka membangun peta, robot harus mengetahui posisi (localisation). Untuk menentukan posisi, robot membutuhkan peta (mapping). Pada perkembangannya, localisation dan mapping harus dilakukan secara bersamaan untuk membentuk kecerdasan robot dalam bereksplorasi secara otomatis pada suatu lingkungan baru, konsep tersebut kemudian disebut sebagai Simultaneous Localisation and Mapping (SLAM). Sedangkan untuk konsep Visual SLAM (VSLAM) adalah pengembangan SLAM dengan sensor utama berupa kamera, dan konsep tersebut masih relatif baru serta belum diimplementasikan seutuhnya pada sistem embedded.Untuk memulai pengimplementasian di bidang tersebut, penulis merealisasikan dalam bentuk studi, pengujian, dan simulasi implementasi pada suatu robot DDMR. Robot yang telah dilengkapi kamera tersebut bertugas untuk capture gambar, yang selanjutnya akan diambil oleh komputer sebagai bahan membuat peta. Komunikasi yang digunakan adalah komunikasi WiFi dengan konsep protokol RSync dan Cron, memungkinkan komputer mengambil gambar dari robot dengan resolusi waktu millisecond. Dari hasil pengujian, diperoleh waktu optimal pada tiap pengiriman adalah 86 ms, sedangkan untuk mengolah gambar menjadi peta dibutuhkan waktu 200 ms. Gambar dari perspektif robot yang telah disimpan di komputer akan dijalankan sekuensial serta diolah untuk membuat peta. Hasil peta tersebut menandakan bahwa konsep VSLAM berhasil diimplementasikan.

Kata Kunci: *Raspberry Pi, Robot, Sensor Kamera, VSLAM*

ABSTRACT

In order to build a map, robot must know the position (localisation). To determine the position, that robot requires a map (mapping). In its development, localisation and mapping must be done simultaneously to make an automatically observable robot's intelligence at a new environment, that concept is then referred as Simultaneous Localisation and Mapping (SLAM). Visual SLAM (VSLAM) is development of SLAM with camera as main sensor, and the concept is still relatively new and not yet fully implemented in embedded systems. To begin the implementation at that field, the author realizes in study, test, and implement to a robot called DDMR. The robot has camera that capture images, which would then be taken by computer as a material to make a map. Communication through WiFi with RSync and Cron protocols allow the computer takes a picture from the robot with millisecond time resolution. From the test results, obtained an optimal delivery time for each transfer is 86 ms, whereas for image processing into the map takes 200 ms. The captured images that have been stored in computer will run sequentially and processed to create a map. Results of these maps indicate that the VSLAM concept has been succesfully implemented.

Keywords: *Camera Sensor, Raspberry Pi, Robot, VSLAM*

KATA PENGANTAR

Puji Syukur kehadirat Tuhan Yang Maha Kuasa atas berkat dan rahmat-Nya, sehingga penulis dapat menyelesaikan Laporan Skripsi yang berjudul "**IMPLEMENTASI *VISUAL SIMULTAEOUS LOCALISATION AND MAPPING (VSLAM)* MENGGUNAKAN *SINGLE CAMERA***" dengan baik.

Adapun maksud dan tujuan dari penulisan laporan ini merupakan salah satu syarat untuk dapat menyelesaikan studi dan mendapatkan gelar Sarjana Teknik Jurusan Teknik Elektro S-1, Konsentrasi Teknik Komputer ITN Malang.

Penulis menyadari tanpa adanya kemauan dan usaha serta bantuan dari berbagai pihak, maka laporan ini tidak dapat diselesaikan dengan baik. Oleh karena itu, penulis mengucapkan terima kasih kepada yang terhormat :

1. Dr. Ir. Lalu Mulyadi, MT selaku Rektor Institut Teknologi Nasional Malang.
2. Ir. Anang Subardi, MT selaku Dekan Fakultas Teknologi Industri Institut Teknologi Nasional Malang.
3. M. Ibrahim Ashari, ST, MT selaku Ketua Program Studi Teknik Elektro S-1 Institut Teknologi Nasional Malang.
4. Dr. Eng. Aryuanto Soetedjo, ST, MT selaku Dosen Pembimbing Satu Skripsi.
5. Bima Aulia Firmadani, ST selaku Dosen Pembimbing Dua Skripsi.
6. Sahabat-sahabat dan rekan-rekan yang tidak dapat disebutkan satu persatu, yang telah membantu baik dari segi teknis maupun dukungan moral dalam terselesaikanya skripsi ini.

Usaha telah penulis lakukan semaksimal mungkin, namun jika ada kekurangan dan kesalahan dalam penyusunan, penulis mohon saran dan kritik yang sifatnya membangun, begitu juga sangat diperlukan untuk menambah kesempurnaan laporan ini. Semoga dapat bermanfaat bagi rekan-rekan mahasiswa pada khususnya dan pembaca pada umumnya.

Malang, Maret 2015

Penulis

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

BAB I
PENDAHULUAN

1.1 Latar Belakang

Robot adalah benda bergerak yang dibuat untuk membantu pekerjaan manusia. Robot biasanya digunakan untuk tugas yang berat, berbahaya, pekerjaan yang berulang dan kotor. Paradigma penelitian tentang robot pun berkembang dari robot yang digerakkan manusia ke arah pergerakan robot otomatis. Robot otomatis memiliki banyak keunggulan yaitu efiesiensi kerja, waktu dan biaya. Sebuah robot otomatis yang mampu mengeksplorasi lingkungan dalam bekerja membutuhkan suatu peta dan letak/posisi.

Dalam rangka membangun peta, robot harus mengetahui posisi (*localisation*). Untuk menentukan posisi, robot membutuhkan peta (*mapping*). Pada perkembangannya, *localisation* dan *mapping* harus dilakukan secara bersamaan untuk membentuk kecerdasan robot dalam bereksplorasi secara otomatis pada suatu lingkungan baru, konsep tersebut kemudian disebut sebagai *Simultaneous Localisation and Mapping* (SLAM). Permasalahan SLAM yang diangkat adalah mungkin tidaknya sebuah robot bergerak secara otomatis dan secara bertahap menyusun sebuah peta sekaligus secara konsisten mendeteksi posisinya pada peta tersebut (Durrant-Whyte dan Bailey, 2006).

Oleh karena itu, penulis ingin membahas pengimplementasian konsep VSLAM pada *Differential Drive Mobile Robot* (DDMR) demi mengatasi masalah yang disebutkan diatas. Hingga saat ini, konsep SLAM yang menggunakan sensor selain kamera telah banyak digunakan, misalnya untuk memetakan sebuah ruangan (museum, sekolah, kantor), sistem pemandu ruangan, penunjuk jalan satuan pemadam kebakaran, dsb. Sedangkan untuk konsep Visual SLAM (VSLAM) masih relatif baru dan belum diimplementasikan dalam berbagai bidang.

Tantangan pada VSLAM adalah pengolahan gambar yang membutuhkan spesifikasi sumber daya tinggi. Namun seiring berjalannya waktu, banyak periset yang tertarik pada VSLAM karena kamera (yang digunakan sebagai sensor utama) adalah panca indera manusia untuk melihat lingkungan sekitarnya. Di sisi lain, tren perkembangan robot mengarah pada teknologi *vision* yaitu penggunaan kamera sebagai satu-satunya sensor yang memiliki banyak fungsi. Besar harapan

penulis agar nantinya hasil dari pengimplementasian konsep VSLAM ini dapat diterapkan lebih lanjut untuk bidang-bidang seperti yang telah disebutkan di atas.

1.2 Rumusan Masalah

Seperti masalah yang telah diutarakan pada latar belakang di atas, penulis merumusan masalah sebagai berikut:

1. Bagaimana cara mengimplementasikan konsep VSLAM pada sistem *embedded.*
1. Bagaimana robot dapat melakukan *visual* eksplorasi pada lingkungan tertentu.
2. Bagaimana cara mengelola hasil eksplorasi agar mampu dimonitoring jarak jauh.

1.3 Tujuan

Adapun perancangan dan pembuatan bertujuan untuk:

2. Mengujicoba implementasi konsep VSLAM pada sistem *embedded.*
3. Mengembangkan kemampuan robot yang tidak hanya bereksplorasi, tetapi juga mengirimkan hasil eksplorasinya.
4. Mengetahui sejauh mana manfaat yang diberikan apabila robot melakukan eksplorasi secara otomatis dengan hasil yang mampu dimonitoring.

1.4 Batasan Masalah

Agar perancangan dan pembuatan dapat sesuai dengan tujuan yang diharapkan dan tetap fokus pada konsep awal, maka diperlukan beberapa batasan-batasan diantaranya adalah :

1. Tidak membahas konstruksi mekanik dan perangkat keras robot secara detail.
2. Tidak membahas kecerdasan robot dalam melakukan navigasi.
3. Lingkungan yang dimaksud adalah lingkungan dengan cahaya cukup.
4. Peta yang dihasilkan berupa jalur 2 dimensi.
5. Tidak membahas skala peta.

1.5 Metodologi

Untuk memulai pengimplementasian di bidang tersebut, penulis merumuskan dalam bentuk studi, pengujian, dan simulasi implementasi dengan memberikan beberapa metodologi yang dapat dilakukan, meliputi:

1. Kajian Literatur

   Pengumpulan data dan informasi yang dilakukan dengan mencari bahan-bahan kepustakaan dan referensi dari berbagai sumber sebagai landasan teori yang berhubungan dengan permasalahan. Antara lain:

   a. Teori tentang SLAM, berupa cara kalibrasi dan teknik deteksi *feature* gambar
   b. Dasar pemrograman pada Raspberry Pi dan komputer.
   c. Dasar pengaksesan kamera RaspiCam.
   d. Akses *library* yang digunakan untuk mendukung program.
   e. Dasar komunikasi antara Raspberry Pi dengan komputer.

2. Diskusi

   Dilakukan dengan pembimbing dan dosen keahlian lainnya.

3. Lokasi

   Proses perancangan, pembuatan hingga pengujian dilakukan di laboratorium Pemrograman Komputer dan Multimedia teknik elektro S-1 ITN Malang.

4. Perancangan Sistem

   Proses perancangan atau desain yang akan diterapkan pada sistem, seperti: *flowchart* program, perancangan *toolchain* untuk membuat program, perancangan kalibrasi kamera, perancangan deteksi fitur gambar, perancangan komunikasi robot dengan komputer.

5. Pembuatan Sistem

   Proses implementasi dan pengujian unit dari hasil perencanaan sistem, berupa: pembuatan dan pengujian pengambilan gambar, pembuatan dan pengujian kalibrasi kamera, pembuatan dan pengujian transfer gambar, pembuatan dan pengujian olah gambar.

6. Pengujian Sistem

   Proses ujicoba keseluruhan sistem untuk mengetahui adanya kesalahan, agar sistem sesuai dengan konsep yang telah dirancang sebelumnya.

7. Pelaporan hasil pengujian dan kesimpulan.

1.6 Sistematika Penulisan

Guna mempermudah dalam mempelajari dan memahami pembahasan penulisan skripsi ini, penulis menggunakan sistematika sebagai berikut:

1. **BAB 1 PENDAHULUAN**

   Pada bab ini akan dibahas secara singkat mengenai latar belakang, tujuan dan manfaat, serta batasan masalah, dan sistematika penulisan.

2. **BAB II TINJAUAN PUSTAKA**

   Pada bab ini diuraikan secara singkat mengenai teori pustaka-pustaka serta penelitian yang terkait.

3. **BAB III PERANCANGAN DAN ANALISA**

   Pada bab ini akan dijelaskan secara umum mengenai desain implementasi konsep, prinsip kerja, serta analisa sistem pengambilan gambar, analisa kalibrasi kamera, analisa pengiriman gambar, analisa pemrosesan gambar, dan analisa sistem keseluruhan.

4. **BAB IV PEMBUATAN DAN PENGUJIAN**

   Pada bab ini akan dibahas langkah-langkah pembuatan sistem serta pengujian terhadap sistem tersebut.

5. **BAB V PENUTUP**

   Pada bab ini berisi tentang semua kesimpulan yang berhubungan dengan hasil skripsi yang telah dicapai, dan saran yang digunakan sebagai pertimbangan dalam pengembangan selanjutnya.

BAB II
DASAR TEORI

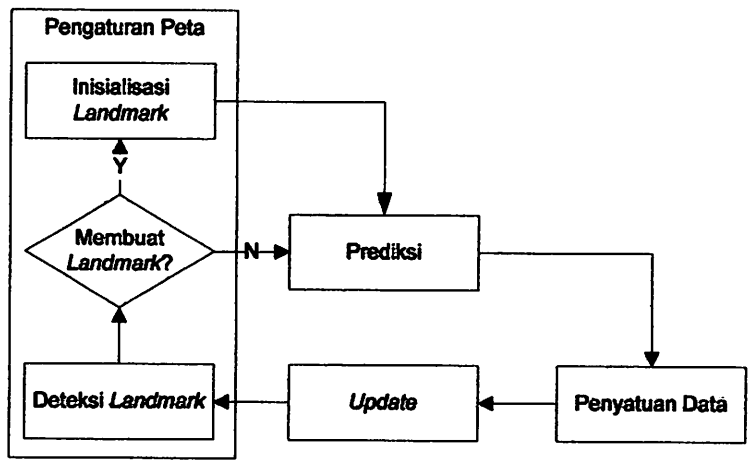
2.1 *Simultaneous Localisation and Mapping* (SLAM)

Konsep pergerakan otomatis robot *mobile* meliputi banyak bidang pengetahuan, metode, dan algoritma yang dirancang dengan tujuan untuk kontrol lintasan, penghindar rintangan, lokalisasi, membangun peta, dan sebagainya. Keberhasilan perencanaan dan navigasi jalur robot tergantung pada ketersediaan kedua estimasi yang cukup handal yaitu lokasi kendaraan dan representasi akurat dari area navigasi. Kemudian muncullah masalah *Simultaneous Localisation and Mapping*.

SLAM bukanlah merupakan sebuah algoritma tunggal, melainkan sebuah konsep utuh yang di dalamnya memuat banyak algoritma yang bekerja secara bersama-sama *(hybrid)* pada kondisi-kondisi tertentu (Riisgaard, Søren).

SLAM pertama kali dikembangkan oleh Hugh Durrant-Whyte dan John J. Leonard (dalam paper *Mobile Robot Localisation by Tracking Geometric Beacons, 1991*) berdasarkan penelitian yang dikerjakan oleh Smith, Self, dan Cheeseman (dalam paper Estimating Uncertain Spatial Relationships in Robotics, 1990). Durran-Whyte dan Leonard awalnya menamai istilah tersebut dengan SMAL, tetapi akhirnya diubah menjadi SLAM.

Proses lokalisasi dan pemetaan SLAM terdiri dari beberapa tahap (lihat gambar 2.1). SLAM dapat diterapkan pada pergerakan 2D maupun 3D, juga dapat bekerja pada lingkungan *indoor* maupun *outdoor*.



Gambar 2.1 Tahapan SLAM

2.1.1 Komponen Istilah pada SLAM

Istilah "*featur*/fitur" digunakan untuk merujuk ke titik tertentu di lingkungan yang dapat dideteksi oleh *mobile* robot. Dalam praktiknya, fitur adalah pola yang tersebar pada lingkungan robot, bisa berupa sudut yang secara natural dipilih oleh robot. Memililh, megidentifikasi, dan membedakan fitur, bukanlah persoalan sepele, namun hal itu berada di luar lingkup laporan ini. Akan diasumsikan bahwa fitur tersebut dianggap mampu dibedakan dan diidetifikasi ketika robot menscan lokasi yang sesuai.

Istilah "*landmark*" adalah fitur yang dapat dengan mudah kembali diamati dan dibedakan dari lingkungan hidup. *Landmark* harus mampu diamati kembali atau dideteksi dari dari posisi dan sudut yang berbeda. *Landmark* harus cukup unik sehingga mudah diidentifikasi pada satu waktu tanpa harus terganggu dengan hal lain. Contohnya ketika mengamati kembali dua *landmark* pada satu titik maka dengan mudah menentukan landmark mana yang telah terdeteksi sebelumnya. *Landmark* seharusnya benda mati yang diam di tempat (*stationary*). *Landmark* digunakan oleh robot untuk mencari tahu di mana letak robot (untuk melokalisasi diri sendiri).

Istilah "*pose*" atau "robot *pose*" digunakan secara sinonim untuk merujuk pada bagaimana robot diposisikan, hal ini termasuk variabel-variabel yang berkaitan dengan robot itu sendiri. Contoh sederhana adalah sebuah vektor dengan koordinat x dan y dengan orientasi theta. Pada kenyataannya *pose* robot merupakan hal kompleks yang mencakup pitch, roll, dan lain-lain.

Istilah "*state*" mengacu pada gabungan antara pose terbaru dan perkiraan peta yang dilakukan oleh robot.

Istilah "peta" akan selalu merujuk pada suatu vektor perkiraan lokasi fitur. Peta berbasis *landmark* (*landmark-based map*) adalah peta yang dibentuk dari landmark dengan banyak fitur pada suatu lingkungan. *Landmark* dapat berupa sudut, pojok, segmen garis, atau suatu titik. Landmark tersebut diasumsikan jarang dan tidak ambigu.

2.2 Visual SLAM

Salah satu jenis sensor yang dapat digunakan dalam algoritma SLAM adalah kamera; yang kemudian biasa disebut Visual SLAM dan telah menjadi

salah satu topik hangat robotika dan penelitian visi komputer sejak satu dekade terakhir. Ada dua alasan utama untuk menggunakan kamera sebagai sensor untuk SLAM:

1. Visual memiliki daya tarik intuitif yang besar, karena pandangan adalah salah satu indra hewan dan manusia yang digunakan untuk menyesuaikan diri;
2. kamera bersifat non-invasif, berukuran relatif kecil, murah dan ada di mana-mana, berbeda dengan peralatan seperti sensor laser yang yang lebih mahal, lebih besar dan membutuhkan lebih banyak daya untuk bekerja.

2.2.1 Visual MonoSLAM

Visual MonoSLAM akan membuat peta 3 dimensi yang konsisten dari lingkungan dengan menggunakan gerakan kamera tunggal. Lingkungan akan diwakili oleh titik-titik 3D. Informasi mendalam hanya dapat dikumpulkan jika titik yang sama diamati dari setidaknya dua *pose* dikenal. Oleh karena itu *pose* kamera perlu diperkirakan juga. Perkiraan tersebut dilakukan melalui EKF, dengan asumsi lingkungan yang statis. Oleh karena itu perubahan gambar yang diterima akan memberikan kontribusi terhadap gerakan kamera, hal itu tergantung pada jumlah gerakan dan kedalaman titik paralaks yang akan mulai berbeda. Dengan mengulangi pengamatan poin yang sama dan membandingkan lokasi proyeksi dengan lokasi diharapkan akan memperbaiki estimasi pose kamera dan titik estimasi lokasi. Metode yang mendasari hal itu adalah triangulasi (triangulation), proses triangulasi tersebut dapat dilakukan dengan Extended Kalman Filter (EKF). EKF adalah EKF adalah perluasan dari Kalman Filter untuk model sistem non-linear.

EKF akan mempertahankan posisi dan ketidakpastian robot serta *landmark* pada waktu tertentu. Posisi yang diperkirakan robot serta posisi perkiraan landmark disatukan ke dalam state vektor, dan secara paralel, matriks kovarians yang mewakili ketidakpastian dari masing-masing posisi ini juga dipertahankan. Kalman Filter bekerja dalam tiga langkah yang berbeda: memprediksi, mengukur dan memperbarui.

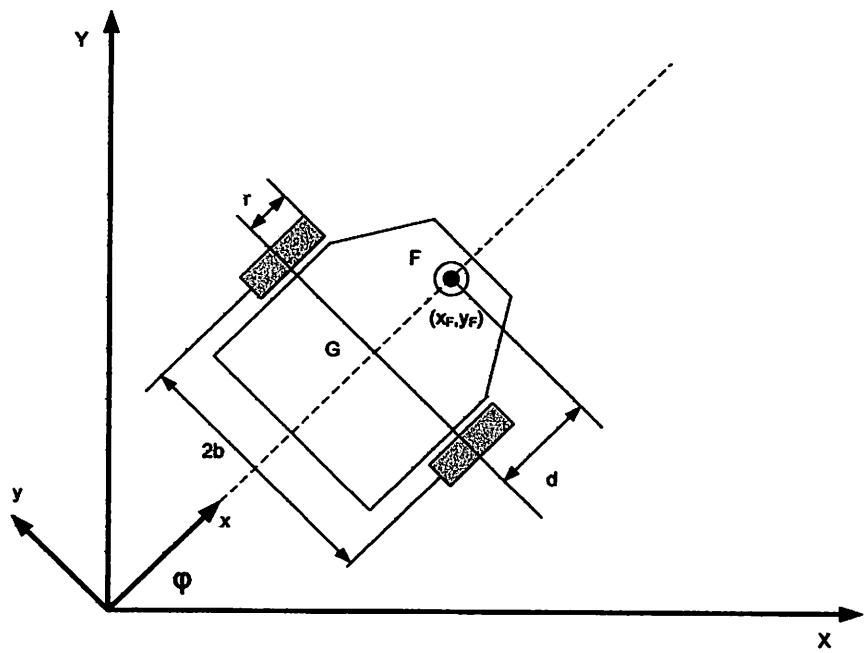
Pada tahap prediksi, vektor *state* dan matriks kovarians untuk waktu berikutnya akan diperkirakan menggunakan state saat ini dan model gerak. Pada

tahap pengukuran, pengukuran jangkauan yang sebenarnya akan dibuat, biasanya menggunakan perangkat *on-board* seperti odometer. Pada tahap pembaruan, berdasarkan kesalahan antara prediksi dan pengukuran, vektor state dan matriks kovarians diperbarui menggunakan penguatan Kalman.

Selain ketidakpastian dalam posisi robot dan fitur poin, matriks kovarians juga merupakan kovarians antara titik-titik fitur. Karena sifat dari perulangan Kalman, kesalahan dan *noise* pada penguatan Kalman berpengaruh pada seluruh matriks kovarians. Dengan demikian, ketidakpastian tentang lokasi setiap titik fitur akan terus menurun seiring dengan sistem yang mengamati lingkungan lebih banyak.

2.3 DDMR (*Differential Drive Mobile Robot*)

Mobile robot adalah robot yang dapat bergerak dari suatu titik ke titik yang lain baik dengan menggunakan roda atau motor yang disusun menyerupai kaki. Salah satu mobile jenis mobile robot yang sering digunakan adalah jenis DDMR (*Differential Drive Mobile Robot*) seperti yang ditunjukkan pada gambar 2.2 berikut:



Gambar 2.2 DDMR Pada Medan 2D Kartesian

Robot diasumsikan berada dalam kawasan 2D pada koordinat Catesian XY.

Parameter-parameter dalam gambar adalah:

φ = Sudut arah hadap robot

2b = lebar robot yang diukur dari garis tengah roda ke roda

R = jari-jari roda (roda kiri dan kanan adalah sama dan sebangun)

D = jarak antara titik tengah antara 2 roda, G dengan titik acuan F

(x,y) = koordinat acuan di tubuh robot terhadap sumbu XY

Dalam kajian kinematik ini robot diasumsikan bergerak relatif pelan dan roda tidak slip terhadap permukaan jalan. Maka komponen x dan y dapat diekspresikan dalam suatu persamaan *nonholonomic* sebagai berikut:

$$\dot{x}_G \sin\varphi - \dot{y}_G \cos\varphi = 0 \quad \text{.....(2.1)}$$

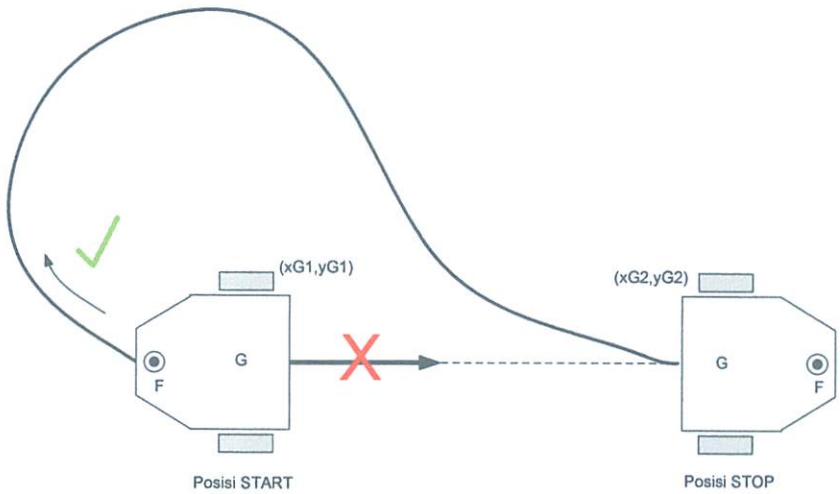
Untuk titik F sebagai acuan analisa, persamaan di atas dapat ditulis:

$$\dot{x}_F \sin\varphi - \dot{y}_F \cos\varphi + \dot{\varphi} = 0 \quad \text{.....(2.2)}$$

Masalah klasik dalam kontrol kinematik DDMR ini adalah bahwa ia memiliki dua aktuator, namun parameter kontrolnya lebih dari dua, yaitu x untuk gerakan ke arah X (1 DOF) dan y untuk arah Y (1DOF) yang diukur relatif terhadap perpindahan titik G dan gerakan sudut hadap φ yang diukur dari garis hubung titik G dan F terhadap sumbu X (1DOF). Inilah ciri khas dari sistem *nonholonomic*.

Dari persamaan (2.1) nampak bahwa derajat kebebasan dalam kontrol kinematiknya berjumlah tiga yaitu : (x,y,φ). Karena ketiga parameter ini perlu dikontrol secara simultan untuk mendapatkan gerakan *nonholonomic*.

Perpindahan kedudukan robot dari START ke STOP bila dipandang pada titik G adalah perpindahan dari koordinat (x_{G1},y_{G1}) ke (x_{G2},y_{G2}) secara translasi. Namun hal ini tidak dapat dilakukan sebab robot harus dikontrol agar bergerak maju, sehingga ia harus membuat manuver belok membentuk lingkaran terlebih hingga pada posisi yang memungkinkan untuk mengarahkannya ke koordinat (x_{g2},y_{G2}). Oleh karena itu diperlukan titik acuan F yang berada di luar garis yang menghubungkan kedua roda agar sudut hadap dapat dihitung.



Gambar 2.3 Contoh Manuver DDMR

Bentuk umum persamaan kinematik untuk DDMR ini dapat dinyatakan dalam persamaan kecepatan sebagai berikut.

$$\begin{pmatrix} \dot{x}_F \\ \dot{y}_F \\ \dot{\theta}_F \end{pmatrix} = T_{NH} \begin{pmatrix} \dot{\theta}_L \\ \dot{\theta}_R \end{pmatrix} \text{ atau } \dot{q}(t) = T_{NH}(q)\dot{\theta}(t) \quad \text{.....(2.3)}$$

$\mathbf{T_{NH}}$ adalah matriks transformasi *nonholonomic*, $\dot{\theta}_L$ dan $\dot{\theta}_R$ adalah kecepatan radial roda kiri dan kanan, dan q adalah sistem koordinat umum robot.

$$q = [x_F, y_F, \varphi]^T \text{ atau } q = \begin{bmatrix} x_F \\ y_F \\ \varphi \end{bmatrix} \quad \text{.....(2.4)}$$

Jika $\mathbf{T_{NH}}$ diuraikan dari persamaan 2.2 dengan memperhatikan gambar 2.3 maka dapat ditentukan,

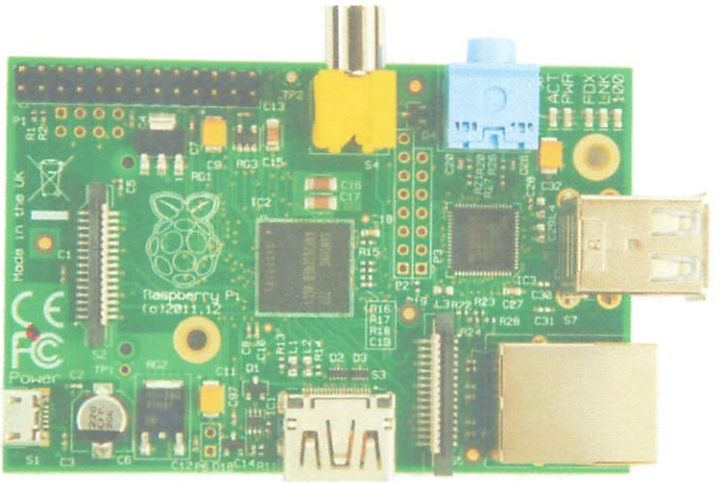
$$T_{NH}(q) = \begin{pmatrix} \frac{r}{2}\cos\varphi + \frac{d \cdot r}{2b}\sin\varphi & \frac{r}{2}\cos\varphi - \frac{d \cdot r}{2b}\sin\varphi \\ \frac{r}{2}\sin\varphi - \frac{d \cdot r}{2b}\cos\varphi & \frac{r}{2}\sin\varphi + \frac{d \cdot r}{2b}\cos\varphi \\ -\frac{r}{2b} & \frac{r}{2b} \end{pmatrix} \quad \text{.....(2.5)}$$

Kinematik inversnya dapat ditulis

$$\dot{\theta}(t) = T_{NH}^{-1}(q)\dot{q}(t) \quad \text{.....(2.6)}$$

2.4 Raspberry Pi

Raspberry Pi adalah sebuah *minicomputer* atau komputer kecil seukuran kartu kredit yang dapat digunakan untuk pembelajaran ataupun untuk aplikasi di bidang kontrol. Seperti layaknya komputer konvensional, Raspberry Pi telah dilengkapi dengan perangkat keras seperti CPU, prosessor, RAM, GPU, I/O USB *Port* dan ethernet *port*.



Gambar 2.4 Raspberry Pi *Board* Model B

(*sumber* : https://www.sparkfun.com/products)

Pada saat ini ada dua model yang telah berada di pasaran yaitu model A dan model B. model A adalah versi terdahulu dari model B sehingga perangkat keras yang ada pun masih terbatas jika dibandingkan dengan model B.

Kebutuhan *power supply* untuk model A dan model B berbeda, hal itu dikarenakan terdapat perbedaan *hardware* yang terpasang pada *board*. Nilai nominal tegangan supply untuk Raspberry Pi adalah 5 volt (*model A dan B*) sedangkan arus *supply* untuk model A adalah 500-700mA sedangkan untuk model B berkisar antara 700-1500mA. Nilai arus *supply* ini tidak mutlak karena nilai arus yang dibutuhkan adalah tergantung dari pemakaian peripheral yang terpasang. Untuk menghindari kerusakan yang terjadi karena kesalahan dalam pemberian supply, pada board Raspberry Pi telah terpasang rangkaian pengaman seperti

dioda proteksi, rangkaian tegangan *clamp*, dan *fuse* yang dapat direset (*resetable fuse*).

Perbedaan spesifikasi model A dan model B dapat dilihat pada tabel 2.1 berikut:

Tabel 2.1 Perbedaan Spesifikasi Model A dan Model B

Hardware	Model A	Model B
SoC (*System on Chip*)	Broadcom BCM2835(CPU+GPU)	
CPU	700Mhz ARM11 ARM1176JZF-S core	
GPU	Broadcom VideoCore IV, OpenGL ES 2.0, OpenVG 1080p30 H.64	
Memori (SDRAM)iB	256 MB	512 MB
USB 2.0 ports	1 buah	2 buah
Video Output	*Composite* video/*Composite* RCA, HDMI	
Audio Output	Konektor TRS/3.5 mm *Jack*, HDMI	
Media penyimpanan	SD/MMC/SDIO	
Ethernet *Port*	Tidak ada	10/100 Ethernet RJ45
I/O	GPIO,SPI(*Serial Peripheral Interface*),I2C,I2S,UART	

Seperti terlihat pada tabel 2.1, prosesor Raspberry Pi adalah ARM. ARM itu sendiri adalah keluarga prosesor RISC (*Reduce Dinstruction Set Computing*) yang digunakan secara ekstensif di pasar perangkat *mobile*. Prosesor ARM pada awalnya dikembangkan oleh A Corn Computer Sin bekerja sama dengan Apple. Meskipun akronim ARM singkatan dari *Advanced* RISC *Machine*, namun pada awalnya adalah Acorn RISC *Machine*.

Untuk dapat beroperasi seperti layaknya komputer maka Raspberry Pi harus diinstall OS (*Operating System*) terlebih dahulu. Untuk saat ini ada beberapa OS yang dapat diinstal pada raspberry pi yaitu: Raspbian OS, Arch Linux ARM, Debian ARM, Fedora Remix, Raspbmc, RISC OS bahkan Android.Sistem operasi yang digunakan semuanya berbasis Linux dan belum mendukung untuk sistem operasi berbasis Windows atau Apple. Sedangkan untuk bahasa pemrograman yang didukung oleh Raspberry Pi diantaranya adalah : Python, C/C++, C# (Mono Develop), Java, Erlang, Pascal (*Lazarus*), PHP, Javascript (*Node JS*).

2.4.1 *Cross Compiler* Raspberry Pi

Kompiler yang berjalan pada komputer akan membuat *binaries* untuk tipe mesin komputer, misal dengan arsitektur intel. Sedangkan untuk Raspberry Pi harus menggunakan kompiler yang dapat membuat binaries untuk arsitektur ARM machine. Untuk itulah dibutuhkan *cross compiler*.

"arm-linux-gnuabihf-gcc" adalah salah satu kompiler yang dapat menjalankan program pada mesin Intel, disamping itu dapat membuat binaries untuk mesin ARM. Sehingga untuk mendevelop dan memprogram program/libraries pada komputer, yang kemudian dapat dideploy dan dijalankan di Raspberry Pi.

Tujuan dari menggunakan *cross compiler* adalah untuk mempercepat performa dalam proses *develop, build* atau kompilasi. Hal ini dilakukan karena Raspberry Pi memiliki *resource* RAM yang terbatas.

Jika komputer berjalan pada sistem operasi dengan 32-bit, pilih "gcc-linaro-arm-linux-gnueabihf-raspbian". Jika komputer berjalan pada sistem operasi dengan 64-bit, pilih "gcc-linaro-arm-linux-gnueabihf-raspbian-x64".

2.4.2 Konfigurasi Raspberry Pi

Pada Raspi-config, pengguna dapat menyelesaikan beberapa tugas konfigurasi dasar dan meningkatkan fungsi unit Raspberry Pi (lihat gambar 12). Meskipun dapat menjalankan Raspberry Pi-*config* setiap saat, sangat ideal untuk melakukan sebagian besar *tweaking* dan kustomisasi sejak awal.

Expand_rootfs: Secara default Raspbian hanya menggunakan SD card yang dibutuhkan sebagai inti sistem operasi. Ketika ingin mengakses seluruh SD card sehingga memiliki banyak ruang penyimpanan untuk proyek-proyek masa depan gunakan tombol panah pada keyboard Anda untuk mengklik ke **expand_rootfs** dan tekan enter lalu akan muncul tulisan "Boot partition has been resized. The file system will be enlarged upon next reboot." Klik **'OK'.**

Overscan: Tidak setiap pengguna akan perlu mengkonfigurasi *overscan* tersebut . Jika terlihat ada ruang hitam yang signifikan di sekitar tepi layar (menggunakan bagian tengah layar , bukan layar penuh) dapat mengaktifkan *overscan* untuk meningkatkan lebar layar. Klik di ***overscan*** , dan kemudian pilih ***'enable'*** .

Configure_keyboard: Gunakan perintah ini untuk mengkonfigurasi keyboard non - US dan mengaktifkan rangkaian layout.

Change_pass: Memungkinkan untuk mengubah *password default* dari ' pi ' menjadi apapun yang diinginkan.

Change_locale: Aktifkan lokal untuk sistem operasi; diperlukan untuk non Inggris.

Change_timezone: Untuk menjaga waktu akurat. Pilih opsi **change_timezone,** kemudian pilih wilayah geografis yang tepat, misal US (Amerika Serikat) , diikuti oleh zona waktu yang sesuai dalam wilayah itu, misal *Eastern* (Timur).

Memory_split: Mengubah sistem mengalokasikan memori bersama untuk GPU dan prosesor utama.

Overclock: Raspberry Pi memiliki prosesor 700MHz ARM . *The Foundation* Raspberry Pi sebenarnya mendukung *overclocking* hingga 1000Mhz (1GHz).

SSH: Untuk mengaktifkan, pilih SSH *server* lalu pilih **'enable'** . Pilih **'OK'** setelah Raspi-config menunjukkan server telah diaktifkan .

Boot_behavior: Di sini kita dapat menunjukkan apakah kita ingin Raspbian untuk *boot up* ke *command line* atau *desktop.*

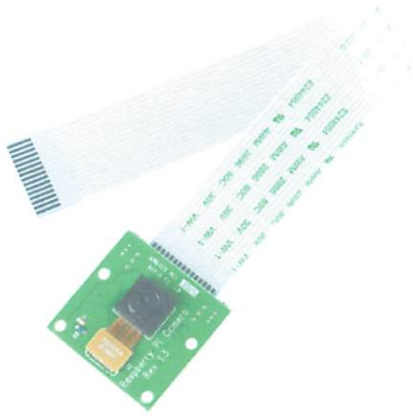
Update: Untuk *upgrade* Raspi-config *tool.*

2.5 Modul Kamera Raspberry Pi

Modul kamera Raspberry Pi adalah suatu perangkat kamera yang khusus didesain untuk digunakan pada *board* Raspberry Pi melalui konektor CSI (*Camera Serial Interface*). Modul kamera ini memiliki dimensi 25 mm x 20 mm x 9 mm dan berat 3 g, sehingga cocok digunakan untuk aplikasi *embedded.*

Modul Kamera Raspberry Pi merupakan perangkat yang sensitif terhadap ESD (*Electrostatic Discharge*) yaitu perpindahan muatan elektrostatic antara dua objek yang terjadi saat kedua objek berbeda potensial. Hal ini terjadi jika tubuh kita terlalu banyak muatan positif sehingga bila kita memegang suatu perangkat yang sensitif maka akan terjadi peristiwa *static discharge* yang bisa merusak *device*. Untuk mencegah terjadinya ESD maka sebelum kita memegang modul

kamera sebaiknya tubuh kita melakukan *grounding* terlebih dahulu dengan memegang permukaan lantai.



Gambar 2.5 Modul Kamera Raspberry Pi

(*sumber*: *https://www.sparkfun.com/products/11868*)

Spesifikasi modul Kamera :

1. Tipe Kamera OmniVision OV5647 Color CMOS QSXGA (5 Megapixel)
2. Ukuran Sensor 3.67 x 2.74 mm
3. Jumlah Piksel= 2592 x 1944
4. Ukuran piksel = 1.4 x 1.4 µm
5. Lensa : f=3.6 mm, f/2.9
6. *Angle of View* (Jangkauan sudut) : 54 x 41 derajat
7. *Field of View* (Jangkauan Pandangan) : 2.0 x 1.33 m pada jarak 2 m
8. Persamaan Lensa SLR (*full frame*) : 35 mm
9. Fokus : 1 m sampai dengan tak terbatas
10. Video : 1080p pada 30 fps dengan codec H.264 (AVC)
11. Kecepatan sampai dengan 90 fps pada mode VGA

Pada skripsi Cosmas Eric, sensor kamera pada robot telah diuji dengan dibagi menjadi dua bagian yaitu pengujian pada sudut vertikal dan pengujian pada sudut horizontal. Prinsip pengukuran sudut menggunakan persamaan segitiga siku-siku dimana tinggi segitiga adalah jarak depan dan alas segitiga adalah jarak halangan dengan titik tengah kamera.

1. Sudut vertikal

   Dari hasil pengujian diketahui bahwa maksimal panjang alas segitiga adalah +10 cm dan -10 cm dan tinggi segitga = 30 cm sehingga sudut θ dapat dicari dengan persamaan :

   $$\tan\theta = \frac{alas}{tinggi}$$

   $$\tan\theta = \frac{10}{30}$$

   $$\tan\theta = 0.3$$

   $$\theta = arc\tan 0.3$$

   $$\theta = 16.69^0$$

   $$\alpha = 2 \times \theta = 33.39^0$$

   Nilai Sudut jangkauan vertikal kamera adalah sebesar 33.39^0

2. Sudut Horizontal

   Dari hasil pengujian diketahui bahwa maksimal panjang alas segitiga adalah -12 dan +13 sedangkan tinggi segitiga = 30 cm sehingga sudut θ dapat dicari dengan persamaan :

   $$\tan\theta = \frac{alas}{tinggi}$$

   $$\tan\theta = \frac{13}{30}$$

   $$\tan\theta = 0.43$$

   $$\theta = arc\tan 0.43$$

   $$\theta = 23.42^0$$

   $$\alpha = 2 \times \theta = 46.85^0$$

   Nilai jangkauan maksimal sudut horizontal kamera adalah $46{,}85^0$.

2.6 Cron

Cron merupakan sebuah *daemon* (pada sistem operasi Windows seperti "Windows *Service*") yang berjalan terus menerus di latar belakang/*background* pada komputer Linux. Tugasnya adalah untuk secara otomatis menjalankan tugas-tugas sesuai dengan jadwal yang juga di komputer. Cron terdiri dari 2 bagian yakni crontab dan crond. Crontab adalah program yang digunakan untuk

mendaftarkan aplikasi yang ingin dijalankan dan juga jadwalnya. Crond adalah daemon yang akan mengecek konfigurasi crontab secara berkala dan menjalankan program sesuai jadwalnya. Setiap pengguna di komputer memiliki crontab mereka sendiri. Ada tiga elemen untuk mengatur cron: cron harus diinstal, cron harus berjalan, dan crontab itu sendiri harus ada.

Cron dijalankan secara otomatis saat *boot*. Hal ini dimulai dari skrip yang berada di struktur direktori /etc/rc. *Daemon* cron (crond) terletak di /etc/init.d, dan *link* untuk terhubung ke direktori itu berada dalam struktur direktori /etc/rc.d/rcN. Sebagai contoh, *link* untuk *runlevel* 5 adalah di /etc/rc.d/rc5.d/S90crond. Ketika dimuat, cron mencari di direktori /var/spool/cron untuk *file* crontab yang memetakan ke akun di /etc/passwd.

Selama sistem operasi berjalan, cron memeriksa semua crontab yang disimpan setiap menit, serta mencari perintah yang harus dieksekusi selama menit tertentu. Setelah berhasil menyelesaikan atau gagal dari perintah, cron menciptakan output yang secara *default* kepada pemilik crontab. Selama pengecekan dari menit ke menit, cron juga memeriksa apakah salah satu crontab telah dimodifikasi (dengan memeriksa modtime pada direktori spool cron yang mendapat perubahan oleh program crontab setiap kali crontab berubah), dan *reload* jika hal itu terjadi. Sehingga tidak harus berhenti dan restart cron jika akan mengubah crontab.

Jika komputer sedang *down* atau cron dihentikan selama periode pengerjaan tugas yang dijadwalkan, tugas itu tidak akan dijalankan setelah komputer dihidupkan atau cron *restart*. Hal ini memiliki implikasi ketika perubahan jam sistem, seperti waktu siang. Jika Anda memiliki cronjob dijadwalkan selama jam yang dilewati di musim semi ketika *daylight savings time* menyebabkan jam dipindahkan dari 1 AM menjadi 2 AM, tugas itu tidak akan berjalan sama sekali. Sebaliknya, jika Anda memiliki tugas yang dijadwalkan selama jam pada musim gugur ketika jam diatur kembali, tugas akan dijalankan dua kali. Dengan kata lain, cron tidak bisa ditempatkan dalam antrian itu, jika komputer down untuk jangka waktu panjang, dapat dilakukan *rescanned* untuk menjalankan pekerjaan yang dilewati selama *downtime*.

Aturan crontab yaitu:

```
# .---------------- menit (0 - 59)
# |  .-------------- jam (0 - 23)
# |  |  .------------ hari keberapa dalam sebulan (1 - 31)
# |  |  |  .---------- bulan (1 - 12) atau (jan,feb,mar,apr ...)
# |  |  |  |  .------- hari keberapa dalam seminggu (0 - 6) (Sunday=0 or 7)
# |  |  |  |  |        atau (sun,mon,tue,wed,thu,fri,sat)
# |  |  |  |  |
# *  *  *  *  *--- user-name command yang akan di eksekusi
```

Tabel 2.2 Contoh Cara Kerja *Sintaks* Cron

Cron *Sintaks*	*Command* Berjalan Setiap
05 * * * *	Berjalan setiap 1 jam sekali lewat 5 menit, Contoh: Pukul 00.05, 01.05, 02.05, dst.
05 02 * * *	Berjalan setiap hari pukul 02:05 AM
30 20 01 * *	Berjalan pukul 08:30 PM tiap awal bulan. (Hanya hari pertama di tiap bulan)
00 07 25 12 *	Berjalan pada tiap tanggal 25 Desember, pukul 07:00 AM
30 16 * * 5	Berjalan setiap jumat pada pukul 4:30 PM
*/5 * * * *	Berjalan setiap 5 menit sekali (0,5,10,15,....,45,50,55)
*/10 9-16 1,15 * *	Berjalan setiap 10 menit antara 9am dan 4pm (Instance terakhir akan berjalan pada jam 3.50pm) di hari pertama dan hari ke lima belas setiap bulan.

Seperti terlihat pada tabel di atas, *wildchar* yang dapat digunakan adalah karakter *asteriks* atau "*" yang berarti *field* yg ditandai "*" diabaikan. Untuk tanda *slash* yang diikuti angka "/<angka>" menandakan tiap pada rentang waktu tertentu.

2.7 RSync

RSync adalah sebuah aplikasi perangkat lunak *open source*, awalnya ditulis untuk sistem Unix, tetapi sekarang juga berjalan pada *platform* Windows dan Mac. RSync digunakan untuk sinkronisasi *file* dan direktori dari satu lokasi ke lokasi lain dan meminimalkan transfer data antara masing-masing lokasi. Transfer data diminimalkan dengan menggunakan algoritma yang akan mengirimkan, atau

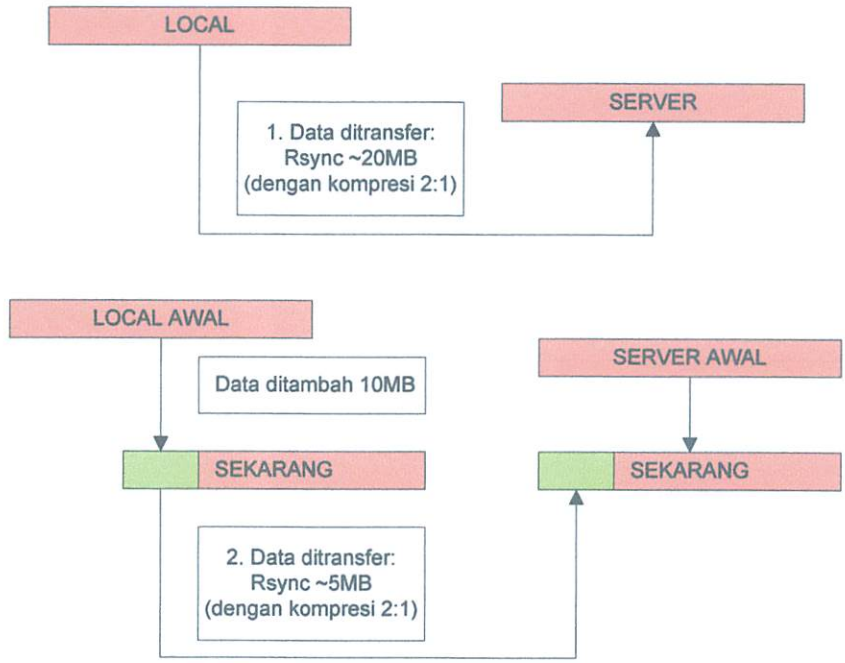
hanya sebagian kecil yang dari *backup* data yang telah berubah, sampai ke tingkat bit (teknologi ini juga dikenal sebagai *in-file delta incremental* transfer).

Beberapa fitur RSync:

1. Dapat memperbarui struktur seluruh direktori dan sistem *file*.
2. Secara opsional mempertahankan *link* simbolik, *hard link*, kepemilikan berkas, hak akses, perangkat dan waktu.
3. Tidak memerlukan hak-hak istimewa untuk menginstal.
4. *Pipelining* internal yang mengurangi *latency* untuk beberapa *file*.
5. Dapat menggunakan rsh, ssh atau soket langsung sebagai transportasi.
6. mendukung anonim RSync yang ideal untuk mirroring.

Selain itu Rsync juga mengkompres semua paket data yang dikirim untuk mengurangi perpindahan yang berlebihan. RSync menggunakan metode *checksum* untuk melakukan transfer data tingkat bit. Metode ini dengan menciptakan string alfanumerik pendek berdasarkan data yang diwakilinya. Pertama-tama RSync mengecek dahulu apakah data telah berubah dengan melihat ukuran file dan tanggal modifikasi. Jika tidak ada data yang berubah, Rsync tidak akan mentransfer data, menghemat waktu dan *bandwidth* (lihat gambar 2.6 di bawah).

Jika file tidak cocok, RSync menggunakan metode *checksum* yang disebut "*rolling checksum*" pada *file* yang diubah untuk melihat di mana telah diubah atau ditambahkan. Kemudian RSync hanya akan mentransfer data yang berubah atau ditambahkan ke dalam *file*. RSync dapat melayani data yang dimasukkan atau ditambahkan, data yang dihapus, serta data yang digeser, dengan *overhead* transfer yang minimal.



Gambar 2.6 Pengiriman RSync

2.8 *Library* Scene (SceneLib)

Scene adalah *library* yang fleksibel untuk SLAM di C++, memberikan cara umum untuk menerapkan SLAM di aplikasi yang berbeda tanpa mereplikasi metode. Seperti yang diketahui, SLAM merupakan urutan penyebaran probabilitas distribusi yang mewakili perkiraan ketidakpastian posisi dari pergerakan robot dan lokasi fitur yang diamati. Sehingga Scene bertujuan untuk memisahkan algoritma inti probabilitas propagasi dari rincian aplikasi tertentu yang diperlukan untuk keberhasilan dalam aplikasi dunia nyata. Rincian ini dapat dipasang ke Scene sebagai kelas "model" yang dipakai oleh pengguna untuk template yang didefinisikan dengan baik untuk mewakili spesifik sistem yang nyata.

Perangkat lunak ini telah berkembang secara bertahap dan signifikan dari sumber asli sistem SLAM menggunakan visi komputer, khusus untuk satu robot. Hal itu didorong oleh kebutuhan dan keinginan untuk menerapkan banyak komponen yang ditransfer ke berbagai masalah dalam visi komputer dan robotika.

Dalam beberapa tahun terakhir sampai dengan tahun 2006 Dr. Davidson telah sukses besar dalam menggunakan Scene secara *real-time*, 3D SLAM menggunakan satu kamera yang kemudian disebut MonoSLAM. Scene telah dikembangkan pada dunia nyata dengan aplikasi *real-time* sebagai tujuannya,

tetapi dengan fleksibilitas sehubungan dengan spesifik robot dan sensor. Hal ini telah secara substansial dicapai sejauh ini karena robot dengan jenis karakteristik gerakan dapat dikombinasikan dengan semua jenis sensor untuk mengukur setiap jenis fitur; atau dengan beberapa jenis sensor pengukuran yang berbeda jenis fitur.

Akhirnya, Scene dapat digunakan untuk melakukan estimasi gerak dan pemetaan untuk sejumlah robot atau sensor lain, bergerak atau *stasioner*, mengamati berbagai aspek statis atau dinamis lingkungan. Ini harus mencakup semua jenis masalah pelacakan sebagai sistem SLAM tradisional. Selain itu, tujuan desain asli adalah untuk menawarkan juga fleksibilitas sehubungan dengan algoritma pilihan, yang disediakan oleh beberapa metode untuk menyebarkan distribusi probabilitas. EKF adalah satu-satunya alat manajemen ketidakpastian saat ini ditawarkan dalam Scene.

Subdirektori utama SceneLib adalah Scene, SceneImproc dan MonoSLAM, masing-masing berisi kode yang dikompilasi ke dalam perpustakaan yang terpisah dengan nama yang sama.

Library inti Scene menyediakan semua dukungan utama untuk aplikasi membangun peta.

Library SceneImproc berisi beberapa fungsi pengolah gambar seperti korelasi dan pencarian yang mungkin digunakan secara umum dalam aplikasi yang menggunakan pencitraan.

Library MonoSLAM menyediakan dukungan khusus untuk SLAM dengan kamera tunggal.

Isi signifikan rinci SceneLib adalah sebagai berikut:

1. Docs/

   models.tex:

   Informasi lengkap mengenai kerangka kerja umum dalam adegan untuk mewakili spesifik aplikasi dengan model matematika yang mewakili spesifikasi di Scene, dan penjelasan prosedur yang diperlukan untuk menerapkan Scene.

2. Scene/

   models_base.h/cpp:

   Berisi kelas dasar abstrak untuk gerak robot dan model pengukuran fitur.

scene_single.h/cpp dan feature.h/cpp:

Berisi kelas Scene_Single yang menyimpan secara penuh kovarians EKF peta lokasi robot dan fitur. Kelas Fitur merupakan salah satu fitur di peta.

kalman.cpp:

Berisi fungsi dan data untuk melakukan *update* pada data yang disimpan dalam Scene_Single menggunakan EKF.

internal_measurement.h/cpp:

Dukungan untuk pengukuran internal.

sim_or_rob.cpp dan simul.cpp:

Menyediakan kemampuan untuk mengkompilasi semua aplikasi dalam mode simulasi serta mode aslinya, dan simulasi *noise* robot dan lingkungannya ketika kita ingin menjalankan dalam mode simulasi.

3. Contains/

settings.h/cpp:

Penanganan untuk kelas yang akan dijalankan dari file teks saat peluncuran dieksekusi.

control_general.cpp:

Berisi beberapa fungsi umum, berlaku untuk mengendalikan kelas yang dapat dipanggil dari program aplikasi inidividual.

newbits.cpp:

Beberapa potongan kecil yang digunakan oleh sisa perhitungan.

exceptions.h:

Handler untuk kesalahan runtime.

identifier.h/cpp:

Kelas dasar untuk bagian data mengidentifikasi fitur (misalnya gambar patch)

4. SceneImproc/

correlate.cpp:

Fungsi korelasi gambar.

improc.cpp:

Fungsi untuk mencari gambar dan deteksi fitur yang menarik.

5. MonoSLAM/

models_impulse.h/cpp:

Mendefinisikan model gerak yang berasal dari generik ThreeD_Motion_Model di Scene, yang mewakili gerakan halus dari kamera.

models_zeroorder.h/cpp:

Sebuah alternatif posisi konstan (*random walk*) model gerak.

models_wideangle.h/cpp:

Kelas model pengukuran fitur untuk titik kamera.

model_creators.h/cpp:

Contoh dari kelas model dapat digunakan di sini ketika inisialisasi disajikan.

widecamera.h/cpp:

Sebuah abstraksi dari kamera dengan atau tanpa distorsi radial (digunakan oleh models_wideangle.h tetapi juga dapat digunakan secara lebih luas).

identifier_imagemono.h:

Mendefinisikan kelas membungkus patch gambar untuk digunakan sebagai identifier dari fitur dalam peta Scene.

monoslam.h/cpp dan monoslaminterface.h:

Mendefinisikan MonoSLAM, kelas yang mengontrol operasi utama program. Semua kelas utama yang dipakai sebagai anggota dari MonoSLAM. Urutan tindakan perintah real-time utama berada di sini, ditambah berbagai fungsi untuk melaporkan *state* dan mengubah parameter. *File* monoslaminterface.h hanya mendefinisikan beberapa antarmuka tambahan untuk MonoSLAM jika mungkin diperlukan.

robot.h/cpp:

Mengimplementasikan antarmuka Scene untuk melakukan pengukuran dan mengendalikan robot (walaupun tidak harus diperlukan robot).

Robot berasal dari Sim_Or_Rob, *library* antarmuka di SceneLib/Scene baik untuk simulasi atau aplikasi robot.

nonoverlappingregion.h/cpp:

Mendefinisikan fungsi pelayanan untuk menemukan sebuah daerah di distribusi dan kamera *state*.

threeddraw.h/cpp:

Fungsi untuk menggambar keadaan kamera dan fitur menggunakan VW::ThreeDToolGLCommon untuk tampilan 3D. Berisi fungsi yang berbeda untuk menggambar eksternal kamera untuk *monitoring*, pelacakan atau melakukan *augmented reality*.

6. include/
7. libs/

Setelah kompilasi dan 'make install', *file* include dan libs akan muncul di sini.

BAB III
PERANCANGAN DAN ANALISA SISTEM

3.1 Pendahuluan

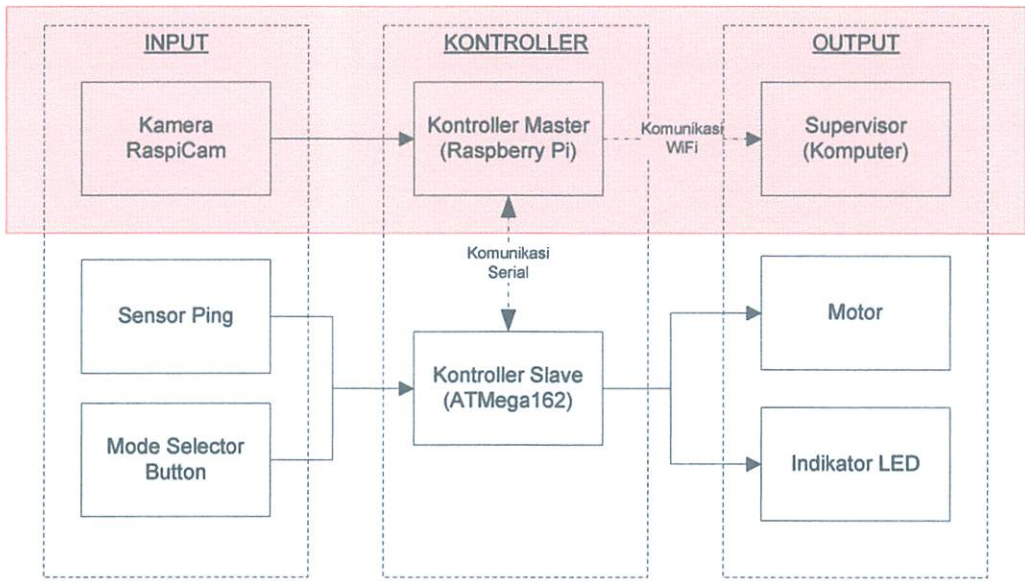
Pada bab ini akan membahas mengenai perencanaan sistem, prinsip kerja dan perancangan perangkat lunak baik dari sisi robot maupun komputer supervisor yang berkaitan dengan keperluan sistem yaitu desain sistem pengambilan, penyimpanan, dan transfer gambar yang diimplementasikan pada robot DDMR, serta pengaksesan dan pengolahan gambar yang mengimplementasikan konsep Visual SLAM sehingga menghasilkan peta pada komputer supervisor. Pada perancangan ini akan diimplementasikan konsep dan teori dasar yang telah dibahas sebelumnya, sehingga tujuan dari perencanaan dapat tercapai dengan baik.

3.2 Kebutuhan Sistem

Kebutuhan sistem yang akan dirancang akan dibagi menjadi dua bagian utama yaitu perangkat keras (*hardware*) dan perangkat lunak (*software*). Kebutuhan sistem perangkat keras pada robot meliputi bagian input, kontroler, output. Pada bagian input terdiri dari rangkaian sensor yang dipakai yaitu 1 modul kamera Raspberry Pi dan 3 buah modul sensor jarak berbasis ultrasonik dan rangkaian switch (*push button/tactile switch*) untuk menentukan mode yang diinputkan oleh user.

Bagian kontroler terdiri dari modul SBC (*Single Board Computer*) Raspberry Pi yang dikombinasikan dengan mikrokontroler Atmega162 dengan konsep Master Slave dimana Raspberry Pi dikonfigurasikan sebagai master dan mikrokontroler Atmega162 sebagai slave. Raspberry Pi dipilih sebagai kontroler master karena memiliki *resource* yang cukup besar dan memiliki kecepatan pengolahan yang cepat jika dibandingkan dengan mikrokontroler konvensional. Sedangkan untuk bagian output terdiri dari indikator LED untuk mengetahui status robot, data-data yang dikirim ke komputer, serta motor untuk menggerakkan robot maju, mundur, dan belok.

Diagram blok kebutuhan sistem keseluruhan dapat dilihat pada gambar di bawah,



Gambar 3.1 Kebutuhan Sistem Keseluruhan

Penjelasan:

1. Input, masukan nilai aktual dari sensor dan mode *selector* dari user. Bagian input terdiri dari:
   a. Sensor kamera yang akan mengambil gambar.
   b. Sensor ultrasonik ping kiri atau kanan yang akan membaca nilai jarak robot terhadap dinding kiri/kanan.
   c. Sensor ultrasonik depan digunakan untuk membaca jarak halangan depan.
   d. Mode *selector* digunakan untuk menentukan mode telusur kiri atau kanan.
2. Kontroler, yaitu bagian pengolahan dari nilai yang dibaca oleh sensor.

   Kontrol pada perancangan sistem ini dibagi menjadi 2 bagian yaitu kontroler *master* (Raspberry Pi) dan kontroler *slave* (ATmega162).
3. Output, bagian output terdiri dari beberapa bagian, yaitu :
   a. Keluaran gambar yang dikirimkan melalui WiFi untuk diproses pada aplikasi komputer.
   b. Keluaran Parameter Fuzzy yang dikirimkan melalui WiFi untuk ditampilkan pada aplikasi *online Debugging*.
   c. *Driver* motor yang akan menggerakkan motor.

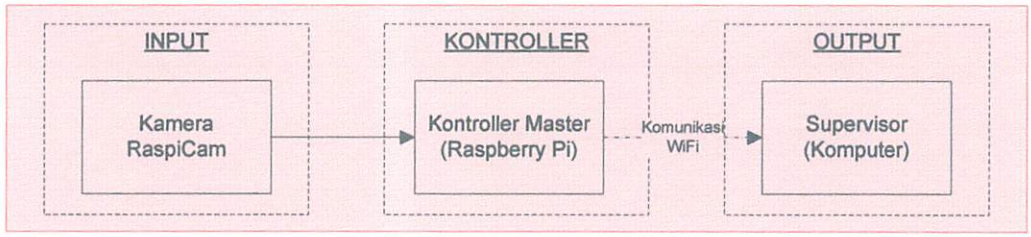
d. *Display* LED, digunakan untuk menampilkan mode yang dipilih oleh user dan aktivitas yang sedang terjadi.

3.3 Perancangan Sistem

3.3.1 Prinsip Kerja

Fokus sistem yang akan dirancang adalah pengembangan sistem *mobile* robot dalam suatu lingkungan tertentu (lihat gambar 3.2), robot yang pada penelitian sebelumnya telah dapat bernavigasi pada trajektori dalam bentuk sekat-sekat dinding dimana terdapat halangan-halangan di dalam trajektori tersebut, akan dikembangkan sehingga selain bernavigasi, robot juga mampu mengambil dan menyimpan gambar untuk selanjutnya diakses oleh komputer lalu diolah agar menghasilkan peta dalam bentuk 2 dimensi, karena peta adalah 2D sehingga bentuk peta adalah trajektori yang dilewati robot.

Pada saat *start*, robot terlebih dahulu akan membaca penekanan *push button* oleh pengguna yang akan menentukan mode telusur dinding yang akan dijalankan yaitu mode telusur dinding kanan atau mode telusur dinding kiri. Setelah mode ditentukan maka robot pada kondisi *standby*, namun karena Raspberry Pi adalah sebuah komputer yang memerlukan waktu untuk *booting* maka *start* robot harus menunggu sinyal bahwa Raspberry Pi (kontroler master) telah siap melakukan proses kontrol. Jika kontroler *master* telah siap melakukan pemrosesan maka robot dapat *start*.

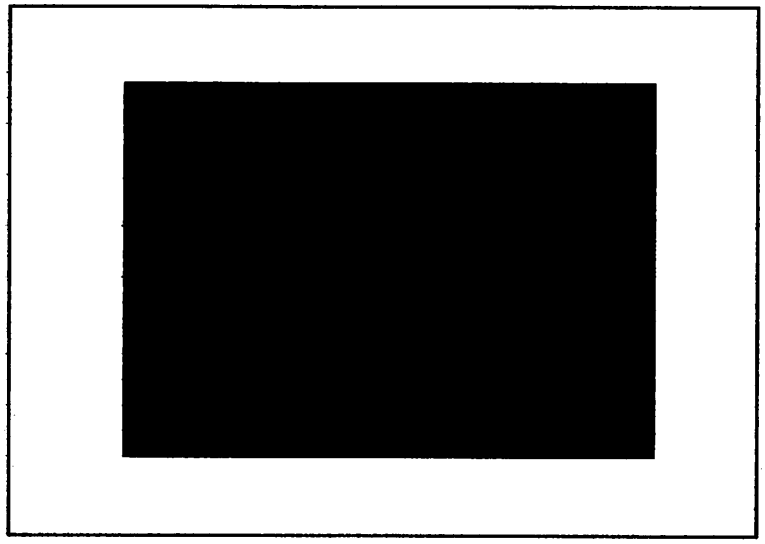


Gambar 3.2 Diagram Blok Fokus Sistem

3.4 Perancangan Target Kalibrasi

Untuk menjalankan sistem dibutuhkan suatu target kalibrasi. Target ini digunakan untuk mendeteksi fitur/*patch* pada jarak yang telah diketahui dari kamera, sehingga dapat dilakukan penskalaan pada benda lain. Tanpa inisialisasi awal target, sistem tidak dapat mengetahui ukuran asli suatu objek, misal membedakan antara kecil dan dekat atau besar dan jauh.

Perancangan deskriptor ini berupa kertas putih ukuran A4 yaitu 21 cm x 29,7 cm, yang ditempel kertas hitam ukuran A5 yaitu 14,8 cm x 21 cm (setengah A4) dengan posisi *landscape* (lihat gambar 3.3). Deskriptor ini akan ditempel di dinding dengan jarak tertentu dari kamera robot.



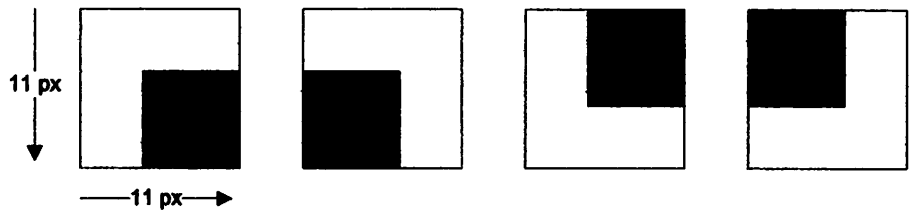
Gambar 3.3 Target Kalibrasi/Deskriptor

Deskriptor yang dirancang memiliki warna dengan tepi yang tegas, yaitu hitam dengan putih. Diharapkan dengan warna tersebut pengolahan gambar menjadi lebih mudah, karena program detektor mampu mengenali deskriptor dengan cepat.

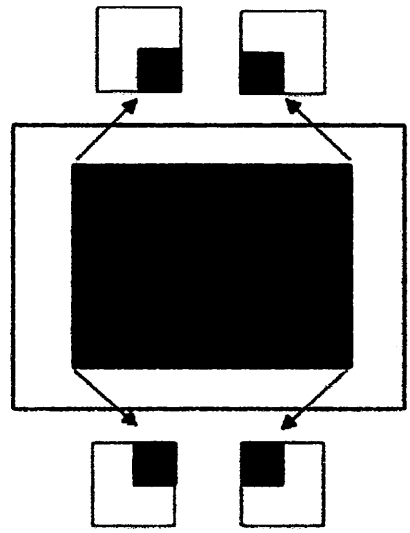
3.5 Perancangan Patch

Hal pertama yang harus dilakukan dalam Visual SLAM adalah menginisialisasi *patch* pada *scene*. Program akan *scanning frame* mencari fitur yang baik untuk dilakukan *tracking*. Fitur tersebut harus memilki gradien yang jelas sehingga dapat dideteksi apabila terjadi perubahan arah cahaya. Fitur ini juga harus bagian unik dari gambar. Setelah ditemukan, akan dilakukan kalibrasi kamera.

Pada perancangan kali ini *patch* yang akan digunakan berukuran 11x11 piksel, yang diambil dari sisi pojok target deskriptor (lihat gambar 3.4 dan 3.5).



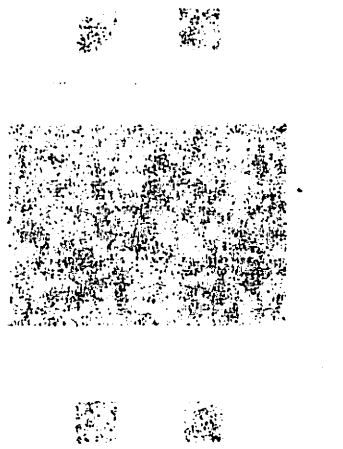
Gambar 3.4 Komponen *Patch*

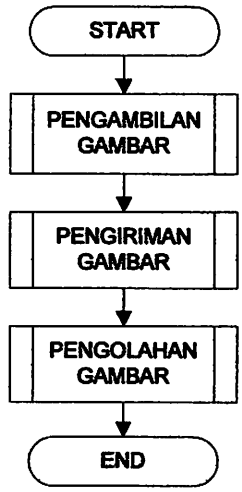


Gambar 3.5 Ilustrasi Pengambilan *Patch* dari Deskriptor

3.6 Perancangan Perancangan Pemrosesan Gambar

Gambar sebagai komponen pembentuk peta akan melalui proses seperti terlihat pada gambar 3.6, dengan subproses yang akan dibahas masing-masing pada subbab berikutnya.Subbab tersebut akan membahas perancangan untuk pengambilan gambar, pengiriman gambar, dan pengolahan gambar yang dilakukan baik pada robot maupun pada komputer supervisor.





Gambar 3.6 Diagram Alur Pemrosesan Gambar

3.6.1 Perancangan Pengambilan Gambar

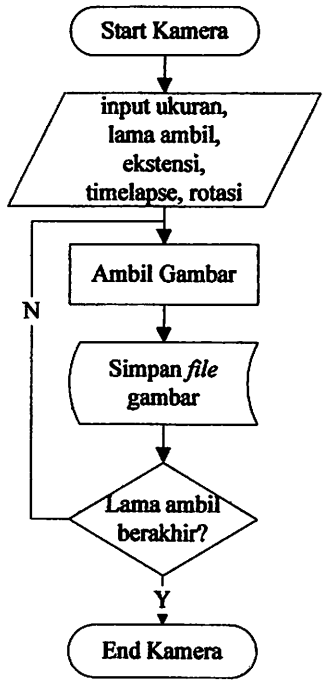
Pengambilan gambar dilakukan oleh sensor kamera pada robot, metode yang dilakukan adalah metode *timelapse*, yaitu pengambilan gambar secara sekuensial terus menerus selama waktu yang ditentukan. Sehingga kamera akan terus mengambil gambar sampai waktu berakhir (lihat gambar 3.7).

Jumlah gambar yang diambil bergantung pada beban prosessor, jadi tidak dapat ditentukan jumlah *frame per second* (fps). Gambar yang diambil akan disimpan pada *file* gambar ekstensi jpg dengan diberi urutan. Urutan tersebut yang nantinya akan diambil dan diolah oleh komputer.

Berikut adalah perancangan parameter gambar yang diambil:

Ukuran	:	W 320 H 240
Lama Pengambilan	:	Saat robot berjalan
Timelapse	:	0
Nama *File*	:	rawoutputXXXX
Ekstensi	:	.jpg
Rotasi	:	180°

Seperti terlihat pada perancangan parameter, terdapat rotasi sebesar 180°. Rotasi tersebut berguna untuk memutar gambar, gambar diputar karena kabel kamera pada robot bersifat sensitif, untuk itu kabel dibiarkan menggantung bebas tanpa tersentuh mekanik robot dan mengharuskan kamera pada posisi terbalik.

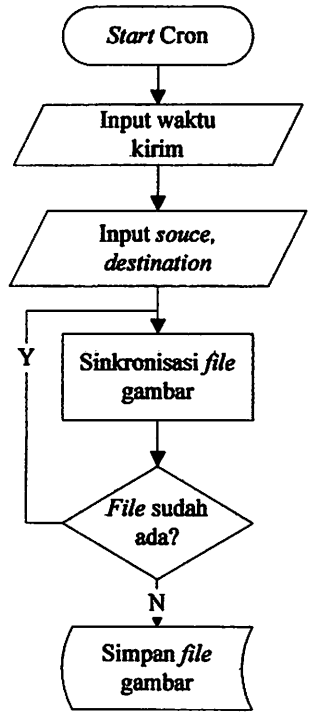


Gambar 3.7 Diagram Alur Subrutin Pengambilan Gambar

3.6.2 Perancangan Pengiriman Gambar

Pada perancangan pengiriman gambar, komputer bertugas mengambil gambar yang disimpan di robot untuk selanjutnya dikirim dan disimpan di komputer. Metode yang dipakai adalah RSync (*Remote Synchronization*). Keunggulan metode ini yaitu tidak mengirim *file* yang telah ada di komputer, sehingga waktu pengiriman menjadi cepat dan efisien.

Dikarenakan RSync adalah *shell command* yang hanya akan dieksekusi ketika *command* diketikkan di layar terminal, maka dibutuhkan suatu program yang mengotomatisasi RSync tersebut, program yang dirancang adalah program Cron yang telah dimodifikasi sehingga mampu mengeksekusi RSync secara terus-menerus dalam resolusi waktu yang kecil yaitu *milisecond* (lihat gambar 3.8).



Gambar 3.8 Diagram Alur Subrutin Pengiriman Gambar

Di sisi *client* (robot):

1. *Client* menyediakan suatu direktori tempat berisinya data-data gambar yang nanti akan dibackup secara berkala oleh RSync.
2. *Client* mengaktifkan *share* direktori yang ditunjuk agar *server backup* RSync dapat mengambil data pada direktori tersebut.

Di sisi server backup RSync (komputer):

3. *Server* menyiapkan direktori *backup* yang nantinya akan digunakan sebagai lokasi *backup* data dari *client*.
4. *Server* melakukan proses penjadwalan backup data terhadap *client* secara periodik melalui Cron.
5. *Server backup* menjalankan *RSync* melalui bash script, artinya proses transfer data dari *client* menuju *server* backup pada waktu-waktu tertentu.
6. Setelah *server backup* melakukan proses RSync ke *client* maka seluruh isi direktori *client* yang telah *dishare* akan *dicopy* ke *server backup* secara keseluruhan, karena bersifat sinkronisasi maka kedua sisi direktori akan dibuat sama oleh RSync.

Pada Cron terdapat inefisien pada rentang penjadwalan, yaitu Cron hanya berjalan pada resolusi waktu *second* atau detik. Sedangkan kebutuhan sistem adalah *real-time*. Untuk itu dirancang sebuah program yang mengeksekusi RSync pada resolusi *millisecond* namun tetap dengan menggunakan konsep Cron.

Pada perancangan sistem, jumlah eksekusi per detik dibuat 70 kali. Sehingga didapat 86ms untuk tiap pengiriman gambar (lihat perhitungan di bawah).

Perhitungan waktu pengiriman:

$$pengiriman = \frac{1\ sekon}{\sum eksekusi}$$

$$pengiriman = \frac{6000}{70}$$

$$pengiriman = 86\ ms$$

Berikut adalah perancangan eksekusi Cron:

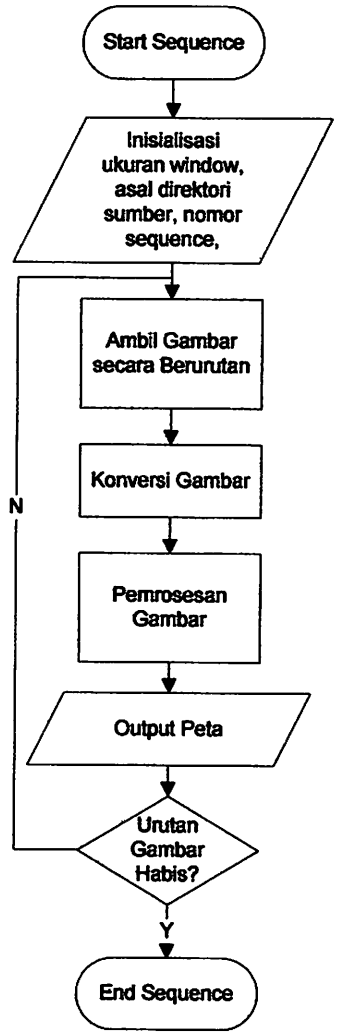
1. Pada *start-up*, mencari *file* bernama .crontab dalam direktori *home* untuk semua akun. Untuk setiap *file* crontab yang telah ditemukan, menentukan waktu berikutnya untuk menjalankan setiap perintah di masa depan.
2. Menempatkan setiap perintah pada daftar tugas dengan waktu yang sesuai "*five field specifier*"
3. Pada program utama
    a. Memeriksa daftar tugas di antrian, menghitung seberapa lama di masa depan tugas itu harus dijalankan.
    b. Sleep/menunggu hingga jangka waktu yang ditentukan.
    c. Pada *awakening*/saat eksekusi dan setelah memverifikasi waktu yang tepat, melaksanakan tugas di antrian (di latar belakang) dengan hak istimewa dari pengguna yang menciptakannya.
    d. Menentukan waktu berikutnya di masa depan untuk menjalankan tugas tersebut dan kembali pada daftar antrian tugas pada waktu itu.

3.6.3 Perancangan Pengolahan Gambar

Driver untuk kamera Raspberry Pi tidak mendukung pengambilan gambar dengan ekstensi .pgm, sedangkan pada program dibutuhkan gambar dengan

ekstensi .pgm. Untuk itu dirancang pengolahan agar gambar yang berekstensi .jpg diubah menjadi ekstensi .pgm, yaitu dari *channel* RGB ke *Grayscale*. Hal ini didasari deskriptor dan *patch* yang berwarna hitam-putih. Apabila gambar tetap dipertahankan pada mode RGB dikhawatirkan detektor akan kesulitan mencari fitur pada gambar.

Pengambilan gambar *sequence* dimulai dari angka 1. Waktu pergantian antar frame adalah 200 mS. *Sequence* akan diakhiri apabila gambar yang disequence habis (lihat gambar di bawah).



Gambar 3.9 Diagram Alur Subrutin Pengolahan Gambar

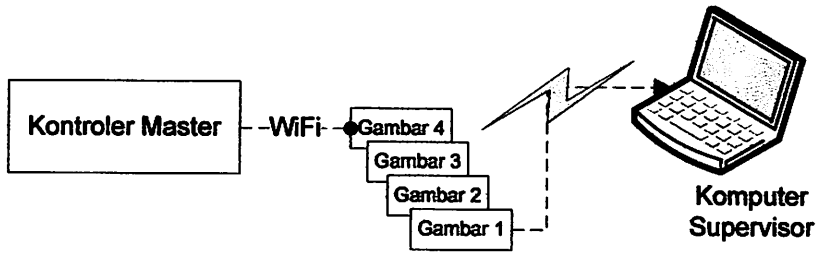
3.7 Perancangan Komunikasi

Komunikasi dilakukan via WiFi, dilakukan dengan pemberian IP *static* pada kontroler master (*Raspberry Pi*) dengan komputer. *Port* yang digunakan adalah *port* 22 untuk SSH. Secara *default*, Raspberry Pi akan diberikan

IP secara otomatis oleh router (disebut Dynamic IP atau DHCP) ketika terhubung ke jaringan. Namun, ini bisa berubah setiap kali terlepas dari jaringan misalnya saat Raspberry Pi mati. Memiliki IP statis tidak selalu penting, namun akan membuat akses ke Raspberry Pi *via* SSH lebih sederhana, karena akan selalu tahu bahwa Raspberry Pi memiliki alamat yang sama.

Alamat IP Raspberry Pi : 192.168.1.1
Alamat IP Komputer : 192.168.1.2

Karena Raspberry Pi membutuhkan *password* untuk tiap kali akses, sedangkan komputer harus mengambil data gambar dari Raspberry Pi secara terus-menerus dan otomatis, maka dilakukan *bypass password* pada Raspberry Pi, khusus untuk pengambilan gambar pada direktori gambar yang sudah dishare (lihat gambar di bawah).



Gambar 3.10 Komunikasi Antara Robot dengan Komputer

BAB IV
PENGUJIAN DAN PEMBAHASAN SISTEM

4.1 Pendahuluan

Bab ini ditujukan untuk melakukan pengujian dan pembahasan dari sistem yang telah dirancang sebelumnya agar dapat diketahui bagaimana kinerja dari keseluruhan sistem maupun kinerja masing-masing bagian. Dari hasil pengujian tersebut akan dijadikan dasar untuk menentukan kesimpulan serta mengetahui kekurangan yang harus segera diperbaiki agar kinerja keseluruhan sistem sesuai dengan perencanaan dan perancangan yang telah dibuat.

Pengujian dan pembahasan sistem ini akan dibagi dalam 3 bagian utama, yaitu :

1. Pengujian dan pembahasan perangkat lunak robot yang meliputi pembahasan pengujian inisialisasi target awal, pengujian pengambilan gambar dari modul kamera pada robot yang bergerak, pengujian jarak dari modul WiFi robot ke komputer.
2. Pengujian dan pembahasan perangkat lunak komputer yang meliputi pengujian pengaksesan gambar dari direktori robot, pengujian penyimpanan target, pengujian *timelapse* gambar, pengujian pencarian fitur pada lingkungan, pengujian pembuatan peta.
3. Pengujian keseluruhan sistem.

4.2 Pengujian Iniasialisasi Deskriptor

Tujuan pengujian inisialisasi terget awal untuk mengetahui sejauh mana target yang telah ditentukan dapat diinialisasi oleh kamera robot. Fungsi dari inisialisasi target awal adalah untuk melakukan kalibrasi dan penetuan poin fitur yang nantinya akan dijadikan acuan/patokan untuk membuat peta.

4.2.1 Peralatan yang digunakan

1. Robot DDMR lengkap dengan modul kamera RaspiCam.
2. Alat ukur panjang berupa penggaris.
3. Deskriptor target berupa kertas putih A4 dengan tengah warna hitam A5.
4. Program pengambil gambar.
5. Komputer dengan koneksi WiFi.

4.2.2 Langkah-langkah yang dilakukan

1. Hidupkan Robot DDMR pada mode diam/*standby*.
2. Hubungkan komputer dengan robot melalui WiFi dengan SSID "Mark-3AP".
3. Koneksikan komputer ke Raspberry Pi dengan memasukkan IP 192.168.1.1.
4. Buka terminal Raspberry Pi menggunakan aplikasi Putty.
5. Jalankan program pengambilan gambar dari jarak-jarak yang telah ditentukan.
6. Catat hasil pengambilan target.

4.2.3 Hasil Pengujian

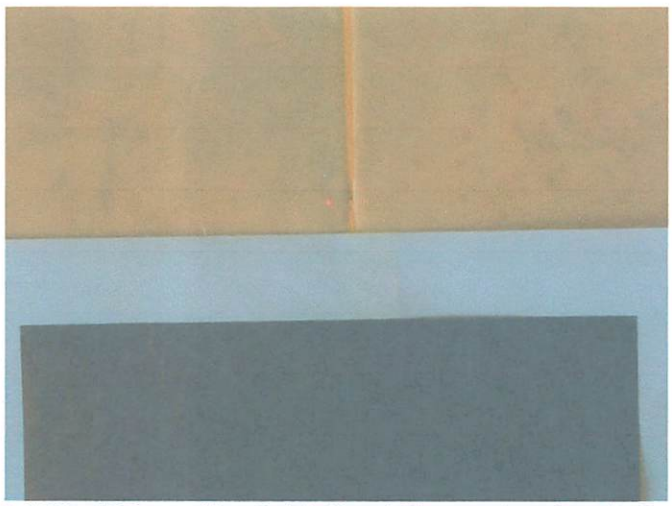
Tabel 4.1 Hasil Pengujian Inisialisasi Target Awal

Jarak Robot dengan Target (cm)	Jarak Target dengan Tanah (cm)	Inisialisasi Target
60	0	X
	15	X
120	0	√
	15	√
180	0	√
	15	√

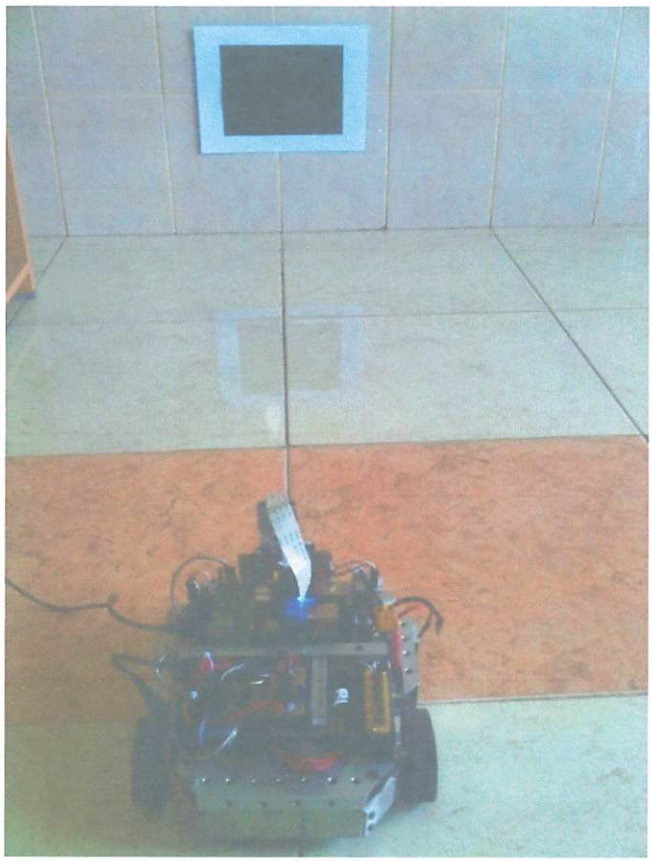
Keterangan :

√ = Berhasil 4 *corner*

X = Kurang dari 4 *corner*



Gambar 4.1 Hasil Pengujian Inisialisasi Target Awal pada Jarak 60 cm dengan ketinggian target 0 cm dari tanah



Gambar 4.2 Pengujian Inisialisasi Target Awal pada Jarak 180 cm

4.2.4 Analisa Pengujian

Dari hasil pengujian terlihat pada gambar 4.1 target awal tidak mampu dideteksi dengan baik oleh kamera RaspiCam sehingga tidak dapat dijadikan acuan untuk inisialisasi target awal, sedangkan pada gambar 4.2 target awal terlihat 4 buah pojok/*corner* mampu dideteksi oleh robot. Kesimpulan pada analisa ini yaitu untuk jarak inisialisasi target awal harus dilakukan lebih dari jarak 60 cm baik di ketinggian 0 atau 15 dari permukaan lantai.

4.3 Pengujian Pengambilan Gambar pada Robot Bergerak

Robot yang sedang bergerak harus mampu mengambil dan menyimpan gambar secara bersamaan, hasil gambar tersebut diharapkan tidak kabur (jelas), karena gambar yang kabur akan membuat program *supervisor* kesulitan dalam menentukan poin fitur/*patch* yang dicari.

4.3.1 Peralatan yang digunakan

1. Robot DDMR lengkap dengan modul kamera RaspiCam.
2. Program pengambil gambar.
3. Komputer dengan koneksi WiFi.

4.3.2 Langkah-langkah yang dilakukan

1. Hidupkan Robot DDMR.
2. Hubungkan komputer dengan robot melalui WiFi dengan SSID "Mark-3AP".
3. Koneksikan komputer ke Raspberry Pi dengan memasukkan IP 192.168.1.1.
4. Buka terminal Raspberry Pi menggunakan aplikasi Putty.
5. Jalankan program pengambilan gambar dari tempat-tempat yang telah ditentukan.
6. Catat hasil pengambilan gambar.

4.3.3 Hasil Pengujian

Tabel 4.2 Hasil Pengujian Pengambilan Gambar pada Robot Bergerak

FPS	Lingkungan	Hasil Gambar
5	Lurus	√
	Sudut	√
	Putar	X
10	Lurus	√
	Sudut	√
	Putar	X
15	Lurus	√
	Sudut	√
	Putar	X
20	Lurus	√
	Sudut	√
	Putar	√
25	Lurus	√
	Sudut	√
	Putar	√

Keterangan :

√ = Berhasil

X = Gambar kabur

Gambar 4.3 Pengujian Pengambilan Gambar pada Lintasan Kontes Robot Pemadam Api Indonesia (di Lab. Robotika ITN Malang)

Gambar 4.4 Pengujian Pengambilan Gambar pada Lingkungan Natural (di Dalam Rumah)

4.3.4 Analisa Pengujian

Pada tabel 4.2 diatas memiliki dua variabel input yaitu FPS *(Frame per Second)* dan lingkungan/area, dua variabel input tersebut dipilih karena masing-masing memiliki potensi yang menyebabkan gambar hasil *capture* menjadi kabur. Dari pengujian di atas, dapat dianalisa bahwa semakin tinggi nilai FPS, gambar yang berhasil ditangkap semakin banyak sehingga *sequence* gambar *(timelapse)* yang dihasilkan menjadi lebih lembut. Di sisi lain, semakin banyak liukkan yang dilakukan robot akibat mengikuti lingkungan/area menyebabkan gambar kabur secara horizontal, karena robot bergerak tidak stabil menyebabkan fokus kamera hilang.

4.4 Pengujian Pengujian Nomor Urutan Pengambilan Gambar

Robot yang sedang bergerak harus mampu mengambil (*capture*) namun juga menyimpan gambar secara bersamaan, pengujian ini berfungsi untuk mengetahui sejauh mana hasil penyimpanan gambar secara berurutan, karena pada landasan teori maksimal RaspiCam mampu *capture* hingga 30 fps. Sedangkan untuk *capture* dan simpan belum diketahui hasil maksimalnya. Percobaan dilakukan selama 6 detik.

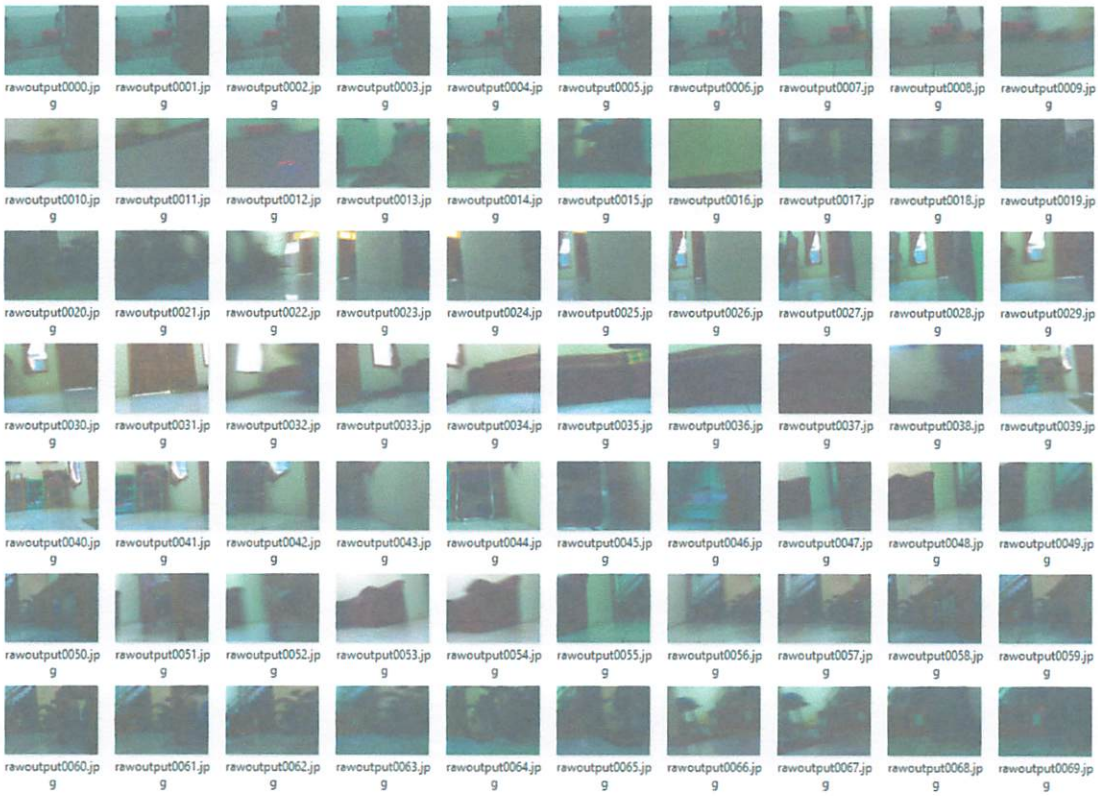
4.4.1 Peralatan yang digunakan

1. Robot DDMR lengkap dengan modul kamera RaspiCam.
2. Program pengambil gambar.
3. Komputer dengan koneksi WiFi.

4.4.2 Langkah-langkah yang dilakukan

1. Hidupkan Robot DDMR.
2. Hubungkan komputer dengan robot melalui WiFi dengan SSID "Mark-3AP".
3. Koneksikan komputer ke Raspberry Pi dengan memasukkan IP 192.168.1.1.
4. Buka terminal Raspberry Pi menggunakan aplikasi Putty.
5. Jalankan program pengambilan gambar dari tempat-tempat yang telah ditentukan.
6. Catat hasil pengambilan gambar.

4.4.3 Hasil Pengujian



Gambar 4.5 Hasil Pengambilan Urutan Gambar

Pada gambar di atas terlihat hasil pengujian pengurutan gambar, gambar yang diambil harus berurutan, apabila gambar tidak urut, yaitu ada yang hilang atau *corrupt*. Maka sequence pada program akan mengeluarkan perintah untuk menambahkan gambar pada urutan selanjutya dan berhenti mengeksekusi pembuatan peta hingga gambar selanjutnya diletakkan pada direktori penyimpanan gambar pada komputer.

Pengambilan dilakukan dengan menggunakan perhitungan timelapse 0, sehingga robot tidak dipaksa untuk menghasilkan gambar sekian pada waktu yang ditentukan. Apabila dilakukan pemaksaan akan menyebabkan kamera terdesak untuk memenuhi tuntuan jumlah gambar, namun mengabaikan urutan. Pengabaian urutan ini mengakibatkan kesalahan *sequence* yang fatal pada komputer. Gambar di atas adalah berjumlah 70 gambar yang disimpan pada robot lalu diambil oleh komputer secara bertahap.

Tabel 4.3 Hasil Pengujian Nomor Urutan Pengambilan Gambar

FPS	Nomor Urutan Hilang	Hasil Urutan Gambar
5	-	X
	15	
	-	
10	10	X
	40	
	60	
15	10,11	X
	33,35	
	68,71	
20	11,12,13	X
	40,45,46	
	100,111	
25	3,4,6,...	X
	66,67,69,...	
	111,113,...	

Keterangan:

X = Gambar Tidak Urut

4.4.4 Analisa Pengujian

Pada tabel di atas, fps *(frame per second)* sangat mempengaruhi urutan gambar, untuk itu dibuat program timelapse dengan fps tidak tentu, artinya dinamis mengikuti kemampuan penyimpanan.

4.5 Pengujian Jarak WiFi Robot ke Komputer

4.5.1 Peralatan yang digunakan

1. Robot DDMR.
2. Program pengambil gambar.
3. Komputer dengan koneksi WiFi.

4.5.2 Langkah-langkah Pengujian

1. Hidupkan Robot DDMR.
2. Hubungkan komputer dengan robot melalui WiFi dengan SSID "Mark-3AP".
3. Koneksikan komputer ke Raspberry Pi dengan memasukkan IP 192.168.1.1.
4. Buka terminal Raspberry Pi menggunakan aplikasi Putty dan *start* robot.
5. Jalankan program pengambilan gambar dari tempat-tempat yang telah ditentukan.
6. Catat hasil pengambilan gambar.

4.5.3 Hasil Pengujian

Tabel 4.4 Hasil Pengujian Jarak WiFi Robot ke Komputer

Jarak Robot dengan Komputer (m)	Koneksi WiFi
2	√
4	√
6	√
8	√
10	√
12	√

Keterangan :

√ = Koneksi WiFi Terhubung

4.5.4 Analisa Pengujian

Pada tabel di atas dapat diketahui jarak robot hingga 12 meter tidak mempengaruhi komunikasi antara robot dengan WiFi komputer. Karena modul WiFi yang digunakan memiliki kemampuan memancar yang handal.

4.6 Pengujian Transfer Gambar

Komputer harus mampu mengambil gambar yang telah disimpan oleh robot yang sedang bergerak.

4.6.1 Peralatan yang digunakan

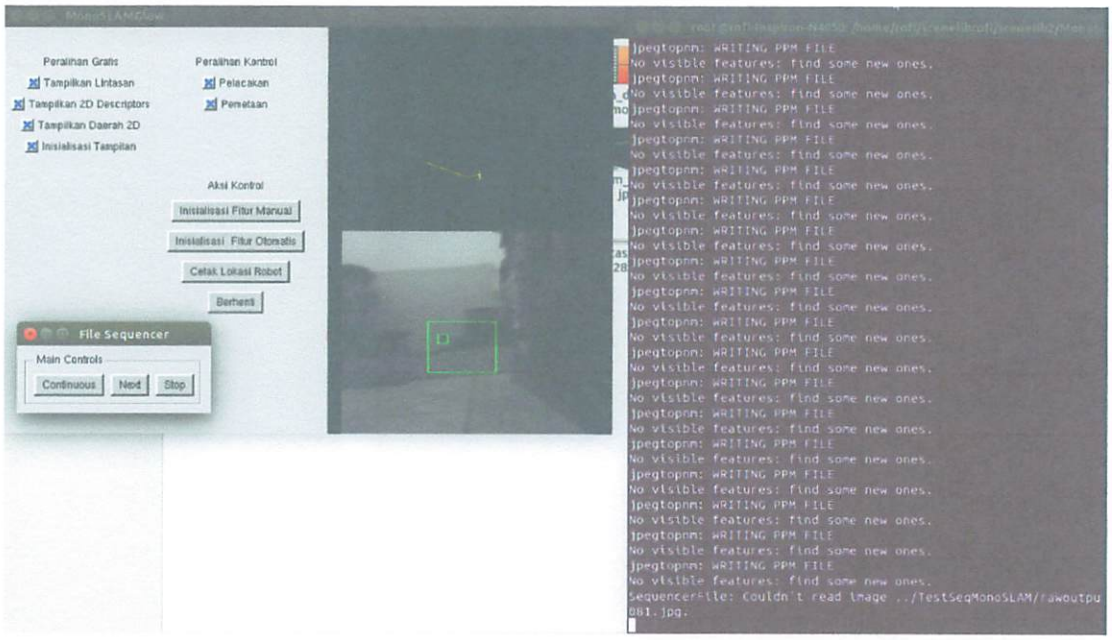
1. Robot DDMR lengkap dengan modul kamera RaspiCam.

2. Program pengambil gambar.
3. Komputer dengan koneksi WiFi.

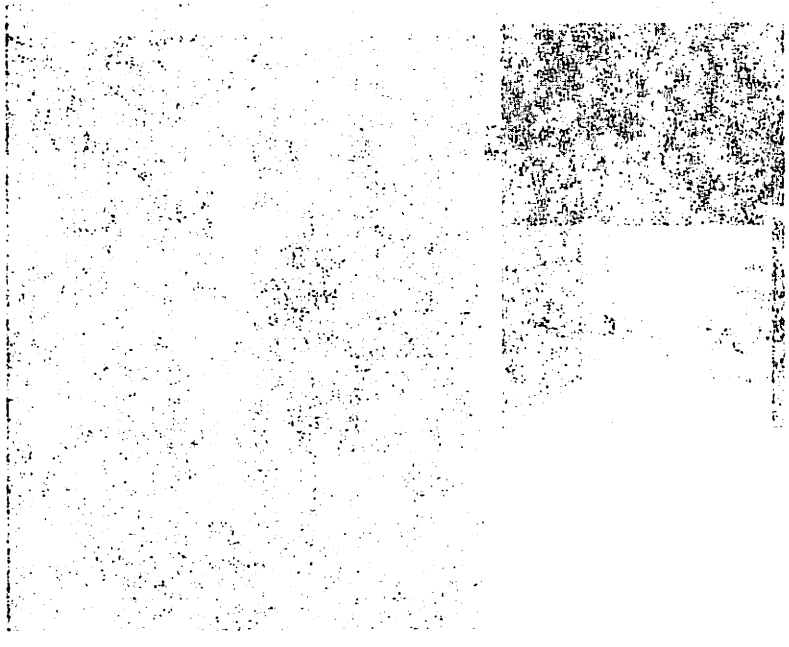
4.6.2 Langkah-langkah yang dilakukan

1. Hidupkan Robot DDMR.
2. Hubungkan komputer dengan robot melalui WiFi dengan SSID "Mark-3AP".
3. Koneksikan komputer ke Raspberry Pi dengan memasukkan IP 192.168.1.1.
4. Buka terminal Raspberry Pi menggunakan aplikasi Putty.
5. Jalankan program pengambilan gambar dari tempat-tempat yang telah ditentukan.
6. Catat hasil pengambilan gambar.

4.6.3 Hasil Pengujian



Gambar 4.6 Hasil Pengambilan Urutan Gambar



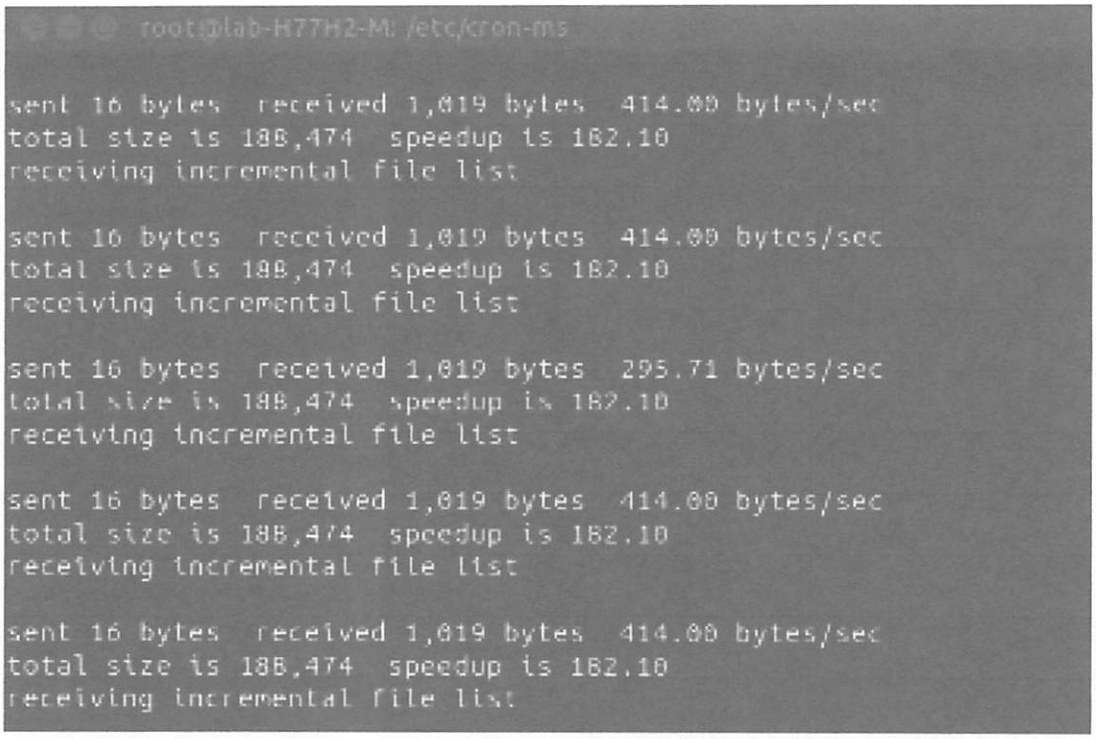
Tabel 4.5 Hasil Pengujian Transfer Gambar

Jadwal Eksekusi	**∑ gambar disimpan robot**	**∑ gambar transfer di komputer**
15 kali/detik	2	2
	2	2
	3	3
	2	2
	3	3
10 kali/detik	3	3
	4	4
	5	5
	4	4
	3	3
5 kali/detik	5	5
	6	6
	7	7
	8	8
	8	8

4.6.4 Analisa Pengujian

Sejauh ini seluruh gambar terkirim dengan sempurna dan lengkap tanpa ada yang *corrupt*.

Gambar di bawah menunjukkan hasil pengiriman gambar dari robot ke komputer, apabila terjadi kesalahan pengiriman maka akan muncul tampilan *error*.



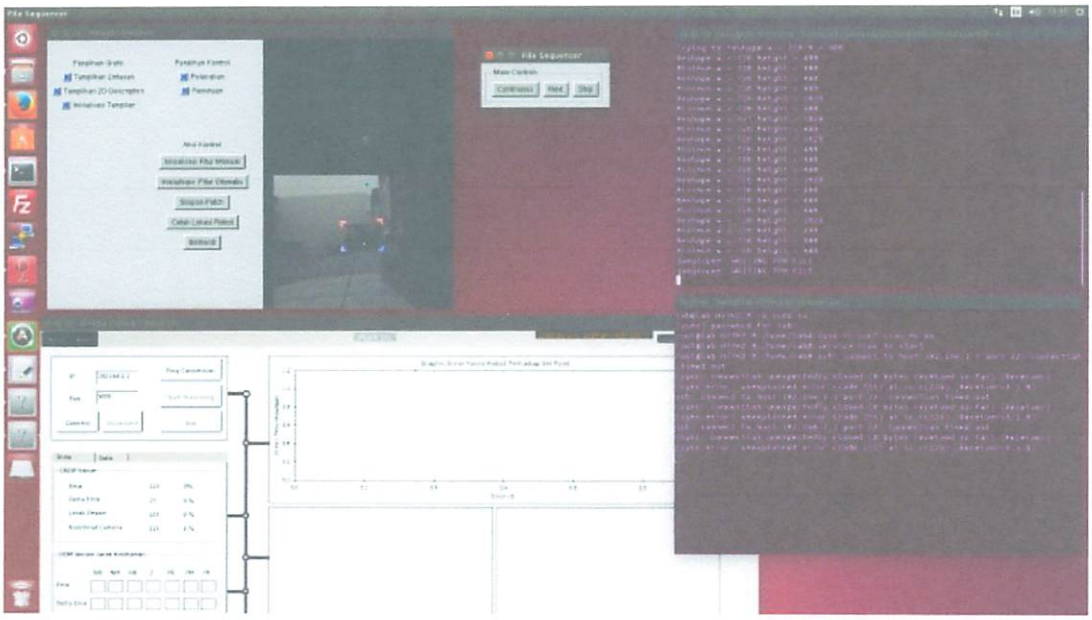
Gambar 4.7 Hasil Pengiriman Gambar

4.7 Pengujian Peta pada Komputer

Berikut adalah hasil pengujian peta pada komputer, pengujian dilakukan di laboratorium pemrograman komputer dan multimedia Teknik Elektro S-1 ITN Malang.

Gambar di bawah adalah tampilan keseluruhan program, dapat dilihat ada empat windows/jendela aktif, masing-masing jendela memiliki fungsi yaitu untuk tampilan hasil peta, informasi pembentukan peta, serta informasi pengiriman gambar. Sedangkan jendela keempat untuk melihat *error* dari pergerakan robot.

Pada saat program aplikasi dibuka akan muncul patch yang telah disimpan, dan ditempatkan pada jarak tertentu seperti terlihat pada gambar di bawah. Inisialisasi *patch* dapat disebut dengan *ekstraksi fitur* yang merupakan suatu pengambilan ciri/*feature* dari suatu bentuk yang nantinya nilai yang didapatkan akan dianalisis untuk proses selanjutnya.



Gambar 4.8 Keseluruhan Aplikasi

Gambar 4.9 Hasil Pengambilan Gambar Awal

Gambar 4.10 Hasil Gambar Setelah Dikonversi

Pengecekan dilakukan dalam berbagai arah *tracking* pengecekan pada koordinat kartesian dari citra digital yang dianalisis, yaitu vertikal, horizontal, diagonal kanan, dan diagonal kiri.

```
Reading patch known_patch0.pgm
VW::ReadPPMImage(mono)
Added known feature 1
Creating a CAMERA_WIDE_POINT feature measurement model
Reading patch known_patch1.pgm
VW::ReadPPMImage(mono)
Added known feature 2
Creating a CAMERA_WIDE_POINT feature measurement model
Reading patch known_patch2.pgm
VW::ReadPPMImage(mono)
Added known feature 3
Creating a CAMERA_WIDE_POINT feature measurement model
Reading patch known_patch3.pgm
VW::ReadPPMImage(mono)
Added known feature 4
```

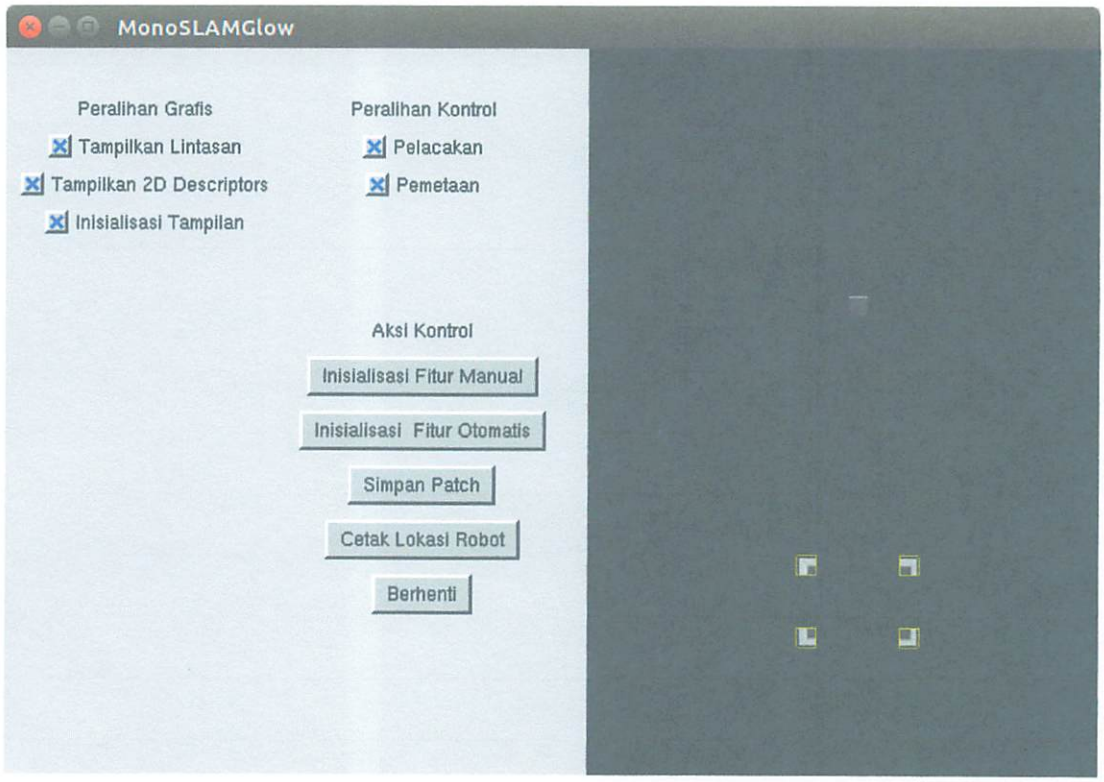
Gambar 4.11 Hasil Penambahan *Patch* ke Gambar Deskriptor

Gambar 4.12 Ukuran dan Gambar Asli *Patch* Nol

Ukuran dan Gambar Asli *Patch* Satu

Gambar 4.14 Ukuran dan Gambar Asli *Patch* Dua

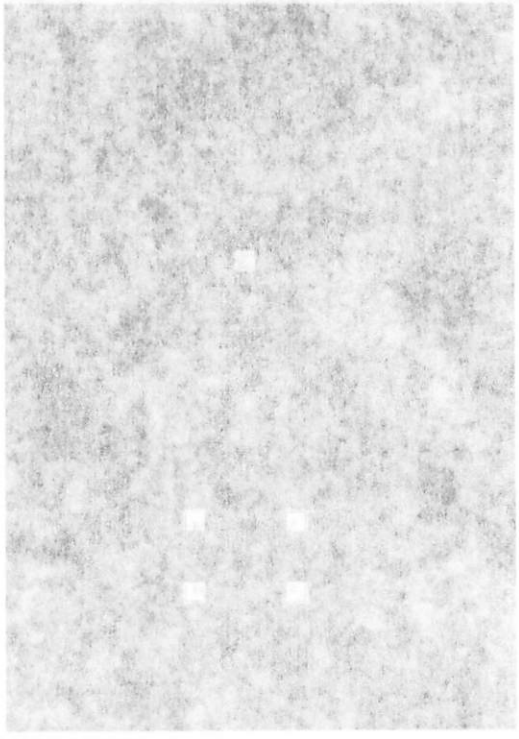
Gambar 4.15 Ukuran dan Gambar Asli *Patch* Tiga



Gambar 4.16 Inisialisasi Patch Awal

Ketika *start* deskriptor akan muncul, apabila terdeteksi warna merah maka deskriptor dinyatakan benar dan siap untuk membuat peta seperti terlihat pada gambar 4.17.

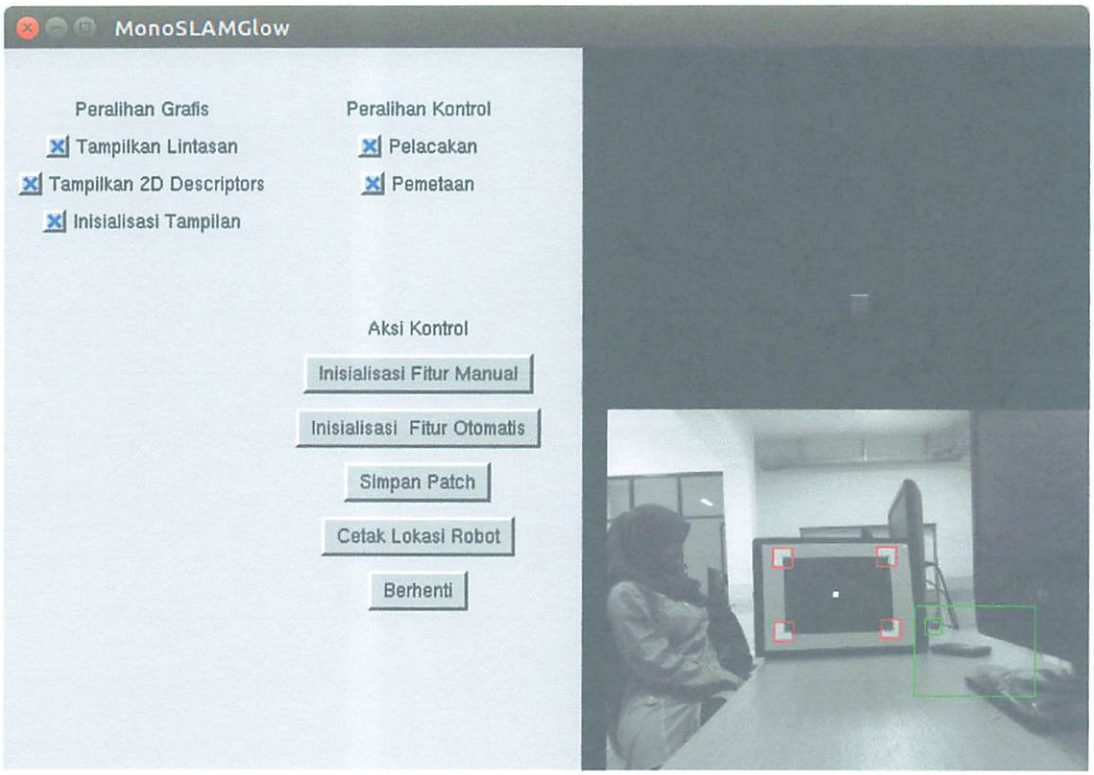
Algoritma pecocokan fitur mencoba untuk mencocokkan fitur dari satu gambar ke gambar selanjutnya, dalam rangka menjami pencocokan yang stabil, maka diperlukan fitur deskriptor untuk selanjutnya dijadikan landmark



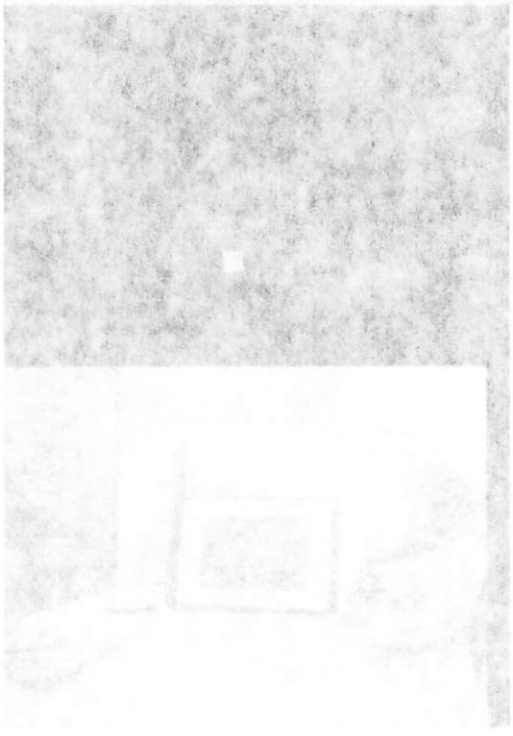
Gambar 4.10 Ilustrasi Patch Aktif

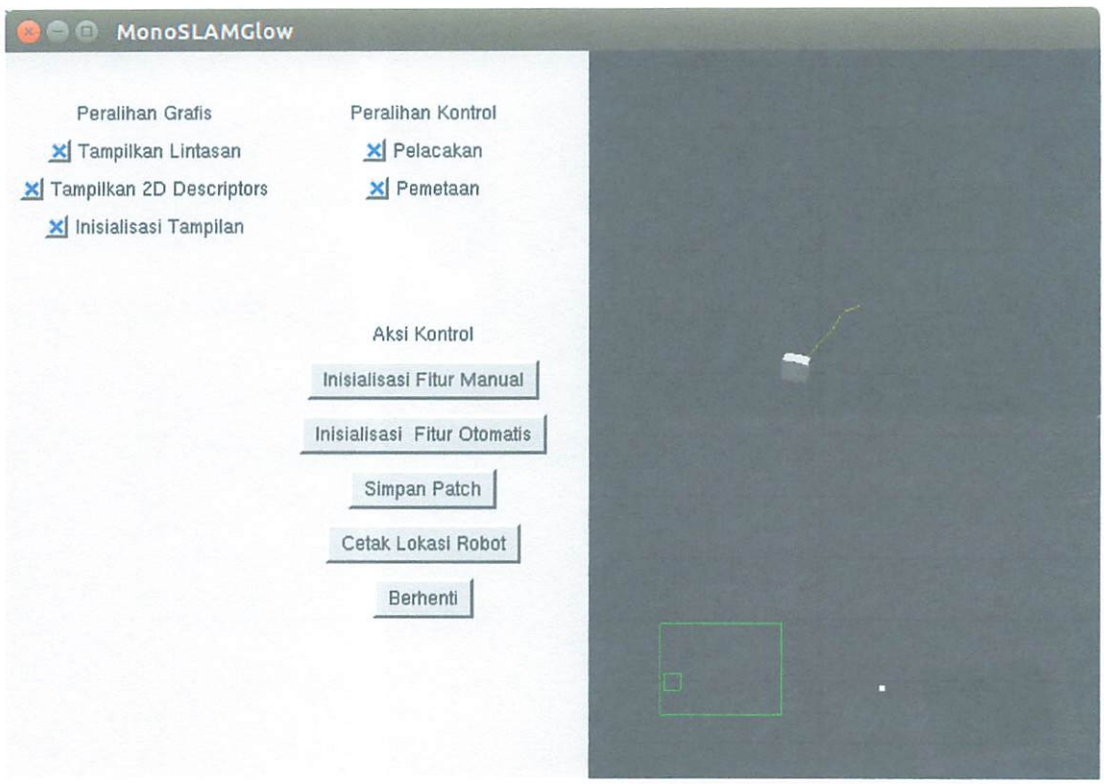
membuat peta. Algoritma pecocokan fitur kemudian akan mencocokkan bagian pola gambar yang memiliki bentuk sama dengan *patch* deskriptor.

Box/Kotak hijau adalah fitur-*detektor*, berguna untuk mencari pola pada bagian gambar yang cocok dengan *patch* (lihat gambar 4.17).



Gambar 4.17 Patch Terdeteksi di Deskriptor



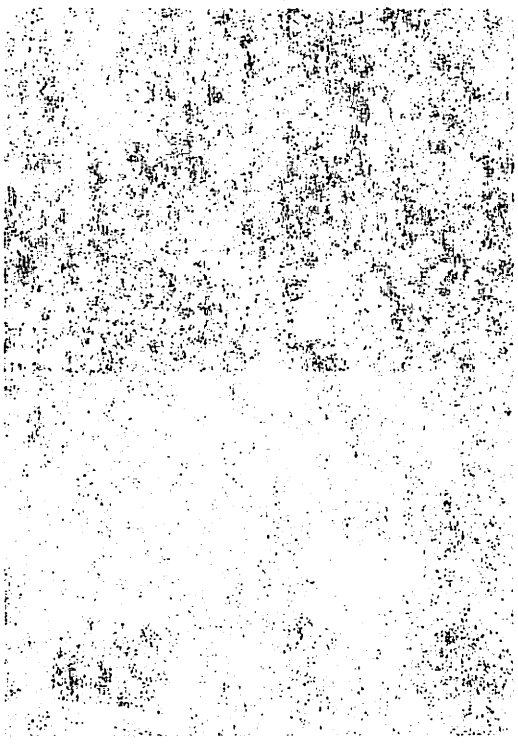


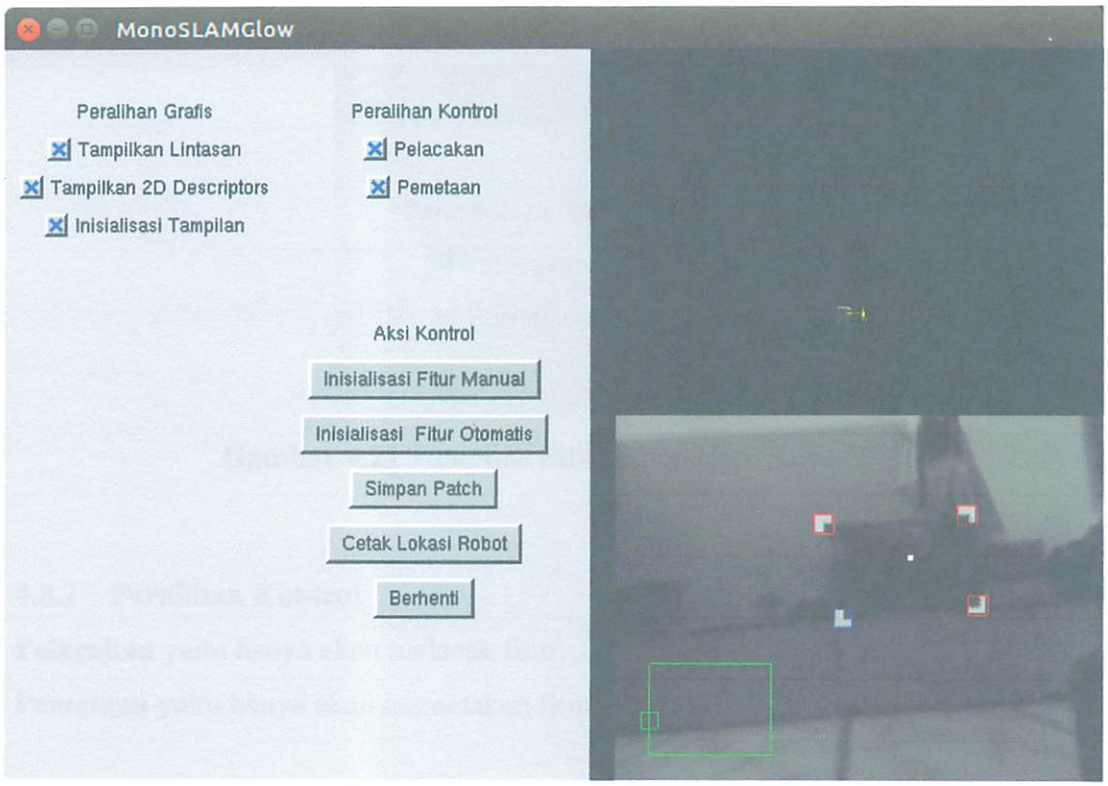
Gambar 4.18 Hasil Peta pada Cahaya Gelap

Pada lingkungan dengan cahaya gelap, detektor akan kesulitan menemukan patch, ditandai dengan muncul peringatan pada jendela informasi pembentukan peta, hal ini berdampak pada hasil peta, dan berpotensi menjadikan peta salah arah (lihat gambar 4.18).

Pada lingkungan dengan cahaya terang, detektor akan menemukan patch, ditandai dengan adanya kotak merah dan biru, merah berarti pola gambar sesuai dengan patch, biru berarti pola gambar tidak sesuai *patch* atau *failed patch*.

Peta diidentifikasi benar apabila pergerakan belok robot diikuti dengan berbeloknya peta pada komputer, seperti terlihat pada gambar di bawah.





Gambar 4.19 Hasil Peta pada Cahaya Terang

4.8 Komponen pada Software

Gambar 4.20 Tampilan Pilihan Peralihan Grafis

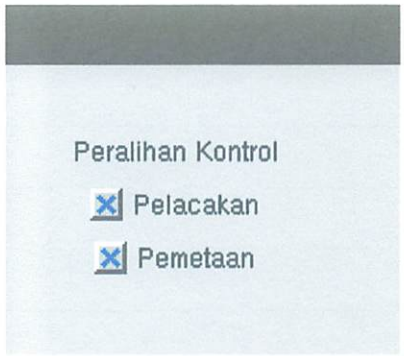
4.8.1 Peralihan Grafis

Berisi tiga buah pilihan untuk melihat perubahan grafis pada *window* penampil peta:

Tampilkan Lintasan, yaitu untuk menampilakn hasil lintasan/trajektori.

Tampilkan 2D *Deskriptor*, yaitu untuk menampilkan hasil *patch* pada awal mula program dijalankan, sekaligus untuk mengetahui apakah *patch* benar atau tidak.

Inisialisasi Tampilan, yaitu untuk menampilkan inisialisasi target



Gambar 4.21 Tampilan Pilihan Peralihan Kontrol

4.8.2 Peralihan Kontrol

Pelacakan yaitu hanya akan melacak fitur.

Pemetaan yaitu hanya akan memetakan fitur.

Gambar 4.22 Tampilan Pilihan Aksi Kontrol

4.8.3 Aksi kontrol

Inisialisasi Fitur Manual, yaitu untuk menginisialisasikan fitur secara manual.

Inisialisasi Fitur Secara Otomatis, yaitu untuk menginisialisasikan fitur secara otomatis.

Simpan *Patch*, yaitu untuk menyimpan *Patch.*

Cetak Lokasi Robot, yaitu untuk menyimpan lokasi robot yang telah dilalui.

Berhenti, yaitu untuk menutup program.

Gambar 4.23 Tampilan Pilihan Main Controls

4.8.4 Main Controls

Continous yaitu untuk melanjutkan gambar secara *sequence* dengan *delay* 200 ms.

Next yaitu untuk menjalankan gambar *sequence step by step* tiap diklik.

Stop yaitu untuk menghentikan gambar yanng berjalan terus menerus.

BAB V
PENUTUP

5.1 Kesimpulan

Setelah dilakukan perancangan, pengujian dan analisa sistem, maka dapat disimpulkan beberapa hal yang dapat digunakan untuk perbaikan dan pengembangan selanjutnya, yaitu :

1. Program hanya dapat mendeteksi fitur pada lingkungan yang kaya dengan gradasi warna dan memiliki cahaya cukup. Dari hasil pengujian diketahui bahwa lingkungan dengan cahaya kurang (gelap) membuat gambar hasil tangkapan kamera menjadi susah diolah, ditandai dengan tidak ditemukannya fitur oleh detektor.
2. Menjalankan keseluruhan program membutuhkan spesifikasi komputer yang tinggi karena berhubungan dengan pengolahan gambar yang cepat.
3. Dengan menggunakan RSync dan Cron memungkinkan untuk terus-menerus mengirim gambar sebagai bahan pembuat peta.
4. VSLAM dapat diterapkan pada sistem *embedded* dengan resolusi waktu *millisecond.* Dari hasil pengujian, diperoleh waktu pengiriman optimal pada 86 ms, sedangkan untuk mengolah gambar menjadi peta dibutuhkan waktu 200 ms.
5. Peta yang berupa trajektori menandakan implementasi VSLAM berhasil dilakukan pada sistem *embedded* dengan input dari objek bergerak berupa robot DDMR.

5.2 Saran

Pada perancangan dan pembuatan tugas akhir ini tak lepas dari kekurangan-kekurangan baik dari perancangan sistem maupun peralatan yang digunakan, untuk itu agar sistem dapat menjadi lebih baik maka diperlukan beberapa perbaikan dan penyempurnaan, yaitu :

1. Metode inisialisasi fitur *patch* sebaiknya ditingkatkan menjadi fitur tepi dan garis.

2. Robot sebaiknya ditambahkan sensor *odometry* sebagai penghitung jarak. Dengan dapat dihitungnya jarak tersebut, diharapkan skala asli robot dengan skala model pada komputer dapat ditentukan nilainya.
3. Untuk menerapkan VSLAM secara keseluruhan, sebaiknya dibuat program yang mampu merespon hasil fitur dari komputer untuk memberikan perintah bergerak kepada robot.
4. Trajektori peta ditingkatkan menjadi bentuk 3 Dimensi.

DAFTAR PUSTAKA

Anonim. (2014). *Raspberry Wiki Page (Rpi Hardware).* URL: http://elinux.org/RPi_Hardware. Diakses pada 10 September 2014.

Anonim. (2014). *Raspberry Wiki Page (Rpi Camera Module).* URL: http://elinux.org/Rpi_Camera_Module. Diakses pada 10 September 2014.

Anonim. *Rsync Features.* URL: https://rsync.samba.org/features.html. Diakses pada 10 Januari 2015.

Davison, Andrew J. (2003). *Real-Time Simultaneous Localisation and Mapping With A Single Camera.*

Davison, A. J., Reid, I. D., Molton N. D., dan Stasse, O. (2007). *Monoslam: Realtime Single Camera Slam.*

Hentzen, Whil. (2004). *Cron Explained* . Hentzenwerke Publishing, Inc.

Pitowarno, Endra. (2006). *Robotika Disain, Kontrol, dan Kecerdasan Buatan.* Andi : Yogyakarta.

Riisgaard, S. dan Blas, M. R. *Slam for Dummies.*

Salvi, Andrea (2013). Tesis: *Using Simultaneous Localization And Mapping Techniques in Robogames.*

Self, Smith, dan Cheeseman *(1990). Estimating Uncertain Spatial Relationships In Robotics.*

Septian, Cosmas Eric. (2014). Skripsi: *Rancang Bangun Differential Drive Mobile Robot untuk Penjejak Dinding dan Penghindar Halangan dengan Navigasi Sensor Ultrasonik dan Modul Kamera Raspberry Pi Menggunakan Metode Kendali Logika Fuzzy.* ITN Malang.

Tridgell, Andrew. (1999). Phd Tesis*: Efficient Algorithms for Sorting and Synchronization.* Australian National University.

Whyte, Hugh Durrant dan Leonard, John J. (*1991*). *Mobile robot localization by tracking geometric beacons.*

HALAMAN LAMPIRAN

LAMPIRAN

Lampiran 1. Langkah-langkah Instalasi MonoSLAMGlow

1. Install developtment tools pada sistem operasi Ubuntu 14.04

```
$ sudo apt-get install build-essential
```

2. Install OpenGL dan OpenGLUT

   Sebelum menginstal kedua library di atas, pastikan telah memiliki gcc compiler yang telah terinstal terlebih dahulu.

```
$ sudo apt-get install g++
$ sudo apt-get install freeglut3 freeglut3-dev libglut3
libglut3-dev libglew1.5 libglew1.5-dev libglu1-mesa libglu1-
mesa-dev libgl1-mesa-glx libgl1-mesa-dev mesa-common-dev
```

3. Install Pthread

```
$ sudo apt-get install libpthread-stubs0 libpthread-stubs0-dev
```

4. Install libraw1394, libraw1394-dev, libdc1394 dan libdc1394-dev

```
$ sudo apt-get install libraw1394-11 libraw1394-dev
libraw1394-doc
$ sudo apt-get install libdc1394-22-dbg libdc1394-utils
libdc1394-22 libdc1394-22-dev libdc1394-22-doc
```

5. Install libjpeg62-dev dan Xforms Toolkit menggunakan Synaptic Package Manager, lalu ketikkan perintah di bawah:

```
$ sudo apt-get install libxpm-dev
$ sudo apt-get install libopenjpeg-dev

$ cd xforms-1.0.93sp1
$ ./configure
$ make
$ sudo make install
```

6. Download SceneLib
7. Ekstrak SceneLib

Ekstrak hasil download pada suatu folder yang sama, karena masing-masing komponen akan saling bersangkutan, sehingga akan berjalan apabila diletakkan pada folder yang sama, untuk: glow_104, MonoSLAMGlow, TestSeqMonoSLAM, SceneLib and VW34.

```
$ sudo tar xvfz vw34.tar.gz
$ sudo tar xvfz scenelib.tar.gz
$ sudo tar xvfz glow_104.tar.gz
$ sudo tar xvfz monoslamglow.tar.gz
$ sudo tar xvfz testseqmonoslam.tar.gz
```

8. Compile GLOW Toolkit

compile GLOW Toolkit menggunakan kode di bawah:

```
$ cd glow_104
$ cd glow_src
$ sudo make
$ sudo ln -s libglow.a.1.0.2 libglow.a
```

9. Modifikasi pada file 'VW34/configure.ac'

Berikan tanda komentar (#) untuk komponen 'X', 'XFORMS' dan 'Firewire'.

```
# X
#AC_CHECK_HEADERS(X11/Xlib.h X11/Xutil.h,
#         [libx=yes],
#                 [libx=no; AC_MSG_WARN(**** x libraries not found
**** libVWX.a will not be compiled)])
AM_CONDITIONAL(X, test x$libx = xyes)

# XFORMS
#AC_CHECK_HEADERS(forms.h,
#         [xforms=yes],
#                   [xforms=no; AC_MSG_WARN(**** xforms not found
**** libVWXForms.a will not be compiled)])
AM_CONDITIONAL(XFORMS, test x$xforms = xyes)

# Firewire
#AC_CHECK_LIB(dc1394_control, dc1394_dma_setup_capture,
#            [firewire=yes],
#       [firewire=no; AC_MSG_WARN(**** libdc1394 not found ****
libVWFirewire.a will not be compiled)],
#     [-lraw1394])
AM_CONDITIONAL(FIREWIRE, test x$firewire = xyes)
```

12. Tambahkan '#include <cstdio>'

Untuk menghindari error, tambahkan library '#include <cstdio>' di atas kode pada masing-masing file di bawah:

VW34/VWGL/Interface/slider.cpp

13. Tambahkan '#include <typeinfo>'

Untuk menghindari error, tambahkan library '#include <typeinfo>' di atas kode pada masing-masing file di bawah:

SceneLib/Scene/scene_single.cpp

14. Modifikasi pada file 'VW34/VW/GeomObjects/point3d.h'

Tambahkan baris 148 dan baris149 dengan kode di bawah:

```
};

 Point3D operator+(const Point3D &p1, const Point3D &p2);
 Point3D operator/(const Point3D &p1, const double a);

 typedef Point3D Vector3D;

}; // end namespace VW
```

15. Modifikasi pada file 'VW34/VW/GeomObjects/point2d.h'.

Tambahkan baris 134 dan baris 135 dengan kode seperti di bawah:

```
};

 Point2D operator+(const Point2D &p1, const Point2D &p2);
 Point2D operator/(const Point2D &p1, const double a);

 typedef Point2D Vector2D;

}; // end namespace VW
```

16. Modifikasi pada file 'VW34/VW/GeomObjects/lineseg2d.h'.

Tambahkan baris 155 dan baris 156 dengan kode seperti di bawah:

```
};

 std::ostream& operator<<(std::ostream& s, const LineSeg2D &ls);
 std::istream& operator>>(std::istream& s, LineSeg2D &ls);

}; // end namespace VW
```

17. Modifikasi pada file 'VW34/VW/GeomObjects/lineseg3d.h'.

Tambahkan baris 93, baris 94 dan baris 96 dengan kode seperti di bawah:

```
};

 double Norm2(const LineSeg3D &ls);
 double Norm(const LineSeg3D &ls);
 std::ostream& operator<<(std::ostream& s, const LineSeg3D &ls);
 std::istream& operator>>(std::istream& s, LineSeg3D &ls);

 // Other funcs
 VW::LineSeg3D   operator+(const   VW::LineSeg3D   &line,   const
VW::Point3D &point);
 VW::LineSeg3D   operator-(const   VW::LineSeg3D   &line,   const
VW::Point3D &point);

}; // end namespace VW
```

18. Modifikasi pada file 'VW34/VW/Sequencers/sequencerbase.h'.

Ubah pada baris 127 seperti di bawah:

```
 void CopyImage(ImType& image, unsigned int which_channel=0) { };
```

19. Modifikasi pada file 'VW34/VW/Improc/matchdata.h'.

Tambahkan baris 72 dan baris 73 dengan kode seperti di bawah:

```
std::ostream & operator << (std::ostream & s, const MatchData &
m);
 std::istream & operator >> (std::istream & s, MatchData & m);

 /** Sort into order of increasing match score. */
 void SortAscending(std::vector<MatchData>& list);
 /** Sort into order of decreasing match score. */
 void SortDescending(std::vector<MatchData>& list);
```

20. Compile VW34

Ketika selesai melakukan instalasi library dengan benar, maka tidak akan ada error pada saat compile library VW34.

```
$ cd VW34
$ ./bootstrap
$ ./configure
$ sudo make
$ sudo make install
```

21. Compile SceneLib

Ketika selesai melakukan instalasi library dengan benar, maka tidak akan ada error pada saat compile library SceneLib.

Lampiran 2. Program Komputer Supervisor: MonoSLAMGlow.cpp

```
/*  Program di bawah adalah pengembangan dari free software dengan judul
proyek, yaitu MonoSLAM: Real-Time Single Camera SLAM

   MonoSLAMGlow/monoslamglow.cpp
   Copyright (C) 2005 University of Oxford
   Andrew Davison
   ajd@robots.ox.ac.uk
   Scene home page: http://www.robots.ox.ac.uk/~ajd/Scene/

   This library is free software; you can redistribute it and/or
   modify it under the terms of the GNU Lesser General Public
   License as published by the Free Software Foundation; either
   version 2.1 of the License, or (at your option) any later version.

   This library is distributed in the hope that it will be useful,
   but WITHOUT ANY WARRANTY; without even the implied warranty of
   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
   See the GNU  Lesser General Public License for more details.

   You should have received a copy of the GNU Lesser General Public
   License along with this library; if not, write to the Free Software
   Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA  02111-1307  USA
*/

#include <unistd.h>
#include "MonoSLAM/robot.h"
#include "MonoSLAM/threeddraw.h"
#include "MonoSLAM/model_creators.h"
#include "monoslamglow.h"


#ifdef DEBUGDUMP
#undef DEBUGDUMP
#endif
#define DEBUGDUMP false




MonoSLAMGlow::MonoSLAMGlow(int argc, char *argv[])
  : delta_t(1.0/30.0),
   NUMBER_OF_FEATURES_TO_SELECT(10),
   NUMBER_OF_FEATURES_TO_KEEP_VISIBLE(12),
   MAX_FEATURES_TO_INIT_AT_ONCE(1),
   MIN_LAMBDA(0.5),
   MAX_LAMBDA(5.0),
```

```
   NUMBER_OF_PARTICLES(100),
   STANDARD_DEVIATION_DEPTH_RATIO(0.3),
   MIN_NUMBER_OF_PARTICLES(20),
   PRUNE_PROBABILITY_THRESHOLD(0.05),

ERASE_PARTIALLY_INIT_FEATURE_AFTER_THIS_MANY_ATTEMPTS(
10)
{
 // Set up the main MonoSLAM class object
 SetUpMonoSLAM();

 // Reset counters
 output_tracked_image_number = 0;
 output_raw_image_number = 0;

 // Initialise GlowWindow
 // Set big size because this will be reshaped later
 Init("MonoSLAMGlow",
     GlowWindow::autoPosition,
     GlowWindow::autoPosition,
     1000, 1000, Glow::rgbBuffer, Glow::noEvents);

 SetUpButtons();

 SetUp3DDisplays();

 SetUpImageGrabber();

 Reshape(controlpanel1->Width() + controlpanel2->Width() +
         threedtool->Width(), threedtool->Height() +
         image_threedtool->Height());
}

void MonoSLAMGlow::SetUpMonoSLAM()
{
 // Use standard model creators from MonoSLAM/model_creators.h here
 MonoSLAM_Motion_Model_Creator mm_creator;
 MonoSLAM_Feature_Measurement_Model_Creator fmm_creator;

 monoslaminterface =
  new MonoSLAMInterface("monoslam_state.ini",
                        &mm_creator,
                        &fmm_creator,
                        NULL, // no internal measurement models used here
                        NUMBER_OF_FEATURES_TO_SELECT,
                        NUMBER_OF_FEATURES_TO_KEEP_VISIBLE,
                        MAX_FEATURES_TO_INIT_AT_ONCE,
                        MIN_LAMBDA,
```

```
                        MAX_LAMBDA,
                        NUMBER_OF_PARTICLES,
                        STANDARD_DEVIATION_DEPTH_RATIO,
                        MIN_NUMBER_OF_PARTICLES,
                        PRUNE_PROBABILITY_THRESHOLD,

ERASE_PARTIALLY_INIT_FEATURE_AFTER_THIS_MANY_ATTEMPTS);
}

void MonoSLAMGlow::SetUpButtons()
{
 // Left column of buttons
 controlpanel1 = new GlowQuickPaletteSubwindow(this, 0, 0, 200, 400,
                                        GlowQuickPalette::vertical,
                                        GlowQuickPalette::alignCenter,
                                        10, 10, 10);

 // Buttons which toggle aspects of the graphical displays call
 // the display request redraw functions to update graphics immediately
        controlpanel1->AddLabel("");
 controlpanel1->AddLabel("Peralihan Grafis");



 display_trajectory_button =
  controlpanel1->AddCheckBox("Tampilkan Lintasan",
                        GlowCheckBoxWidget::on, this);

 display_2d_descriptors_button =
  controlpanel1->AddCheckBox("Tampilkan 2D Descriptors",
                        GlowCheckBoxWidget::on, this);

 display_2d_searches_button =
  controlpanel1->AddCheckBox("Tampilkan Daerah 2D",
                        GlowCheckBoxWidget::on, this);


 display_initialisation_button =
  controlpanel1->AddCheckBox("Inisialisasi Tampilan",
                        GlowCheckBoxWidget::on, this);

        controlpanel1->AddLabel("");
        controlpanel1->AddLabel("");
        controlpanel1->AddLabel("");
        controlpanel1->AddLabel("");
        controlpanel1->AddLabel("");
        controlpanel1->AddLabel("");
        controlpanel1->AddLabel("");
```

```
enable_mapping_button =
  controlpanel2->AddCheckBox("Pemetaan",
                             GlowCheckBoxWidget::on, this);



// Standard buttons to call action functions
controlpanel2->AddLabel("");
        controlpanel2->AddLabel("");
        controlpanel2->AddLabel("");
controlpanel2->AddLabel("Aksi Kontrol");

init_feature_button =
  controlpanel2->AddPushButton("Inisialisasi Fitur Manual", this);

auto_init_feature_button =
  controlpanel2->AddPushButton("Inisialisasi  Fitur Otomatis", this);

save_patch_button=
   controlpanel2->AddPushButton("Simpan Patch", this);//Save Patch

print_robot_state_button =
  controlpanel2->AddPushButton("Cetak Lokasi Robot", this);

quit_button =
  controlpanel2->AddPushButton("Berhenti", this);

// delete_feature_button =
  // controlpanel2->AddPushButton("", this);//Delete Feature


        controlpanel2->AddLabel("");
        controlpanel2->AddLabel("");
        controlpanel2->AddLabel("");
        controlpanel2->AddLabel("");
        controlpanel2->AddLabel("");
        controlpanel2->AddLabel("");
        controlpanel2->AddLabel("");
        controlpanel2->AddLabel("");
        controlpanel2->AddLabel("");

output_tracked_images_button =
  controlpanel2->AddCheckBox("", //Output Tracked Images
                             GlowCheckBoxWidget::off, this);

output_raw_images_button =
  controlpanel2->AddCheckBox("", //Output Raw Images
                             GlowCheckBoxWidget::off, this);
```

```
 controlpanel2->Pack(true);
}

void MonoSLAMGlow::SetUp3DDisplays()
{
 // Set display virtual camera parameters to match those of the real camera
 // being used in MonoSLAM
 const Partially_Initialised_Feature_Measurement_Model *default_pifmm =
  monoslaminterface->GetDefaultFeatureTypeForInitialisation();
 const                          Line_Init_Wide_Point_Feature_Measurement_Model
*default_wide_pifmm =
  (const Line_Init_Wide_Point_Feature_Measurement_Model *) default_pifmm;
 Graphics_Fku = default_wide_pifmm->get_camera().Fku();
 Graphics_Fkv = default_wide_pifmm->get_camera().Fkv();
 Graphics_U0 = default_wide_pifmm->get_camera().U0();
 Graphics_V0 = default_wide_pifmm->get_camera().V0();
 Graphics_Kd1 = default_wide_pifmm->get_camera().Kd1();

 // First display for external 3D view
 threedtool = new ThreeDToolGlowWidget(this,
                    controlpanel1->Width() + controlpanel2->Width(), 0,
                                    monoslaminterface->GetCameraWidth(),
                                    monoslaminterface->GetCameraHeight());
 threedtool->DrawEvent.Attach(this, &MonoSLAMGlow::Draw3D);
 threedtool->ProcessHitsEvent.Attach(this, &MonoSLAMGlow::ProcessHits);
 threedtool->SetCameraParameters(Graphics_Fku, Graphics_Fkv,
                                 Graphics_U0, Graphics_V0);

 // Set start position for GL camera in 3D tool
 // This is x, y, z position
 VNL::VectorFixed<3,double> rthreed (0.0, 0.2, -1.5);
 // This is camera orientation in my normal coordinate system
 // (z forward, y up, x left)
 VW::Quaternion qWRthreed (0.0, 0.0, 0.0, 1.0);
 // We need to adjust by this rotation to fit GL coordinate frame
 // (z backward, y up, x right)
 // So rotate by pi about y axis
 VW::Quaternion qROthreed (0.0, 1.0, 0.0, 0.0);
 VW::Quaternion qWOthreed = qWRthreed * qROthreed;
 threedtool->SetCameraPositionOrientation(rthreed, qWOthreed);

 // Second 3D display for images and augmented reality
 image_threedtool = new ThreeDToolGlowWidget(this,
                    controlpanel1->Width() + controlpanel2->Width(),
                                    threedtool->Height(),
```

```
                                        monoslaminterface->GetCameraWidth(),
                                      monoslaminterface->GetCameraHeight());
  image_threedtool->DrawEvent.Attach(this,
&MonoSLAMGlow::ImageDraw3D);
  image_threedtool->ProcessHitsEvent.Attach(this,
                                          &MonoSLAMGlow::ImageProcessHits);
  image_threedtool->SetCameraParameters(Graphics_Fku, Graphics_Fkv,
                                        Graphics_U0, Graphics_V0);

  // Set up initial state of virtual camera for image display to match
  // state vector
  const Scene_Single *scene = monoslaminterface->GetScene();
  scene->get_motion_model()->func_xp(scene->get_xv());
  ThreeD_Motion_Model *threed_motion_model =
   (ThreeD_Motion_Model *) scene->get_motion_model();
  VW::Vector3D r = threed_motion_model->get_rRES();
  VW::Quaternion qWR = threed_motion_model->get_qRES();
  // q is qWR between world frame and Scene robot frame
  // We need to adjust by this rotation to fit GL coordinate frame
  // (z backward, y up, x right)
  // So rotate by pi about y axis
  VW::Quaternion qRO (0.0, 1.0, 0.0, 0.0);
  VW::Quaternion qWO = qWR * qRO;
  image_threedtool->SetCameraPositionOrientation(r, qWO);
}

void MonoSLAMGlow::SetUpImageGrabber()
{
  grabbed_image = new VW::ImageMono<unsigned char>
                          (monoslaminterface->GetCameraWidth(),
                                  monoslaminterface->GetCameraHeight());

#ifdef _REALTIME_
  realtimeflag = true;

  // Firewire camera
  grabber = new VW::SequencerFirewireGlow();
  grabber->SequencerFirewire::Init();
  if (grabber->GetNumberOfChannels()==0) {
    cerr << "No cameras!" << endl;
    exit(0);
  } else {
    assert(grabber->GetNumberOfChannels() == 1);
    // We might want to change some camera settings (zoom, focus etc...)
    VW::FirewireCamera* d=grabber->GetCamera(0);
    d->SetImageMode(VW::FirewireParams::M_320x240_YUV422);
    d->SetFramerate(VW::FirewireParams::FPS_30);
    cout << "Camera 0:\n"<<(*d)<<endl;
```

```
  }
#else
  realtimeflag = false;

  // And create sequencer window
  // grabber = new SequencerFileGlow("rawoutput0000.pgm");
  //system("cls");
  grabber                                    =                                    new
SequencerFileGlow("../TestSeqMonoSLAM/rawoutput0000.pgm");
  grabber->SequencerBase::Init();
  // Set delay between frames on "Continuous" button (milliseconds)
  // Set this high if tracked images are to be saved to disk
  grabber->SetDelay(200);
#endif

  grabber->NewFrameEvent.Attach(this,
&MonoSLAMGlow::HandleNewFrameGlow);
}

void MonoSLAMGlow::OnReshape(int width, int height)
{
  cerr << "Reshape w = " << width << " height = " << height << endl;
  cerr << "Minimum w = " << controlpanel1->Width() + controlpanel2->Width()
+
                  int(monoslaminterface->GetCameraWidth()) << " height = " << 2
* int(monoslaminterface->GetCameraHeight())
      << endl;

  // When we reshape the window we'll keep the button panels the same
  // width as usual; and we won't let the 3D displays get smaller than
  // their original size
  if (width < controlpanel1->Width() + controlpanel2->Width() +
     int(monoslaminterface->GetCameraWidth()) ||
     height < 2 * int(monoslaminterface->GetCameraHeight())) {
   int neww = max(width, controlpanel1->Width() + controlpanel2->Width() +
                  int(monoslaminterface->GetCameraWidth()));
   int newh = max(height, 2 * int(monoslaminterface->GetCameraHeight()));
   cerr << "Trying to reshape w = " << neww << " h = " << newh << endl;
   // This resizes the window (and then this function will be called again)
   Reshape(neww, newh);
  }
  else {
   controlpanel1->Reshape(controlpanel1->Width(), height);
   controlpanel1->Refresh();
   controlpanel2->Reshape(controlpanel2->Width(), height);
   controlpanel2->Refresh();

   // MaxWidth of the 3D displays is now defined
```

```
    int new3Dmaxwidth = width - controlpanel1->Width()
                        - controlpanel2->Width();
    int new3Dmaxheight = height / 2;

    // But we want to keep the 3D displays of a constant aspect ratio
    double width_height_ratio = double(monoslaminterface->GetCameraWidth())
     / double(monoslaminterface->GetCameraHeight());

    int new3Dwidth, new3Dheight;
    if (new3Dmaxwidth / width_height_ratio > new3Dmaxheight) {
     // Wide window; size limited by height
     new3Dwidth = int(width_height_ratio * double(new3Dmaxheight) + 0.5);
     new3Dheight = new3Dmaxheight;
    }
    else {
     // Tall window; size limited by width
     new3Dwidth = new3Dmaxwidth;
     new3Dheight = int(double(new3Dmaxwidth) / width_height_ratio + 0.5);
    }
    threedtool->Move(controlpanel1->Width() + controlpanel2->Width(), 0);
    threedtool->Reshape(new3Dwidth, new3Dheight);

    image_threedtool->Move(controlpanel1->Width() + controlpanel2->Width(),
                           new3Dheight);
    image_threedtool->Reshape(new3Dwidth, new3Dheight);

    threedtool->Refresh();
    image_threedtool->Refresh();
   }

  // Update the viewport
  ::glViewport(0, 0, width, height);
}


void MonoSLAMGlow::OnMessage(const GlowPushButtonMessage& message)
{
 if(message.widget == init_feature_button) {
  InitialiseFeatureGlow();
 }
 else if(message.widget == auto_init_feature_button) {
  AutoInitialiseFeatureGlow();
 }
 else if(message.widget == print_robot_state_button) {
  //PrintRobotStateGlow();
 }
 else if(message.widget == delete_feature_button) {
  //DeleteFeatureGlow();
```

```
  }
  else if(message.widget == save_patch_button) {
   SavePatchGlow();
  }
  else if(message.widget == quit_button) {
   Quit();
  }
  else
    assert(0);
}

void MonoSLAMGlow::OnMessage(const GlowCheckBoxMessage& message)
{
  if(message.widget == rectify_image_display_button ||
     message.widget == display_trajectory_button ||
     //message.widget == display_3d_features_button ||
     //message.widget == display_3d_feature_uncertainties_button ||
     message.widget == display_2d_descriptors_button ||
     message.widget == display_2d_searches_button ||
     message.widget == display_initialisation_button) {
    // These buttons have their state queries by the graphics routine
    // so just request redraw
    GraphicalButtonToggled(true);
  }
  else if (message.widget == toggle_tracking_button ||
           message.widget == enable_mapping_button ||
           message.widget == output_tracked_images_button ||
           message.widget == output_raw_images_button) {
    // These buttons don't need any action because their states are
    // queried by tracking function
  }
  else
    assert(0);
}




// Drawing function which will be registered with 3D tool
void MonoSLAMGlow::Draw3D(bool select_flag)
{
  mutex.Lock();
  // cout << "Timer on entering Draw3D: " << monoslaminterface->GetTimer1()
  //   << endl;

  if (DEBUGDUMP) cout << "Drawing function for 3D display." << endl;
```

```
 External3DDraw(monoslaminterface->GetScene(),
                *threedtool,
                trajectory_store,
                1,
                0,
                display_trajectory_button->GetState(),
                display_3d_features_button->GetState(),
                display_3d_feature_uncertainties_button->GetState(),
                display_3d_features_button->GetState(),
                0);

 // cout << "Timer on leaving Draw3D: " << monoslaminterface->GetTimer1()
 //   << endl;
 mutex.Unlock();
}


void MonoSLAMGlow::ProcessHits(int selected_item,
                               double camera_x, double camera_y)
{
 // Here selected_item will be the label of a feature plus one if the
 // user has clicked on a mapped feature; 0 if not
 if (selected_item > 0) {
   if (Scene_Single::STATUSDUMP)
     cout << "Toggled feature with label: " << selected_item - 1
          << "; hit Delete to delete." << endl;

   monoslaminterface->GetSceneNoConst()->
                      toggle_feature_lab(selected_item - 1);
   monoslaminterface->GetSceneNoConst()->
                        mark_feature_by_lab(selected_item - 1);
 }

 threedtool->RequestDraw();
 image_threedtool->RequestDraw();
}

void MonoSLAMGlow::ImageDraw3D(bool select_flag)
{
 mutex.Lock();
 // cout << "Timer on entering ImageDraw3D: "
 // << monoslaminterface->GetTimer1() << endl;

 if (DEBUGDUMP) cout << "Drawing function for Image display." << endl;

 if (rectify_image_display_button->GetState()) {
   RectifiedInternal3DDraw(monoslaminterface->GetScene(),
                           *image_threedtool,
```

```
                        trajectory_store,
                        grabbed_image,
                        Graphics_U0,
                        Graphics_V0,
                        Graphics_Kd1,
                        0,
                        display_trajectory_button->GetState(),
                        display_3d_features_button->GetState(),
                display_3d_feature_uncertainties_button->GetState(),
                        display_3d_features_button->GetState(),
                        0,
                        1,
                        1);
 }
 else {
  RawInternal3DDraw(monoslaminterface->GetScene(),
                        *image_threedtool,
                        grabbed_image,
       monoslaminterface->GetSceneNoConst()->get_feature_init_info_vector(),
                        display_2d_descriptors_button->GetState(),
                        display_2d_searches_button->GetState(),
                        display_initialisation_button->GetState(),
                        1,
            monoslaminterface->GetInitFeatureSearchRegionDefinedFlag(),
                        BOXSIZE,
                        monoslaminterface->GetLocationSelectedFlag(),
                        monoslaminterface->GetUU(),
                        monoslaminterface->GetVV(),
                        monoslaminterface->GetInitFeatureSearchUStart(),
                        monoslaminterface->GetInitFeatureSearchVStart(),
                        monoslaminterface->GetInitFeatureSearchUFinish(),
                        monoslaminterface->GetInitFeatureSearchVFinish());
 }

 // cout << "Timer on leaving ImageDraw3D: " << monoslaminterface-
>GetTimer1()
 //    << endl;
 mutex.Unlock();
}

void MonoSLAMGlow::ImageProcessHits(int selected_item,
                                    double camera_x, double camera_y)
{
 // Here selected_item will be the label of a feature plus one if the
 // user has clicked on a mapped feature; 0 if not
 if (selected_item > 0) {
  if (Scene_Single::STATUSDUMP)
   cout << "Toggled feature with label: " << selected_item - 1
```

```
        << "; hit Delete to delete." << endl;

   monoslaminterface->GetSceneNoConst()->
     toggle_feature_lab(selected_item - 1);
   monoslaminterface->GetSceneNoConst()->
     mark_feature_by_lab(selected_item - 1);
  }

  monoslaminterface->GetRobotNoConst()->
   set_image_selection(int (camera_x + 0.5), int(camera_y + 0.5));
  if (Scene_Single::STATUSDUMP)
   cout << "Selected image position: " << int (camera_x + 0.5) << ", "
        << int(camera_y + 0.5) << endl;

  threedtool->RequestDraw();
  image_threedtool->RequestDraw();
}


void MonoSLAMGlow::SaveRawImage()
{
  char filename[100];
  sprintf(filename, "rawoutput%04d.pgm", output_raw_image_number++);
  grabbed_image->WriteImage(filename);
  sync();
  if (Scene_Single::STATUSDUMP)
   cout << "Written raw image " << filename << endl;
}

void MonoSLAMGlow::SaveTrackedImage()
{
  // Output one composite of image and 3D tool
  VW::ImageRGB<unsigned char>
   output_image(monoslaminterface->GetCameraWidth(),
                monoslaminterface->GetCameraHeight());
  image_threedtool->GetImage(output_image);
  VW::ImageRGB<unsigned char>
   output3d_image(monoslaminterface->GetCameraWidth(),
                  monoslaminterface->GetCameraHeight());
  threedtool->GetImage(output3d_image);

  VW::ImageRGB<unsigned char>
   outputcomposite_image(2 * monoslaminterface->GetCameraWidth(),
                         monoslaminterface->GetCameraHeight());
  // Fill composite
  for (unsigned int row = 0;
     row < monoslaminterface->GetCameraHeight(); row++) {
   for (unsigned int im1col = 0;
```

```
        im1col < monoslaminterface->GetCameraWidth(); im1col++) {
    outputcomposite_image.SetPixel(im1col, row,
                                   output_image.GetPixel(im1col, row));
   }
   for (int unsigned im2col = 0;
        im2col < monoslaminterface->GetCameraWidth(); im2col++) {
    outputcomposite_image.SetPixel(
                          im2col + monoslaminterface->GetCameraWidth(), row,
                                   output3d_image.GetPixel(im2col, row));
   }
  }
  char filename[100];
  sprintf(filename, "composite%04d.ppm", output_tracked_image_number);
  outputcomposite_image.WriteImage(filename);
  if (Scene_Single::STATUSDUMP)
   cout << "Written image " << filename << endl;
  output_tracked_image_number++;

  // Force write to disk to avoid build-up and later dropped frames
  sync();
}



void MonoSLAMGlow::HandleNewFrameGlow(int frame_number)
{
  mutex.Lock();
  monoslaminterface->ResetTimer1();
  if (Scene_Single::STATUSDUMP)
   cout << "Frame acquired: absolute time = "
        << monoslaminterface->GetTimer0()
        << endl;

  grabber->CopyImage(*grabbed_image, 0);
  monoslaminterface->GetRobotNoConst()->load_new_image(grabbed_image);

  if (Scene_Single::STATUSDUMP)
   cout << "Time after copying image: " << monoslaminterface->GetTimer1()
        << endl;

  if(toggle_tracking_button->GetState()) {
   // One tracking step (class MonoSLAM)
   if (!monoslaminterface->GoOneStep(grabbed_image,
                                     delta_t, enable_mapping_button->GetState())) {
    // If GoOneStep returns false, tracking has gone nasty so stop
    toggle_tracking_button->SetState(GlowCheckBoxWidget::off);
   }
  }
```

```
  if (Scene_Single::STATUSDUMP)
    cout << "Time after all vision: " << monoslaminterface->GetTimer1()
         << endl;

  // After tracking, update the displays
  UpdateDisplayParameters();

  mutex.Unlock();

  if (output_raw_images_button->GetState()) {
    SaveRawImage();
  }
  else {
    // Don't display if we're trying to save an image sequence

    // In real-time mode, only redraw if we have time
    if (!realtimeflag ||
        // Heuristic: we need around 15ms to do graphics
        monoslaminterface->GetTimer1().GetAsSeconds() < delta_t - 0.015) {
      threedtool->RequestDraw();
      image_threedtool->RequestDraw();
    }
    else
      if (Scene_Single::STATUSDUMP)
        cout << "No time to update displays." << endl;
  }

  if (output_tracked_images_button->GetState()) {
    SaveTrackedImage();
  }
}

void MonoSLAMGlow::UpdateDisplayParameters()
{
  // Put the camera in the right place in the image display
  monoslaminterface->GetScene()->get_motion_model()->
    func_xp(monoslaminterface->GetScene()->get_xv());
  ThreeD_Motion_Model *threed_motion_model =
    (ThreeD_Motion_Model *) monoslaminterface->GetScene()-
>get_motion_model();
  VW::Vector3D r = threed_motion_model->get_rRES();
  VW::Quaternion qWR = threed_motion_model->get_qRES();

  if (display_trajectory_button->GetState()) {
    trajectory_store.push_back(r.GetVNL3());
    if (trajectory_store.size() > 800) {
      trajectory_store.erase(trajectory_store.begin());
    }
```

```
  }

  // q is qWR between world frame and Scene robot frame
  // What we need to plot though uses GL object frame O
  // Know qRO: pi rotation about y axis
  // qWO = qWR * qRO
  VW::Quaternion qRO (0.0, 1.0, 0.0, 0.0);
  VW::Quaternion qWO = qWR * qRO;

  image_threedtool->SetCameraPositionOrientation(r, qWO);
}

void MonoSLAMGlow::InitialiseFeatureGlow()
{
  monoslaminterface->InitialiseFeature();

  threedtool->RequestDraw();
  image_threedtool->RequestDraw();
}

void MonoSLAMGlow::AutoInitialiseFeatureGlow()
{
  // Zero u here
  VNL::Vector<double> u(3);
  u.Fill(0.0);

  monoslaminterface->AutoInitialiseFeature(u, delta_t);
}

void MonoSLAMGlow::DeleteFeatureGlow()
{
  monoslaminterface->DeleteFeature();

  threedtool->RequestDraw();
  image_threedtool->RequestDraw();
}

void MonoSLAMGlow::SavePatchGlow()
{
  monoslaminterface->SavePatch();
}

void MonoSLAMGlow::Quit()
{
  exit(0);
}

void MonoSLAMGlow::PrintRobotStateGlow()
```

Chapter 1: RPi Hardware Basic Setup

Typical Hardware You Will Need

While the RPi can be used without any additional hardware (except perhaps a power supply of some kind), it won't be much use as a general computer. As with any normal PC, it is likely you will need some additional hardware.

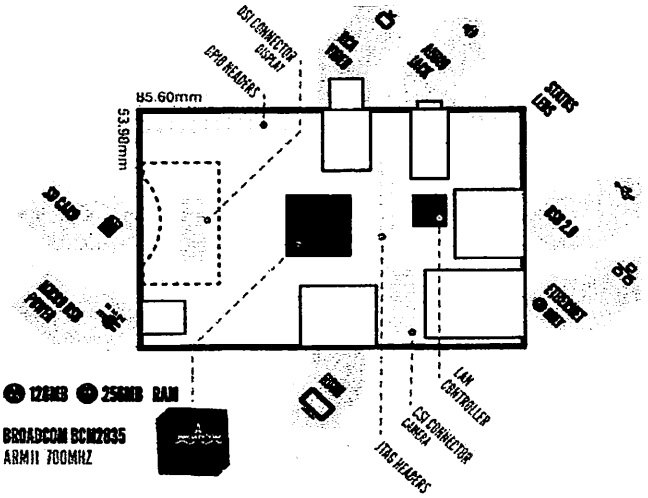
The following are more or less essential:

- Raspberry Pi board
- Prepared Operating System SD Card
- USB keyboard
- Display (with HDMI, DVI, Composite or SCART input)
- Power Supply
- Cables

Highly suggested extras include:

- USB mouse
- Internet connectivity - a USB WiFi adaptor (Model A/B) or a LAN cable (Model B)
- Powered USB Hub
- Case

Connecting Together



You can use the diagram to connect everything together, or use the following instructions:

1. Plug the preloaded SD Card into the Pi.
2. Plug the USB keyboard and mouse into the Pi, perhaps via a USB Hub. Connect the Hub to power, if necessary.

you will need a cable which adapts from 3.5mm to double (red and white) RCA connectors.

Note: There is no VGA output available, so older VGA monitors will require an expensive adaptor.

Using an HDMI to DVI-D (digital) adaptor plus a DVI to VGA adaptor will not work. HDMI does not supply the DVI-A (analogue) needed to convert to VGA - converting an HDMI or DVI-D source to VGA (or component) needs an active converter. (It can work out cheaper to buy a new monitor.) The lack of VGA has been acknowledged as a priority issue.

Power Supply

The unit uses a Micro USB connection to power itself (only the power pins are connected - so it will not transfer data over this connection). A standard modern phone charger with a micro-USB connector will do, but needs to produce at least 700mA at 5 volts. Check your power supply's ratings carefully. Suitable mains adaptors will be available from the RPi Shop and are recommended if you are unsure what to use.

You can use a range of other power sources (assuming they are able to provide enough current ~700mA):

- Computer USB Port or powered USB hub (will depend on power output)
- Special wall warts with USB ports
- Mobile Phone Backup Battery (will depend on power output) (in theory - needs confirmation)

To use the above, you'll need a USB A 'male' to USB micro 'male' cable - these are often shipped as data cables with MP3 players.

Cables

You will probably need a number of cables in order to connect your RPi up.

1. Micro-B USB Power Cable
2. HDMI-A or Composite cable, plus DVI adaptor or SCART adaptor if required, to connect your RPi to the Display/Monitor/TV of your choice.
3. Audio cable, this is not needed if you use a HDMI TV/monitor.
4. Ethernet/LAN Cable

Additional Peripherals

You may decide you want to use various other devices with your RPi, such as Flash Drives/Portable Hard Drives, Speakers etc.

Internet Connectivity

This may be an Ethernet/LAN cable (standard RJ45 connector) or a USB WiFi adaptor. The RPi ethernet port is auto-sensing which means that it may be connected to a router or directly to another computer (without the need for a crossover cable).

USB-Hub

In order to connect additional devices to the RPi, you may want to obtain a USB Hub, which will allow multiple devices to be used.

It is recommended that a **powered** hub is used - this will provide any additional power to the devices without affecting the RPi itself.

USB version 2.0 is recommended. USB version 1.1 is fine for keyboards and mice, but may not be fast enough for other accessories.

Case

Since the RPi is supplied without a case, it will be important to ensure that you do not use it in places where it will come into contact with conductive metal or liquids, unless suitably protected.

Expansion & Low Level Peripherals

If you plan on making use of the low level interfaces available on the RPi, then ensure you have suitable header pins for the GPIO (and if required JTAG) suitable for your needs.

Also if you have a particular low-level project in mind, then ensure you design in suitable protection circuits to keep your RPi safe.

Chapter 2: RPi Advanced Setup

Finding hardware and setting up

You'll need a preloaded SD card, USB keyboard, TV/Monitor (with HDMI/ DVI/ Composite/ SCART input), and power supply (USB charger or a USB port from a powered USB Hub or another computer).

You'll likely also want a USB mouse, a case, and a USB Hub (a necessity for Model A). A powered USB Hub will reduce the demand on the RPi. To connect to the Internet, you'll need either an Ethernet/LAN cable (Model B) or a USB WiFi adaptor (either model).

When setting up, it is advisable to connect the power after everything else is ready.

Serial connection

The Serial Port is a simple and uncomplicated method to connect to the Raspberry Pi. The communication depends on byte wise data transmission, is easy to setup and is generally available even before boot time.

First interaction with the board

Connect the serial cable to the COM port in the Raspberry Pi, and connect the other end to the COM port or USB Serial Adapter in the computer.

Serial Parameters

The following parameters are needed to connect to the Raspberry. All parameters except **Port_Name** and **Speed** are default values and may not need to be set.

- **Port_Name**: Linux automatically assigns different names for different types of serial connectors. Choose your option:
  - Standard Serial Port: ttyS0 ... ttySn
  - USB Serial Port Adapter: ttyUSB0 ... ttyUSBn
- **Speed**: 115200
- Bits: 8
- Parity: None
- Stop Bits: 1
- Flow Control: None

The Serial Port is generally usable by the users in the group **dialout**. To add oneself to the group **dialout** the the following command needs to be executed with **root** privileges:

```
sudoradd -G [dialout] your_name
```

- **Super Easy Way Using GNU Screen**

Enter the command below into a terminal window

```
screen /dev/Port_Name 115200
```

- **Super Easy Way Using Minicom**

Run minicom with the following parameters:

```
minicom -b 115200 -o -D /dev/Port_Name
```

- **GUI method with GtkTerm**

Start *GtkTerm*, select Configuration->Port and enter the values above in the labelled fields.

- **Windows Users**

Windows Users above Windows XP must download putty or a comparable terminal program. Users of XP and below can choose between using *putty* and *Hyperterminal.*

First Dialog

If you get the prompt below, you are connected to the Raspberry Pi shell!

```
prompt:~#
```

First command you might want try is "help":

```
prompt:~# /bin/ip
```

If you get some output, you are correctly connected to the Raspberry Pi! Congratulations!

SD card setup

Now we want to use an SD card to install some GNU/Linux distro in it and get more space for our stuff. You can use either an SD or SDHC card. In the latter case of course take care that your PC card reader also supports SDHC. Be aware that you are not dealing with an x86 processor, but instead a completely different architecture called ARM, so don't forget to install the ARM port for the distro you are planning to use.

Formatting the SD card via the mkcard.txt script

1. Download **mkcard.txt** .
2. $ chmod +x mkcard.txt

3. $ ./mkcard.txt /dev/sd*x*, where *x* is the letter of the card. You can find this by inserting your card and then running dmesg | tail. You should see the messages about the device being mounted in the log. Mine mounts as **sdc**.

Once run, your card should be formatted.

Formatting the SD card via fdisk "Expert mode"

First, lets clear the partition table:

```
-------------------------------------------------------------------------------
-----
$ sudo fdisk /dev/sdb

Command (m for help): o
Building a new DOS disklabel. Changes will remain in memory only,
until you decide to write them. After that, of course, the previous
content won't be recoverable.

Warning: invalid flag 0x0000 of partition table 4 will be corrected by
w(rite)
-------------------------------------------------------------------------------
-----
```

Print card info:

```
-------------------------------------------------------------------------------
-----
Command (m for help): p

Disk /dev/sdb: 128 MB, 128450560 bytes
....
-------------------------------------------------------------------------------
-----
```

Note card size in bytes. Needed later below.

Then go into "Expert mode":

```
-------------------------------------------------------------------------------
-----
Command (m for help): x
-------------------------------------------------------------------------------
-----
```

Now we want to set the geometry to 255 heads, 63 sectors and calculate the number of cylinders required for the particular SD/MMC card:

```
================================================================================
Expert command (m for help): h
Number of heads (1-256, default 4): 255

Expert command (m for help): s
Number of sectors (1-63, default 62): 63
Warning: setting sector offset for DOS compatiblity
================================================================================
```

NOTE: Be especially careful in the next step. First calculate the number of cylinders as follows:

- B = Card size in bytes (you got it before, in the second step when you printed the info out)
- C = Number of cylinders

```
C=B/255/63/512
```

When you get the number, you round it DOWN. Thus, if you got 108.8 you'll be using 108 cylinders.

```
================================================================================
Expert command (m for help): c
Number of cylinders (1-1048576, default 1011): 15
================================================================================
```

In this case 128MB card is used (reported as 128450560 bytes by fdisk above), thus 128450560 / 255 / 63 / 512 = 15.6 rounded down to 15 cylinders. Numbers there are 255 heads, 63 sectors, 512 bytes per sector.

So far so good, now we want to create two partitions. One for the boot image, one for our distro. Create the FAT32 partition for booting and transferring files from Windows. Mark it as bootable.

```
================================================================================
Expert command (m for help): r
Command (m for help): n
Command action
   e   extended
   p   primary partition (1-4)
p
Partition number (1-4): 1
```

```
First cylinder (1-245, default 1): (press Enter)
Using default value 1
Last cylinder or +size or +sizeM or +sizeK (1-245, default 245): +50

Command (m for help): t
Selected partition 1
Hex code (type L to list codes): c
Changed system type of partition 1 to c (W95 FAT32 (LBA))

Command (m for help): a
Partition number (1-4): 1
================================================================================
=====
```

Create the Linux partition for the root file system.

```
================================================================================
=====
Command (m for help): n
Command action
  e   extended
  p   primary partition (1-4)
p
Partition number (1-4): 2
First cylinder (52-245, default 52): (press Enter)
Using default value 52
Last cylinder or +size or +sizeM or +sizeK (52-245, default 245):(press
Enter)
Using default value 245
================================================================================
=====
```

Print and save the new partition records.

```
================================================================================
=====
Command (m for help): p

Disk /dev/sdc: 2021 MB, 2021654528 bytes
255 heads, 63 sectors/track, 245 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes

  Device Boot      Start         End      Blocks   Id  System
/dev/sdc1   *           1          51      409626    c  W95 FAT32 (LBA)
/dev/sdc2              52         245     1558305   83  Linux

Command (m for help): w
The partition table has been altered!

Calling ioctl() to re-read partition table.
```

```
WARNING: Re-reading the partition table failed with error 16: Device or
resource busy. The kernel still uses the old table. The new table will be
used at the next reboot.

WARNING: If you have created or modified any DOS 6.x partitions, please see
the fdisk manual page for additional information.
Syncing disks.
================================================================================
=====
```

Now we've got both partitions, next step is formatting them.

NOTE: If the partitions (/dev/sdc1 and /dev/sdc2) does not exist, you should unplug the card and plug it back in. Linux will now be able to detect the new partitions.

```
================================================================================
=====
$ sudo mkfs.msdos -F 32 /dev/sdc1 -n LABEL
mkfs.msdos 2.11 (12 Mar 2005)

$ sudo mkfs.ext3 /dev/sdc2
mke2fs 1.40-WIP (14-Nov-2006)
Filesystem label=
OS type: Linux
Block size=4096 (log=2)
Fragment size=4096 (log=2)
195072 inodes, 389576 blocks
19478 blocks (5.00%) reserved for the super user
First data block=0
Maximum filesystem blocks=402653184
12 block groups
32768 blocks per group, 32768 fragments per group
16256 inodes per group
Superblock backups stored on blocks:
        32768, 98304, 163840, 229376, 294912

Writing inode tables: done
Creating journal (8192 blocks): done
Writing superblocks and filesystem accounting information:
================================================================================
=====
```

All done!

NOTE: For convenience, you can add the -L option to the mkfs.ext3 command to assign a volume label to the new ext3 filesystem. If you do that, the new (automatic) mount point under /media when you insert that SD card into some Linux hosts will be based on that label. If there's no label, the new mount point will most likely be a long hex string, so assigning a label makes manual mounting on the host more convenient.

Setting up the boot partition

The boot partition must contain:

- bootcode.bin : 2nd stage bootloader, starts with SDRAM disabled
- loader.bin : 3rd stage bootloader, starts with SDRAM enabled
- start.elf: The GPU binary firmware image, provided by the foundation.
- kernel.img: The OS kernel to load on the ARM processor. Normally this is Linux - see instructions for compiling a kernel.
- cmdline.txt: Parameters passed to the kernel on boot.

Optional files:

- config.txt: A configuration file read by the GPU. Use this to override set the video mode, alter system clock speeds, voltages, etc.
- vlls directory: Additional GPU code, e.g. extra codec's. Not present in the initial release.

Additional files supplied by the foundation

These files are also present on the SD cards supplied by the foundation.

Additional kernels. Rename over kernel.img to use them (ensure you have a backup of the original kernel.img first!):

- kernel_emergency.img : kernel with busybox rootfs. You can use this to repair the main linux partition using e2fsck if the linux partition gets corrupted.

Additional GPU firmware images, rename over start.elf to use them:

- arm128_start.elf : 128M ARM, 128M GPU split (use this for heavy 3D work, possibly also required for some video decoding)
- arm192_start.elf : 192M ARM, 64M GPU split (this is the default)
- arm224_start.elf : 224M ARM, 32M GPU split (use this for Linux only with no 3D or video processing. It's enough for the 1080p frame buffer, but not much else)

Writing the image into the SDcard and finally booting GNU/Linux

The easiest way to do this is to use PiCard. It even saves you from some hassles explained above. You will need your SD card + reader and a Linux pc to use PiCard. After that, just plug the card into your Rpi.

Setting up the boot args

Wire up your Raspberry Pi and power it up

As explained in Chapter 1

SD Card Cloning/Backup

```
Note: Update these instructions if required once they've been tried
```

From windows you can copy the full SD-Card by using Win32DiskImager. Alternatively, you can use the following instructions;

```
Note:
Many built-in SD card readers do not work, so if you have problems
use an external SD-USB adapter for this.
```

Required Software Setup

- download a windows utility dd.exe from *http://www.chrysocome.net/dd*
- rename it windd.exe

(This executable can to write to your harddisk so exercise caution using it!)

- make a copy named dd-removable.exe

(That executable refuses to write to your hard disk as it is named dd-removable As long as you use dd-removable.exe you cannot lose your hard disk)

- Connect an SD card to the computer
- run "dd-removable –list"

Should give something like this:

```
rawwrite dd for windows version 0.6beta3.
Written by John Newbigin <jn@it.swin.edu.au>
This program is covered by terms of the GPL Version 2.

NT Block Device Objects
\\?\Device\Harddisk1\Partition0
link to \\?\Device\Harddisk1\DR8
Removable media other than floppy. Block size = 512
size is 4075290624 bytes
```

This "\\?\Device\Harddisk1\Partition0" is the part you need.

Reading an image from the SD Card

BEWARE: DO THIS WRONG AND YOU CAN LOSE YOUR HARDDISK!!!

Obviously, you can NOT use 'dd-removable' to read an image as that executable refuses to write to your hard disk (so extra care is required here as you use 'windd').

- To **read** an SD-card image from the SD-card use:

```
windd bs=1M if=\\?\Device\Harddisk1\Partition0 of=THE_IMAGE_READ -size
Your disk name ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
```

Copying an image to the SD Card

BEWARE: DO THIS WRONG AND YOU CAN LOSE YOUR HARDDISK!!!

- To **copy** an image named "THEIMAGE" to the SD-card do this:

```
dd-removable bs=1M if=THEIMAGE of=\\?\Device\Harddisk1\Partition0
                    Your disk name ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
```

Software Development/Proving

A supported platform for the Raspberry is Qt , which is already being worked on. C/C++ is supported through a gcc cross-compiling tool chain.

After compiling, using QEMU and a Linux VM would be one way of testing your apps. This also works on Windows. Search the forum for the readymade ARM images. The choice of programming languages, IDEs and other tools ON the R-Pi is only determined by:

- The operating system compatibility (at the moment the specific Linux distro used)
- The status of the respective ARM package repositories and their binary compatibility
- The possibility to build other software + its dependencies for the R-Pi from sources.

For more guides and projects involving the Raspberry Pi, see RPi Projects (http://elinux.org/RPi_Projects).

RaspiCam Documentation

July 2013

This document describes the use of the three Raspberry Pi camera applications as of July 2013.

There are three applications provided: `raspistill`, `raspivid` and `raspistillyuv`. Both `raspistill` and `raspistillyuv` are very similar and are intended for capturing images, while `raspivid` is for capturing video.

All the applications are command-line driven, written to take advantage of the mmal API which runs over OpenMAX. The mmal API provides an easier to use system than that presented by OpenMAX. Note that mmal is a Broadcom specific API used only on Videocore 4 systems.

The applications use up to four OpenMAX(mmal) components - `camera`, `preview`, `encoder` and `null_sink`. All applications use the camera component: `raspistill` uses the Image Encode component, `raspivid` uses the Video Encode component, and `raspistillyuv` does not use an encoder, and sends its YUV or RGB output direct from camera component to file.

The preview display is optional, but can be used full screen or directed to a specific rectangular area on the display. If preview is disabled, the `null_sink` component is used to 'absorb' the preview frames. It is necessary for the camera to produce preview frames even if not required for display, as they are used for calculating exposure and white balance settings.

In addition it is possible to omit the filename option, in which case the preview is displayed but no file is written, or to redirect all output to stdout. Command line help is available by typing just the application name in on the command line.

Setting up the camera hardware

Please note that camera modules are static-sensitive. Earth yourself prior to handling the PCB: a sink tap/faucet or similar should suffice if you don't have an earthing strap.

The camera board attaches to the Raspberry Pi via a 15-way ribbon cable. There are only two connections to make: the ribbon cable need to be attached to the camera PCB and the Raspberry Pi itself. You need to get it the right way round, or the camera will not work. On the camera PCB, the blue backing on the cable should be facing away from the PCB, and on the Raspberry Pi it should be facing towards the Ethernet connection (or where the Ethernet connector would be if you are using a model A).

Although the connectors on the PCB and the Pi are different, they work in a similar way. On the Raspberry Pi, pull up the tabs on each end of the connector. It should slide up easily, and be able to pivot around slightly. Fully insert the ribbon cable into the slot, ensuring it is straight, then gently press down the tabs to clip it into place. The camera PCB itself also requires you to pull the tabs away from the board, gently insert the cable, then push the tabs back. The PCB connector is a little more awkward than the one on the Pi itself. You can watch a video showing you how to attach the connectors at www.raspberrypi.org/archives/3890 (scroll down for the video).

Setting up the Camera software

Execute the following instructions on the command line to download and install the latest kernel, GPU firmware and applications. You will need an internet connection for this to work correctly.

```
sudo apt-get update
```

```
sudo apt-get upgrade
```

Now you need to enable camera support, using the `raspi-config` program you will have used when you first set up your Raspberry Pi.

```
sudo raspi-config
```

Use the cursor keys to move to the camera option and select *enable*. On exiting `raspi-config` it will ask to reboot. The *enable* option will ensure that on reboot the correct GPU firmware will be running (with the camera driver and tuning), and the GPU memory split is sufficient to allow the camera to acquire enough memory to run correctly.

To test that the system is installed and working, try the following command:

```
raspistill -v -o test.jpg
```

The display should show a 5-second preview from the camera and then take a picture, saved to the file test.jpg, while displaying various informational messages.

Troubleshooting

If the camera is not working correctly, there are number of things to try.

- Are the ribbon connectors all firmly seated and the right way round? They must be straight in their sockets.
- Is the camera module connector firmly attached to the camera PCB? This is the connection from the smaller black camera module itself to the camera PCB. Sometimes this connection can come loose. Using a fingernail, flip up the connector on the PCB, then reseat it with gentle pressure, it engages with a very slight click.
- Have `sudo apt-get update` and `sudo apt-get upgrade` been run?
- Has `raspi-config` been run and the camera enabled?

If things are still not working, try the following:

Error : raspistill/raspivid not found. This probably means your update/upgrade failed in some way. Try it again.

Error : ENOMEM displayed. Camera is not starting up. Check all connections again.

Error : ENOSPC displayed. Camera is probably running out of GPU memory. Check `config.txt` in the `/boot/` folder. The `gpu_mem` option should be at least 128.

If, after all the above, the camera is still not working, it may have a defect (most likely because it has suffered static shock). Try posting on the Raspberry Pi forum in the camera board section to see if there is any more help available there.

Common Command line Options

Preview Window

`--preview, -p` Preview window settings <'x,y,w,h'>

Allows the user to define the size and location on the screen that the preview window will be placed. Note this will be superimposed over the top of any other windows/graphics.

`--fullscreen, -f` Fullscreen preview mode

Forces the preview window to use the whole screen. Note that the aspect ratio of the incoming image will be retained, so there may be bars on some edges.

`--nopreview, -n,` Do not display a preview window

Disables the preview window completely. Note that even though the preview is disabled, the camera will still be producing frames, so will be using power.

`--opacity, -op` Set preview window opacity

Sets the opacity of the preview windows. 0 = invisible, 255 = fully opaque.

Camera Control Options

`--sharpness, -sh` Set image sharpness (-100 to 100)

Set the sharpness of the image, 0 is the default.

`--contrast, -co` Set image contrast (-100 to 100)

Set the contrast of the image, 0 is the default

`--brightness, -br` Set image brightness (0 to 100)

Set the brightness of the image, 50 is the default. 0 is black, 100 is white.

`--saturation, -sa` Set image saturation (-100 to 100)

Set the colour saturation of the image. 0 is the default.

`--ISO, -ISO` Set capture ISO

Sets the ISO to be used for captures. Range is 100 to 800.

`--vstab, -vs` Turn on video stabilization

In video mode only, turn on video stabilization.

`--ev, -ev` Set EV compensation

Set the EV compensation of the image. Range is -10 to +10, default is 0.

`--exposure, -ex` Set exposure mode

Possible options are:

off	
auto	Use automatic exposure mode
night	Select setting for night shooting
nightpreview	
backlight	Select setting for back-lit subject
spotlight	
sports	Select setting for sports (fast shutter etc.)
snow	Select setting optimized for snowy scenery
beach	Select setting optimized for beach
verylong	Select setting for long exposures
fixedfps	Constrain fps to a fixed value
antishake	Antishake mode
fireworks	Select setting optimized for fireworks

Note that not all of these settings may be implemented, depending on camera tuning.

`--awb, -awb` Set automatic white balance (AWB)

off	Turn off white balance calculation
auto	Automatic mode (default)
sun	Sunny mode
cloudshade	Cloudy mode
tungsten	Tungsten lighting mode
fluorescent	Fluorescent lighting mode
incandescent	Incandescent lighting mode
flash	Flash mode
horizon	Horizon mode

`--imxfx, -ifx` Set image effect

none	No effect
negative	Produces a negative image
solarise	Solarise the image
whiteboard	Whiteboard effect
blackboard	Blackboard effect
sketch	Sketch-style effect
denoise	Denoise the image
emboss	Embossed effect
oilpaint	Oil paint-style effect
hatch	Cross-hatch sketch style
gpen	Graphite sketch style
pastel	Pastel effect
watercolour	Watercolour effect
film	Grainy film effect
blur	Blur the image
saturation	Colour-saturate the image
colourswap	Not fully implemented
washedout	Not fully implemented
posterise	Not fully implemented
colourpoint	Not fully implemented
colourbalance	Not fully implemented
cartoon	Not fully implemented

`--colfx, -cfx` Set colour effect <U:V>

The supplied U and V parameters (range 0 to 255) are applied to the U and Y channels of the image. For example, --colfx 128:128 should result in a monochrome image.

`--metering, -mm` Set metering mode

Specify the metering mode used for the preview and capture.

average	Average the whole frame for metering
spot	Spot metering
backlit	Assume a backlit image
matrix	Matrix metering

`--rotation, -rot` Set image rotation (0-359)

Sets the rotation of the image in viewfinder and resulting image. This can take any value from 0 upwards, but due to hardware constraints only 0, 90, 180 and 270-degree rotations are supported.

`--hflip, -hf` Set horizontal flip

Flips the preview and saved image horizontally.

`--vflip, -vf` Set vertical flip

Flips the preview and saved image vertically.

`--roi, -roi` Set sensor region of interest

Allows the specification of the area of the sensor to be used as the source for the preview and capture. This is defined as x,y for the top left corner, and a width and height, all values in normalised coordinates (0.0-1.0). So to set a ROI at half way across and down the sensor, and an width and height of a quarter of the sensor use :

```
-roi 0.5,0.5,0.25,0.25
```

Application-specific settings

raspistill

`--width, -w`	Set image width <size>
`--height, -h`	Set image height <size>
`--quality, -q`	Set jpeg quality <0 to 100>

Quality 100 is almost completely uncompressed. 75 is a good all-round value.

`--raw, -r`	Add raw Bayer data to jpeg metadata

This option inserts the raw Bayer data from the camera in to the JPEG metadata.

`--output -o`	Output filename <filename>

Specify the output filename. If not specified, no file is saved. If the filename is '-', then all output is sent to stdout.

`--verbose, -v`	Output verbose information during run

Outputs debugging/information messages during the program run.

`--timeout, -t`	Time before capture and shut down

The program will run for this length of time, then take the capture (if output is specified). If not specified, this is set to 5 seconds.

`--timelapse, -tl` Timelapse mode.

The specific value is the time between shots in milliseconds. Note you should specify %04d at the point in the filename where you want a frame count number to appear. For example:

```
-t 30000 -tl 2000 -o image%04d.jpg
```

will produce a capture every 2 seconds over a total period of 30s, named image1.jpg, image0002.jpg...image0015.jpg. Note that the %04d indicates a four-digit number with leading zeros added to pad to the required number of digits. So, for example, %08d would result in an eight-digit number.

`--thumb, -th` Set thumbnail parameters (x:y:quality)

Allows specification of the thumbnail image inserted in to the JPEG file. If not specified, defaults are a size of 64x48 at quality 35.

`--demo, -d` Run a demo mode <milliseconds>

This options cycles through range of camera options, and no capture is done. The demo will end at the end of the timeout period, irrespective of whether all the options have been cycled. The time between cycles should be specified as a millisecond value.

`--encoding, -e` Encoding to use as output file

Valid options are jpg, bmp, gif and png. Note that unaccelerated image types (gif, png, bmp) will take much longer to save than jpg, which is hardware accelerated. Also note that the filename suffix is completely ignored when encoding a file.

`--exif, -x` EXIF tag to apply to captures (format as 'key=value')

Allows the insertion of specific EXIF tags into the JPEG image. You can have up to 32 EXIF tge entries. This is useful for things like adding GPS metadata. For example, to set the longitude:

```
--exif GPS.GPSLongitude=5/1,10/1,15/100
```

would set the longitude to 5degs, 10 minutes, 15 seconds. See EXIF documentation for more details on the range of tags available; the supported tags are as follows:

IFD0.< or
IFD1.<
ImageWidth, ImageLength, BitsPerSample, Compression, PhotometricInterpretation, ImageDescription, Make, Model, StripOffsets, Orientation, SamplesPerPixel, RowsPerString, StripByteCounts, Xresolution, Yresolution, PlanarConfiguration, ResolutionUnit, TransferFunction, Software, DateTime, Artist, WhitePoint, PrimaryChromaticities, JPEGInterchangeFormat, JPEGInterchangeFormatLength, YcbCrCoefficients, YcbCrSubSampling, YcbCrPositioning, ReferenceBlackWhite, Copyright>

EXIF.<
ExposureTime, FNumber, ExposureProgram, SpectralSensitivity, aISOSpeedRatings, OECF, ExifVersion, DateTimeOriginal, DateTimeDigitized, ComponentsConfiguration, CompressedBitsPerPixel, ShutterSpeedValue, ApertureValue, BrightnessValue, ExposureBiasValue, MaxApertureValue, SubjectDistance, MeteringMode, LightSource, Flash, FocalLength, SubjectArea, MakerNote, UserComment, SubSecTime, SubSecTimeOriginal, SubSecTimeDigitized, FlashpixVersion, ColorSpace, PixelXDimension, PixelYDimension, RelatedSoundFile, FlashEnergy, SpacialFrequencyResponse, FocalPlaneXResolution,

FocalPlaneYResolution, FocalPlaneResolutionUnit,
SubjectLocation, ExposureIndex, SensingMethod, FileSource,
SceneType, CFAPattern, CustomRendered,
ExposureMode,WhiteBalance, DigitalZoomRatio,
FocalLengthIn35mmFilm, SceneCaptureType, GainControl,
Contrast, Saturation, Sharpness, DeviceSettingDescription,
SubjectDistanceRange, ImageUniqueID>

GPS.<
GPSVersionID, GPSLatitudeRef, GPSLatitude,
GPSLongitudeRef, GPSLongitude, GPSAltitudeRef, GPSAltitude,
GPSTimeStamp, GPSSatellites, GPSStatus, GPSMeasureMode,
GPSDOP, GPSSpeedRef, GPSSpeed, GPSTrackRef,
GPSTrack, GPSImgDirectionRef, GPSImgDirection,
GPSMapDatum, GPSDestLatitudeRef, GPSDestLatitude,
GPSDestLongitudeRef, GPSDestLongitude,
GPSDestBearingRef, GPSDestBearing, GPSDestDistanceRef,
GPSDestDistance, GPSProcessingMethod,
GPSAreaInformation, GPSDateStamp, GPSDifferential>

EINT.<
InteroperabilityIndex, InteroperabilityVersion,
RelatedImageFileFormat, RelatedImageWidth,
RelatedImageLength>

Note that a small subset of these tags will be set automatically by the camera system, but will be overridden by any exif options on the command line.

`--fullpreview, -fp` Full Preview mode

This runs the preview windows using the full resolution capture mode. Maximum frames per second in this mode is 15fps and the preview will have the same field of view as the capture. Captures should happen more quickly as no mode change should be required. This feature is currently under development.

raspistillyuv

Many of the options for `raspistillyuv` are the same as those for `raspistill`. This section shows the differences.

Unsupported Options:

`--exif, --encoding, --thumb, --raw, --quality`

Extra Options:

`--rgb, -rgb`	Save uncompressed data as RGB888

This option forces the image to be saved as RGB data with 8 bits per channel, rather than YUV420.

Note that the image buffers saved in `raspistillyuv` are padded to a horizontal size divisible by 16 (so there may be unused bytes at the end of each line to made the width divisible by 16). Buffers are also padded vertically to be divisible by 16, and in the YUV mode, each plane of Y,U,V is padded in this way.

raspivid

`--width, -w`	Set image width <size>

Width of resulting video. This should be between 64 and 1920.

`--height, -h`	Set image height <size>

Height of resulting video. This should be between 64 and 1080.

`--bitrate, -b`	Set bitrate

Use bits per second, so 10MBits/s would be -b 10000000. For H264, 1080p a high quality bitrate would be 15Mbits/s or more.

`--output, -o` Output filename <filename>.

Specify the output filename. If not specified, no file is saved. If the filename is '-', then all output is sent to stdout.

`--verbose, -v` Output verbose information during run

Outputs debugging/information messages during the program run.

`--timeout, -t` Time before capture and shut down

The program will run for this length of time, then take the capture (if output is specified). If not specified, this is set to five seconds. Setting 0 will mean the application will run continuously until stopped with Ctrl-C.

`--demo, -d` Run a demo mode <milliseconds>

This option cycles through range of camera options, no capture is done, the demo will end at the end of the timeout period, irrespective of whether all the options have been cycled. The time between cycles should be specified as a millisecond value.

`--framerate, -fps` Specify the frames per second to record

At present, the minimum frame rate allowed is 2fps, the maximum is 30fps. This is likely to change in the future.

`--penc, -e` Display preview image *after* encoding

Switch on an option to display the preview after compression. This will show any compression artefacts in the preview window. In normal operation, the preview will show the camera output prior to being compressed. This option is not guaranteed to work in future releases.

`--intra, -g`	Specify the intra refresh period (key frame rate/GoP)

Sets the intra refresh period (GoP) rate for the recorded video. H.264 video uses a complete frame (I-frame) every intra refresh period from which subsequent frames are based. This options specifies the numbers of frames between each I-frame. Larger numbers here will reduce the size of the resulting video, smaller numbers make the stream more robust to error.

Examples

Still captures

By default, captures are done at the highest resolution supported by the sensor. This can be changed using the -w and -h command line options.

Taking a default capture after two seconds (note times are specified in milliseconds) on viewfinder, saving in image.jpg

```
raspistill -t 2000 -o image.jpg
```

Take a capture at a different resolution

```
raspistill -t 2000 -o image.jpg -w 640
-h 480
```

Now reduce the quality considerably to reduce file size

```
raspistill -t 2000 -o image.jpg -q 5
```

Force the preview to appear at coordinate 100,100, with width 300 and height 200 pixels.

```
raspistill -t 2000 -o image.jpg -p
100,100,300,200
```

Disable preview entirely.

```
raspistill -t 2000 -o image.jpg -n
```

Save the image as a png file (lossless compression, but slower than JPEG). Note that the filename suffix is ignored when choosing the image encoding.

```
raspistill -t 2000 -o image.png –e png
```

Add some EXIF information to the JPEG. This sets the Artist tag name to Mooncake, and the GPS altitude to 123.5m. Note that if setting GPS tags you should set as a minimum GPSLatitude, GPSLatitudeRef, GPSLongitude, GPSLongitudeRef, GPSAltitude and GPSAltitudeRef.

```
raspistill -t 2000 -o image.jpg -x
IFD0.Artist=Mooncake -x
GPS.GPSAltitude=1235/10
```

Set an emboss style image effect.

```
raspistill -t 2000 -o image.jpg -ifx emboss
```

Set the U and V channels of the YUV image to specific values (128:128 produces a greyscale image)

```
raspistill -t 2000 -o image.jpg -cfx
128:128
```

Run preview ONLY for two seconds, no saved image.

```
raspistill -t 2000
```

Take timelapse picture, one every 10 seconds for 10 minutes (10 minutes = 600000ms), named image_number_1_today.jpg, image_number_2_today.jpg onwards.

```
raspistill -t 600000 -tl 10000 -o
image_num_%d_today.jpg
```

Take a picture and send image data to stdout

```
raspistill -t 2000 -o -
```

Take a picture and send image data to file

```
raspistill -t 2000 -o - > my_file.jpg
```

Video Captures

Image size and preview settings are the same as for stills capture. Default size for video recording is 1080p (1920x1080)

Record a 5s clip with default settings (1080p30)

```
raspivid -t 5000 -o video.h264
```

Record a 5s clip at a specified bitrate (3.5MBits/s)

```
raspivid -t 5000 -o video.h264 -b 3500000
```

Record a 5s clip at a specified framerate (5fps)

```
raspivid -t 5000 -o video.h264 -f 5
```

Encode a 5s camera stream and send image data to stdout

```
raspivid -t 5000 -o -
```

Encode a 5s camera stream and send image data to file

```
raspivid -t 5000 -o - > my_file.h264
```

FAKULTAS TEKNOLOGI INDUSTRI
JURUSAN TEKNIK ELEKTRO S-1
Jl. Raya Karanglo, Km. 2 MALANG

PERMOHONAN PERSETUJUAN SKRIPSI

Yang Bertanda Tangan Dibawah Ini:

Nama : BINASHIR ROFIYAH
N I M : 112530
Semester : VII
Fakultas :Teknologi Industri
Jurusan :Teknik Elektro S-I
Konsentrasi :~~TEKNIK ENERGI LISTRIK~~
~~TEKNIK ELEKTRONIKA~~
~~TEKNIK KOMPUTER DAN INFORMATIKA~~
TEKNIK KOMPUTER
~~TEKNIK TELEKOMUNIKASI~~
Alamat : JL WIJAYA NO 47 SINGOSARI - MALANG

Dengan ini kami mengajukan permohonan untuk mendapatkan persetujuan untuk membuat SKRIPSI Tingkat Sarjana. Untuk melengkapi permohonan tersenut, bersama ini kami lampirkan persyaratan-persyaratan yang harus dipenuhi.
Adapun persyaratan- persyaratan pengambilan SKRIPSI adalah sebagai berikut:

1. Telah melaksanakan semua praktikum sesui dengan konsentrasinya (.....)
2. Telah lulus dan menyerahkan laporan Praktek Kerja (.....)
3. Telah lulus seluruh mata kuliah keahlian (MKB)sesuai konsentrasinya (.....)
4. Telah menempuh matakuliah> 134 sks dengan IPK > 2 dan tidak ada nilai E (.....)
5. Telah mengikuti secara aktif kegiatan seminar Skripsi yang diadakan Jurusan (.....)
6. Memenuhi persyaratan administrasi (.....)

Demikian permohonan ini untuk mendapatkan penyelesaian lebih lanjut dan atas perhatiannya kami ucapkan terima kasih.

Telah diteliti kebenarannya data tersebut diatas
Recording Teknik Elektro S-I

(.....)

Malang, 08 September **2014**
Pemohon

(BINASHIR ROFIYAH)

Disetujui
Ketua Prodi Teknik Elektro S-I

M. Ibrahim Ashari, ST, MT
NIP. P. 1030100358

Mengetahui
Dosen Wali

(.....)
SOTYOHADI, ST.

Catatan:
Bagi mahasiswa yang telah memenuhi persyaratan mengambil SKRIPSI agar membuat proposal dan mendapat persetujuan dari Ketua Prodi T. elektro S-I

PERKUMPULAN PENGELOLA PENDIDIKAN UMUM DAN TEKNOLOGI NASIONAL MALANG

INSTITUT TEKNOLOGI NASIONAL MALANG

FAKULTAS TEKNOLOGI INDUSTRI
FAKULTAS TEKNIK SIPIL DAN PERENCANAAN
PROGRAM PASCASARJANA MAGISTER TEKNIK

PT. BNI (PERSERO) MALANG
BANK NIAGA MALANG

Kampus I : Jl. Bendungan Sigura-gura No. 2 Telp. (0341) 551431 (Hunting), Fax. (0341) 553015 Malang 65145
Kampus II : Jl. Raya Karanglo, Km 2 Telp. (0341) 417636 Fax. (0341) 417634 Malang

omor Surat : ITN-243/EL-FTI/2014
ampiran : -
erihal : BIMBINGAN SKRIPSI

epada : Yth. Bapak/Ibu **Dr. Eng. Aryuanto Soetedjo, ST, MT**
Dosen Teknik Elektro S-1
ITN MALANG

Dengan Hormat

Sesuai dengan permohonan dan persetujuan dalam Proposal Skripsi untuk mahasiswa :

Nama	: **BINASHIR ROFI'AH**
Nim	: **1112530**
Fakultas	: **Teknologi Industri**
Program Studi	: **Teknik Elektro S-1**
Konsentrasi	: **Teknik Komputer**

Maka dengan ini pembimbingan tersebut kami serahkan sepenuhnya kepada Saudara/i selama masa waktu :

" Semester Ganjil Tahun Akademik 2014-2015 "

Demikian agar maklum dan atas perhatian serta bantuannya kami sampaikan terima kasih.

Mengetahui
Ketua Program Studi Teknik Elektro S-1

M. Ibrahim Ashari, ST, MT
NIP.P. 1030100358

PERKUMPULAN PENGELOLA PENDIDIKAN UMUM DAN TEKNOLOGI NASIONAL MALANG

INSTITUT TEKNOLOGI NASIONAL MALANG

FAKULTAS TEKNOLOGI INDUSTRI
FAKULTAS TEKNIK SIPIL DAN PERENCANAAN
PROGRAM PASCASARJANA MAGISTER TEKNIK

PT. BNI (PERSERO) MALANG
BANK NIAGA MALANG

Kampus I : Jl. Bendungan Sigura-gura No. 2 Telp. (0341) 551431 (Hunting), Fax. (0341) 553015 Malang 65145
Kampus II : Jl. Raya Karanglo, Km 2 Telp. (0341) 417636 Fax. (0341) 417634 Malang

omor Surat : ITN-243/EL-FTI/2014
ampiran : -
erihal : BIMBINGAN SKRIPSI

epada : Yth. Bapak/Ibu **Bima Aulia Firmandani, ST**
Dosen Teknik Elektro S-1
ITN MALANG

Dengan Hormat

Sesuai dengan permohonan dan persetujuan dalam Proposal Skripsi untuk mahasiswa :

Nama	: **BINASHIR ROFI'AH**
Nim	: **1112530**
Fakultas	: **Teknologi Industri**
Program Studi	: **Teknik Elektro S-1**
Konsentrasi	: **Teknik Komputer**

Maka dengan ini pembimbingan tersebut kami serahkan sepenuhnya kepada Saudara/i selama masa waktu :

" Semester Ganjil Tahun Akademik 2014-2015 "

Demikian agar maklum dan atas perhatian serta bantuannya kami sampaikan terima kasih.

Mengetahui
Ketua Program Studi Teknik Elektro S-1

M. Ibrahim Ashari, ST, MT
NIP.P. 1030100358

PROGRAM STUDI TEKNIK ELEKTRO S-1
FAKULTAS TEKNOLOGI INDUSTRI
INSTITUT TEKNOLOGI NASIONAL MALANG
Kampus II : Jl. Raya Karanglo Km. 2 Telp. (0341) 417636 Malang

BERITA ACARA SEMINAR PROPOSAL SKRIPSI
PROGRAM STUDI TEKNIK ELEKTRO S-1
Konsentrasi : Teknik Komputer

1.	Nim	: **1112530**	
2.	Nama	: **BINASHIR ROFI'AH**	
3.	Konsentrasi Jurusan	: **Teknik Komputer**	
4.	Jadwal Pelaksanaan:	Waktu	Tempat
	2 Oktober 2014	09:00	III.1.4
5.	Judul proposal yang diseminarkan Mahasiswa	IMPLEMENTASI VISUAL SIMULTANEOUS LOCALISATION AND MAPPING (VSLAM) MENGGUNAKAN SINGLE CAMERA	
6.	Perubahan judul yang diusulkan oleh Kelompok Dosen Keahlian		
7.	Catatan : - Tujuan lebih di detailkan ke VSLAM pada Raspberry Pi - Pengujian		
8.	Catatan :		

Persetujuan judul Skripsi

Disetujui, Dosen Keahlian I	Disetujui, Dosen Keahlian II	Disetujui, Dosen Keahlian III
(Ir. F. Yudi C, MT)	(.....)	(.....)

Mengetahui, Ketua Program Studi Teknik Elektro S-1	Disetujui, Calon Dosen Pembimbing ybs	
	Pembimbing I	Pembimbing II
M. Ibrahim Ashari ,ST, MT NIP. P 1030100358	(.....)	(.....)

JURUSAN TEKNIK ELEKTRO S-1
FAKULTAS TEKNOLOGI INDUSTRI
INSTITUT TEKNOLOGI NASIONAL MALANG

BERITA ACARA SEMINAR PROGRESS SKRIPSI
PROGRAM STUDI TEKNIK ELEKTRO S1

KONSENTRASI	T. KOMPUTER

1.	Nama Mahasiswa	BINASHIR ROFI`AH	NIM	1112530
2.	Keterangan	Tanggal	Waktu	Tempat / Ruang
	Pelaksanaan	**12 Desember 2014**		
3.	Judul Skripsi	IMPLEMENTASI VISUAL SIMULTANEOUS LOCALISATION AND MAPPING (VSLAM) MENGGUNAKAN SINGLE CAMERA		
4.	Perubahan Judul	.....		
5.	Catatan :	- lanjutkan. - tambahkan hasil & pembahasan/perhitungan.		
6.	Mengetahui, Ketua Jurusan M. Ibrahim Ashari, ST, MT	Disetujui, Dosen Pembimbing Pembimbing I Dr. Eng. Aryuanto Soetedjo, ST, MT		Pembimbing II Bima Aulia Firmandani, ST

PROGRAM STUDI TEKNIK ELEKTRO S-1
FAKULTAS TEKNOLOGI INDUSTRI
INSTITUT TEKNOLOGI NASIONAL MALANG
Kampus II : Jl. Raya Karanglo Km. 2 Malang

MONITORING BIMBINGAN SKRIPSI

SEMESTER GANJIL TAHUN AKADEMIK 2014-2015

Nama Mahasiswa : BINASHIR ROFI'AH
NIM : 1112530
Nama Pembimbing : Dr. Eng. Aryuanto Soetedjo, ST, MT
Judul Skripsi : IMPLEMENTASI *VISUAL SIMULTANEOUS LOCALISATION AND MAPPING* (VSLAM) MENGGUNAKAN *SINGLE CAMERA*

Minggu Ke-	Hari, Tanggal	Waktu Bimbingan	Materi Bimbingan	Paraf
1	Rabu 15/10/14	12:00	- Monoslam pd Linux? → Exp di Linux + Resp'	
2	Senin 3/11/14	14:30	- Monoslam ada di Linux → Expl sesuai bingkai Kalil	2
3	Senin 17/11/14	14:30	- Exp. VSLAM (Linux, jpg.	
4	Sabtu 22/11/14	12:30	- fps Rasp ican? → Convert JPG → PGM	
5	Sabtu 29/11/14	12:30	- File jpg ✓ - File transfer ✓ FPS ?	
6	Senin 22/12/14	10:00	- Exp - fitur tracking	
7	29/12/14	10:00	- Exp Rasp 1	

Malang, ...
Pembimbing,

Dr. Eng. Aryuanto Soetedjo, ST, MT
NIP Y. 1030800417

PROGRAM STUDI TEKNIK ELEKTRO S-1
FAKULTAS TEKNOLOGI INDUSTRI
INSTITUT TEKNOLOGI NASIONAL MALANG
Kampus II : Jl. Raya Karanglo Km. 2 Malang

MONITORING BIMBINGAN SKRIPSI

SEMESTER GANJIL TAHUN AKADEMIK 2014-2015

Nama Mahasiswa : BINASHIR ROFI'AH
NIM : 1112530
Nama Pembimbing : Dr. Eng. Aryuanto Soetedjo, ST, MT
Judul Skripsi : IMPLEMENTASI *VISUAL SIMULTANEOUS LOCALISATION AND MAPPING* (VSLAM) MENGGUNAKAN *SINGLE CAMERA*

Minggu Ke-	Hari, Tanggal	Waktu Bimbingan	Materi Bimbingan	Paraf
8	3 9/12/14	10:00	Eks. Rampican	
9	6/1/14	12:00	Camera + Rabat	
10	Senin 26/1/15	12:10	Bab 1 - 4	
11	Kamis 5/2/15	13:00	Makalah Seminar	
12				
13				
14				

Malang, ...
Pembimbing,

Dr. Eng. Aryuanto Soetedjo, ST, MT

PROGRAM STUDI TEKNIK ELEKTRO S-1
FAKULTAS TEKNOLOGI INDUSTRI
INSTITUT TEKNOLOGI NASIONAL MALANG
Kampus II : Jl. Raya Karanglo Km. 2 Malang

MONITORING BIMBINGAN SKRIPSI
SEMESTER GANJIL TAHUN AKADEMIK 2014-2015

Nama Mahasiswa : BINASHIR ROFI'AH
NIM : 1112530
Nama Pembimbing : Bima Aulia Firmandani, ST
Judul Skripsi : IMPLEMENTASI *VISUAL SIMULTANEOUS LOCALISATION AND MAPPING* (VSLAM) MENGGUNAKAN *SINGLE CAMERA*

Minggu Ke-	Hari, Tanggal	Waktu Bimbingan	Materi Bimbingan	Paraf
1	Rabu 15 Okt 2014	13.00	Konsep VSLAM pada Linux	
2	Senin, 03 Nov 2014	14.00	Perancangan Timelapse	
3	Sabtu, 22 Nov 2014	12.30	Gambar Timelapse ada yang Hilang diurutan	
4	Sabtu, 20 Des 2014	11.00	Eksplolari Fitur	
5	Selasa, 23 Des 2014	12.00	Eksplorasi Fitur lanjutan	
6	Selasa 07 Des 2014	11.00	Protokol Sinkronisasi	
7	Senin 05 Jan 2014	12.00	Pengiriman Gambar	

Malang, ...

Pembimbing

Bima Aulia Firmandani, ST

PROGRAM STUDI TEKNIK ELEKTRO S-1
FAKULTAS TEKNOLOGI INDUSTRI
INSTITUT TEKNOLOGI NASIONAL MALANG
Kampus II : Jl. Raya Karanglo Km. 2 Malang

MONITORING BIMBINGAN SKRIPSI
SEMESTER GANJIL TAHUN AKADEMIK 2014-2015

Nama Mahasiswa : BINASHIR ROFI'AH
NIM : 1112530
Nama Pembimbing : Bima Aulia Firmandani, ST
Judul Skripsi : IMPLEMENTASI *VISUAL SIMULTANEOUS LOCALISATION AND MAPPING* (VSLAM) MENGGUNAKAN *SINGLE CAMERA*

Minggu Ke-	Hari, Tanggal	Waktu Bimbingan	Materi Bimbingan	Paraf
8	Senin, 12 Jan 2015	10.00	Kalibrasi Kamera	
9	Kamis 15 Jan 2015	09.00	Pengaksesan Gambar	
10	Senin 19 Jan 2015	09.00	Pengujian Program	
11	Rabu 28 Jan 2015	10.00	Laporan Bab I - V	
12	Jumat 6 Feb 2015	12.00	Makalah Seminar Hasil	
13				
14				

Malang, ...
Pembimbing

Bima Aulia Firmandani, ST

PROGRAM STUDI TEKNIK ELEKTRO S-1
FAKULTAS TEKNOLOGI INDUSTRI
INSTITUT TEKNOLOGI NASIONAL MALANG
Kampus II : Jl. Raya Karanglo Km. 2 Malang

MONITORING KEHADIRAN SKRIPSI

SEMESTER GANJIL TAHUN AKADEMIK 2014-2015

Nama Mahasiswa : BINASHIR ROFI'AH
NIM : 1112530
Nama Pembimbing : 1. Dr. Eng. Aryuanto Soetedjo, ST, MT
2. Bima Aulia Firmandani, ST
Tempat Skripsi : Laboratorium Pemrograman Komputer dan Multimedia
Judul Skripsi : IMPLEMENTASI *VISUAL SIMULTANEOUS LOCALISATION AND MAPPING* (VSLAM) MENGGUNAKAN *SINGLE CAMERA*

Minggu Ke-	Hari, Tanggal	Waktu Kehadiran	Kegiatan/Aktivitas	Paraf Ka. Lab.
1	Kamis, 9/10/2014	09.00	Mempersiapkan kebutuhan Skripsi	
2	Sabtu, 11/10/2014	08.00	Instalasi OS	
3	Rabu, 22/10/2014	09.00	Mempersiapkan kebutuhan Software	
4	Sabtu, 08/11/2014	12.00	Instalasi CMake & Git	
5	Senin, 10/11/2014	08.00	Instalasi Eclipse & FileZilla	
6	Kamis, 20/11/2014	10.00	Membuat Program Capture Gambar dan RaspiCam	
7	Sabtu, 29/11/2014	08.00	Trial & Error Program Capture	

Malang, 08 November 2014
Kepala Laboratorium
Pemrograman Komputer dan Multimedia

Solvohadi, ST

PROGRAM STUDI TEKNIK ELEKTRO S-1
FAKULTAS TEKNOLOGI INDUSTRI
INSTITUT TEKNOLOGI NASIONAL MALANG
Kampus II : Jl. Raya Karanglo Km. 2 Malang

MONITORING KEHADIRAN SKRIPSI

SEMESTER GANJIL TAHUN AKADEMIK 2014-2015

Nama Mahasiswa : BINASHIR ROFI'AH
NIM : 1112530
Nama Pembimbing : 1. Dr. Eng. Aryuanto Soetedjo, ST, MT
2. Bima Aulia Firmandani, ST
Tempat Skripsi : Laboratorium Pemrograman Komputer dan Multimedia
Judul Skripsi : IMPLEMENTASI *VISUAL SIMULTANEOUS LOCALISATION AND MAPPING* (VSLAM) MENGGUNAKAN *SINGLE CAMERA*

Minggu Ke-	Hari, Tanggal	Waktu Kehadiran	Kegiatan/Aktivitas	Paraf Ka. Lab.
8	Kamis, 4/12/2014	09.00	Membuat Program Mengurutkan Gambar Yang Disimpan	
9	Sabtu, 13/12/2014	09.00	Trial Error Program Mengurutkan Gambar	
10	Jumat, 19/12/2014	08.00	Membuat Program VSLAM + Rpi	
11	Senin, 05/01/2015	11.00	Trial Error Program VSLAM : Inisialisasi Fitur Target	
12	Kamis, 08/01/2015	08.00	Pengujian Hardware	
13	Rabu, 14/01/2015	08.00	Pengujian Software	
14	Sabtu, 17/01/2015	08.00	Pengujian Keseluruhan	

Malang, 17 Januari 2015
Kepala Laboratorium
Pemrograman Komputer dan Multimedia

ITUT TEKNOLOGI NASIONAL
Bendungan Sigura-gura No. 2
MALANG

No. 015703

UNTUK MAHASISWA

SURAT PUAS TUGAS / PRAKTIKUM

Nomor : ITN _____ / SPT / 2015

erangkan bahwa mahasiswa :

Nama : BINASHIR ROFI'AH NIM : 1112530

Jurusan : Teknik ELEKTRO S-1 / TEKNIK KOMPUTER

Angkatan : 2011 (DUA RIBU SEBELAS)

Telah menyelesaikan tugas : SKRIPSI : IMPLEMENTASI VSLAM MENGGUNAKAN SINGLE CAMERA

Semester : VII - GANJIL 2015 (TUJUH)

Dengan Hasil : 93

Malang, 17 Pebruari 2015

Dosen / Asisten ybs.

.....

INSTITUT TEKNOLOGI NASIONAL MALANG
FAKULTAS TEKNOLOGI INDUSTRI
JURUSAN TEKNIK ELEKTRO

Formulir Perbaikan Ujian Skripsi

Dalam pelaksanaan Ujian Skripsi Janjang Strata 1 Jurusan Teknik Elektro Konsentrasi T. Energi Listrik / T. Elektronika / T.Infokom, maka perlu adanya perbaikan skripsi untuk mahasiswa :

NAMA : Binathir Rafi'i
N I M : 11.23.30
Perbaikan meliputi :

Revisi - abstrak
- kesimpulan

Malang,

(Ir. Nanik)

INSTITUT TEKNOLOGI NASIONAL MALANG
FAKULTAS TEKNOLOGI INDUSTRI
JURUSAN TEKNIK ELEKTRO

Formulir Perbaikan Ujian Skripsi

Dalam pelaksanaan Ujian Skripsi Janjang Strata 1 Jurusan Teknik Elektro Konsentrasi T. Energi Listrik / T. Elektronika / T.Infokom, maka perlu adanya perbaikan skripsi untuk mahasiswa :

NAMA : Binashir Rofi'ah
NIM : 1112530
Perbaikan meliputi :

Malang, 21-2-2015

Yusuf Ismail Nakhoda

PERKUMPULAN PENGELOLA PENDIDIKAN UMUM DAN TEKNOLOGI NASIONAL MALANG
INSTITUT TEKNOLOGI NASIONAL MALANG
FAKULTAS TEKNOLOGI INDUSTRI
FAKULTAS TEKNIK SIPIL DAN PERENCANAAN
PROGRAM PASCASARJANA MAGISTER TEKNIK
T. BNI (PERSERO) MALANG
BANK NIAGA MALANG
Kampus I : Jl. Bendungan Sigura-gura No. 2 Telp. (0341) 551431 (Hunting), Fax. (0341) 553015 Malang 65145
Kampus II : Jl. Raya Karanglo, Km 2 Telp. (0341) 417636 Fax. (0341) 417634 Malang

PERSETUJUAN PERBAIKAN SKRIPSI

Dari hasil ujian skripsi Program Studi Teknik Elektro jenjang Strata Satu (S-1) yang diselenggarakan pada:

Hari : Sabtu
Tanggal : 21 Februari 2015

Telah dilakukan perbaikan skripsi oleh:

Nama : Binashir Rofi'ah
NIM : 1112530
Program Studi : Teknik Elektro S-1
Konsentrasi : Teknik Komputer
Judul Skripsi : **IMPLEMENTASI *VISUAL SIMULTANEOUS LOCALISATION AND MAPPING (VSLAM)* MENGGUNAKAN *SINGLE CAMERA***

No	Materi Perbaikan	Keterangan
1.	Abstrak	
2.	Bab III: Perancangan Perhitungan Eksekusi Transfer Gambar	
3.	Kesimpulan	

Dosen Penguji I,

Irmalia Suryani Faradisa, ST, MT
NIP P. 1030100365

Dosen Penguji II,

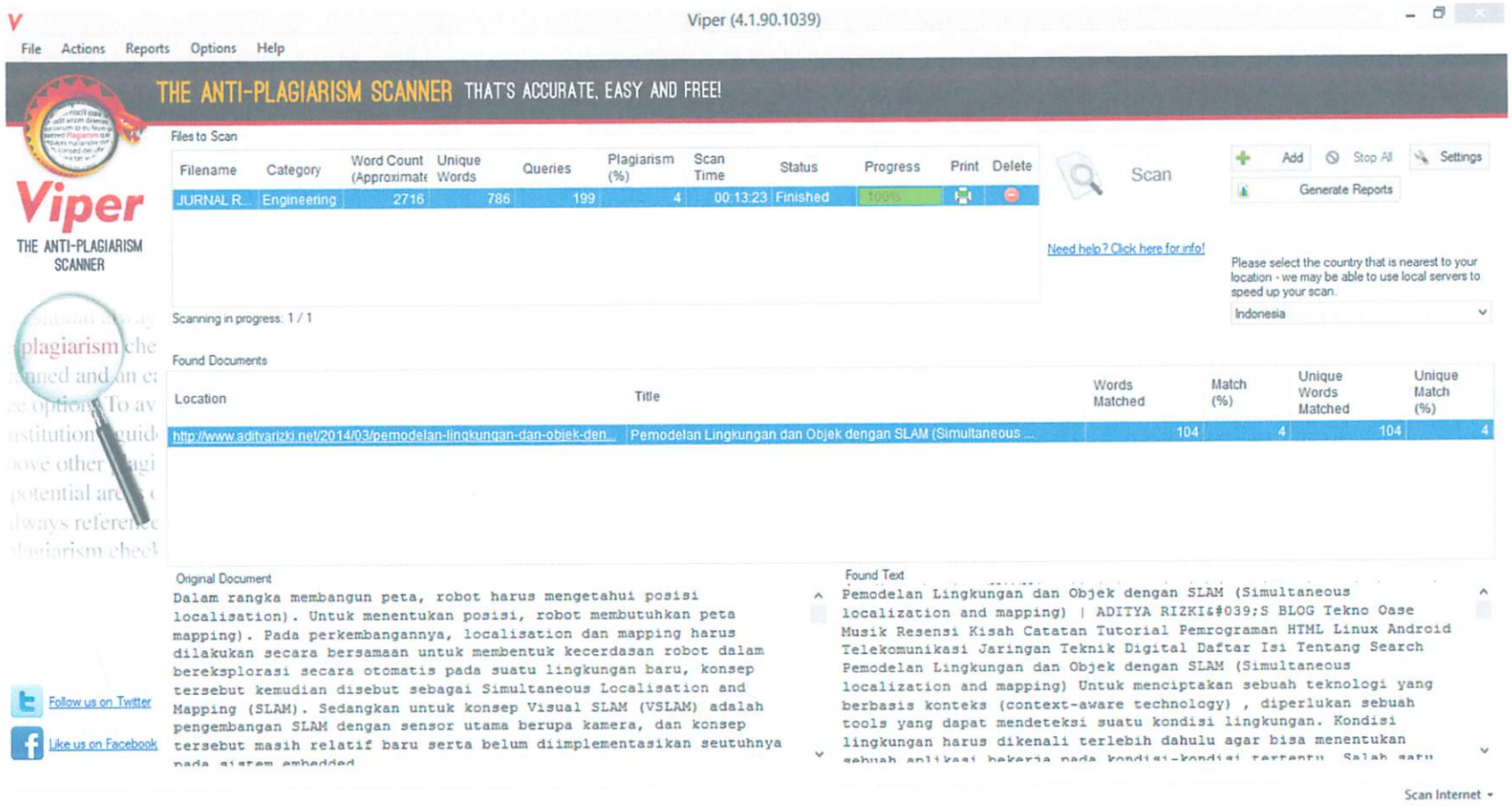
Ir. Yusuf Ismail Nakhoda, MT
NIP Y. 1018800189

Dosen Pembimbing I,

Dr. Eng. Aryuanto Soetedjo, ST, MT
NIP Y. 1030800417

Dosen Pembimbing II,

Bima Aulia Firmandani, ST


Viper (4.1.90.1039)
File Actions Reports Options Help
THE ANTI-PLAGIARISM SCANNER THAT'S ACCURATE, EASY AND FREE!
Viper
THE ANTI-PLAGIARISM SCANNER
Files to Scan
Filename Category Word Count (Approximate Unique Words Queries Plagiarism (%) Scan Time Status Progress Print Delete
JURNAL R... Engineering 2716 786 199 4 00:13:23 Finished 100%
Scan
Add Stop All Settings
Generate Reports
Need help? Click here for info!
Please select the country that is nearest to your location - we may be able to use local servers to speed up your scan.
Indonesia
Scanning in progress: 1 / 1
Found Documents
Location Title Words Matched Match (%) Unique Words Matched Unique Match (%)
http://www.adityarizki.net/2014/03/pemodelan-lingkungan-dan-objek-den... Pemodelan Lingkungan dan Objek dengan SLAM (Simultaneous ... 104 4 104 4
Original Document
Dalam rangka membangun peta, robot harus mengetahui posisi localisation). Untuk menentukan posisi, robot membutuhkan peta mapping). Pada perkembangannya, localisation dan mapping harus dilakukan secara bersamaan untuk membentuk kecerdasan robot dalam bereksplorasi secara otomatis pada suatu lingkungan baru, konsep tersebut kemudian disebut sebagai Simultaneous Localisation and Mapping (SLAM). Sedangkan untuk konsep Visual SLAM (VSLAM) adalah pengembangan SLAM dengan sensor utama berupa kamera, dan konsep tersebut masih relatif baru serta belum diimplementasikan seutuhnya
Found Text
Pemodelan Lingkungan dan Objek dengan SLAM (Simultaneous localization and mapping) | ADITYA RIZKI'S BLOG Tekno Oase Musik Resensi Kisah Catatan Tutorial Pemrograman HTML Linux Android Telekomunikasi Jaringan Teknik Digital Daftar Isi Tentang Search Pemodelan Lingkungan dan Objek dengan SLAM (Simultaneous localization and mapping) Untuk menciptakan sebuah teknologi yang berbasis konteks (context-aware technology) , diperlukan sebuah tools yang dapat mendeteksi suatu kondisi lingkungan. Kondisi lingkungan harus dikenali terlebih dahulu agar bisa menentukan
Follow us on Twitter
Like us on Facebook
Scan Internet

AUTOBIOGRAFI PENULIS

Penulis skripsi ini bernama Binashir Rofi'ah yang akrab dipanggil Rofi, merupakan anak pertama dari 4 bersaudara, lahir di kota Malang pada tanggal 19 Maret 1993. Penulis adalah alumni SMPN 1 Singosari yang lulus pada tahun 2008 dan SMKN 2 Singosari tamatan 2011. Penulis melanjutkan pendidikannya di ITN Malang setelah lolos seleksi dan mendapat beasiswa penuh dari pemerintah Indonesia melalui ITN Malang pada Progam Hibah Kompetisi berbasis Institusi (PKHI). Hingga saat ini, penulis mempunyai hobi membaca, baik karya tulis yang bernilai ilmiah, seni, fiksi, maupun non fiksi. Kesukaan dalam membaca tersebut diawali sejak kecil ketika penulis merasa takjub akan pengetahuan yang belum pernah didapat ternyata diceritakan pada buku-buku yang dibacanya Selain hobi dalam membaca, penulis juga membuka diri pada hal-hal baru yang menyenangkan. Hal ini terbukti pada saat di bangku sekolah penulis pernah mengikuti beberapa ekstrakulikuler, seperti kepramukaan, tata boga, pecinta alam, tim bola voli, keputrian, remaja masjid dan paduan suara. Serta ketika masa di perkuliahan penulis pernah berperan aktif dalam tim robotika ITN Malang dan asisten praktikum di Laboratorium Pemrograman Komputer dan Multimedia. Walaupun masih dalam tahap membangun jati diri, penulis berharap menjadi orang yang bahagia dan berguna bagi diri sendiri dan sesama, serta cita-cita menjadi dosen yang pengetahuannya tumbuh bersama mahasiswanya terkabul, sehingga mampu kembali dengan membawa amal yang mengalir. Penulis dapat dihubungi melalui media sosial di link.of.rofi@gmail.com atau di www.facebook.com/rrooffii.