PENGATURAN GERAK PADA ROBOT BERJALAN BERBASIS CITRA

Oleh :

KARTIKA
NRP. 2202 204 005

PROGRAM STUDI MAGISTER
BIDANG KEAHLIAN ELEKTRONIKA
JURUSAN TEKNIK ELEKTRO
FAKULTAS TEKNOLOGI INDUSTRI
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA
2005

PENGATURAN GERAK PADA ROBOT BERJALAN BERBASIS CITRA

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

Oleh :

Kartika
NRP. 202 204 005

Tanggal Ujian : 23 Juli 2005
Periode Wisuda : September 2005

Disetujui oleh tim penguji tesis:

1. Ir. Djoko Purwanto, M.Eng. Ph.D (Dosen pembimbing I)
   NIP. 131 879 397
2. Ir. Dadet Pramudihanto, M.Eng. Ph.D (Dosen pembimbing II)
   NIP. 131 803 696
3. Prof. Dr. Ir. Mauridhi Hery P., M.Eng
   NIP. 131 569 364
4. Prof. Dr. Siswandono,MS., Apt
   NIP. 130 809 079
5. Ir. Harris Pirngadi, MT
   NIP. 131 843 903
6. Ir. Totok Mujiono, M.IKom (Penguji)
   NIP. 131 864 106
7. Ir. Hendra Kusuma, M.Eng (Penguji)
   NIP. 131 846 104
8. Rachmad Setiawan, ST., MT (Penguji)
   NIP. 132 134 651

Direktur Program Pascasarjana

Prof. Ir. Happy Ratna S., M.Sc.,Ph.D
NIP. 130 541 829

ABSTRAK

PENGATURAN GERAK PADA ROBOT BERJALAN BERBASIS CITRA

ABSTRAK

Nama Mahasiswa	: Kartika
NRP	: 2202 204 005
Pembimbing	: Ir.Djoko Purwanto,MEng., Ph.D
Co-Pembimbing	: Ir. Dadet Pramudihanto, M.Eng., Ph.D

Pekerjaan mengikuti jalur dan penghindaran rintangan adalah dua perilaku sangat penting yang harus dipertimbangkan pada robot berjalan (*mobile robot*) pada landasan tanpa awak dibawah kendali komputer. Banyak kemajuan yang telah dilaksanakan dalam bidang AMR (*Autonomous Mobile Robot*) pada dekade terakhir ini, dan aplikasi yang telah sukses adalah yang diterapkan pada pertahanan, pertanian dan pertambangan, yang menggunakan sensor jarak dan sensor citra. Pada tesis ini dibahas algoritma pekerjaan mengikuti jalur dan penghindaran rintangan pada purwarupa robot barjalan. Pada robot berjalan digunakan kemudi diferensial dan dilengkapi sebuah kamera sebagai sensor citra yang berfungsi untuk mencari target, menentukan jalur dan mendeteksi rintangan.

Pada penelitian ini dilakukan perbandingan unjuk kerja algoritma *pure-pursuit* dan *follow-the-carrot*. Dari hasil percobaan, dalam hal perilaku mengikuti jalur dan penghindaran rintangan, dibuktikan bahwa algoritma *pure-persuit* menghasilkan unjuk kerja yang lebih baik.

Kata kunci : robot berjalan, sensor citra, follow-the-carrot, pure-pursuit

IMAGE BASED MOTION CONTROL FOR MOBILE ROBOT

ABSTRACT

Student Name : Kartika
Supervisor : Ir.Djoko Purwanto, M.Eng., Ph.D
Co-Supervisor : Ir. Dadet Pramudihanto, M. Eng., Ph.D

Path tracking and obstacle avoidance are two important behaviours that must be considered at an unmanned mobile robot on the ground under computer control. A lot of progress has been done in developing AMR (Autonomous Mobile Robot) in the last decade, and successful has been made in military, agriculture and mining, by using distance and image censors. The research discuss about path tracking and obstacle avoidance algorithm at a mobile robot. The mobile robot use differential steering methode and equipped with a camera as image censor to look for goals, path tracking and obstacle avoidance.

In the research conducted comparison behaviour between pure-pursuit and follow-the-carrot algorithm. From result of attempt, in the case of path tracking and obstacle avoidance behaviour, it is proven that pure pursuit algorithm is better then follow-the-carrot algorithm.

Keywords: mobile robot, image censor, pure pursuit, follow-the-carrot

KATA PENGANTAR

KATA PENGANTAR

Pada kesempatan ini, penulis mengucapkan puji syukur ke hadirat Tuhan Yang Maha Esa yang telah melimpahkan rahmat dan anugerah-Nya sehingga penulis dapat menyelesaikan laporan penelitian berupa tesis ini berjudul "**PENGATURAN GERAK PADA ROBOT BERJALAN BERBASIS CITRA**".

Penyusunan tesis ini untuk memenuhi salah satu syarat untuk menyelesaikan program pendidikan S2 dan memperoleh gelar Magister Teknik (MT) di Jurusan Teknik Elektro, Fakultas Teknologi Industri, Institut Sepuluh Nopember, Surabaya.

Banyak pengalaman berharga yang penulis dapatkan selama pelaksanaan kegiatan penelitian ini, dan berharap laporan berupa tesis ini dapat menjadi masukan bagi dunia ilmu pengetahuan dan penelitian di masa mendatang.

Pada kesempatan ini penulis juga ingin menyampaikan terima kasih yang sebesar – besarnya kepada :

1. Kedua Orang Tua, Istri dan kakak-kakak saya yang telah banyak memberikan semangat dalam menyelesaikan tugas belajar ini.
2. Bapak Ir.Djoko Purwanto M.Eng., Ph.D, selaku dosen pembimbing dan koordinator bidang keahlian Elektronika atas waktu, motivasi dan arahan sehingga selesainya tesis ini.
3. Bapak Ir.Dadet Pramudihanto, M.Eng., Ph.D. selaku dosen pembimbing, yang telah meluangkan waktu dan tenaga sehingga selesainya tesis ini.

4. Prof. Dr. Ir. Mauridhi Hery Purnomo, M.Eng selaku Ketua Program Studi Pascasarjana Teknik Elektro, Fakultas Teknologi, Institut Teknologi Sepuluh Nopember Surabaya.
5. Ibu Prof. Ir. Happy Ratna S., MSc.PhD, selaku Direktur Program Pascasarjana Institut Teknologi Sepuluh Nopember Surabaya.
6. Bapak-bapak dosen, karyawan, teman sejawat dan pihak lainnya.

Yang telah banyak membantu, membimbing dan mengarahkan penulis untuk segera menyelesaikan tesis ini. Penulis mengharapkan kritik dan saran pembaca untuk pengembangan riset tentang robotik selanjutnya.

Surabaya, Juli 2005

Penulis

DAFTAR ISI, GAMBAR & TABEL

DAFTAR ISI

Halaman

Halaman Judul i
Halaman Pengesahan ii
Abstrak iii
Kata Pengantar v
Daftar Isi vii
Daftar Gambar ix
Daftar Tabel x

BAB I PENDAHULUAN
1.1 Latar Belakang 1
1.2 Ruang Lingkup Permasalahan 3
1.3 Permasalahn 4
1.4 Tujuan Penelitian 4
1.5 Metodologi Penelitian 5
1.6 Sistematika Penulisan 5

BAB II TEORI PENUNJANG
2.1 Defenisi dan Pengertian Robot 7
2.2 *Trajectory Planning* 9
2.3 *Mobile Robot* 11
2.3.1 Pengemudian 13
2.3.1.1 *Differential Steering* 13
2.3.1.2 *Ackerman Steering* 21
2.3.2 Kinematik Kemudi Diferensial 22
2.4. *Path Tracking* 26
2.4.1 *Follow-the-Carrot* 26
2.4.2 *Pure Pursuit* 28
2.4.3 Jarak *Look-ahead* 32
2.5 Akuisisi Citra 34
2.5.1 Representasi Citra Digital 35
2.5.2 Pemrosesan Citra Digital 39
2.5.2.1 *Therholding* 39
2.5.2.2 *Greyscaling* 40
2.5.2.3 Filter 40
2.5.2.3.1 Konvolusi 41
2.5.2.3.2 Korelasi 42
2.5.2.4 *Dilation dan Erotion* 43
2.5.2.5 *Color Matching* 44
2.6 *Space* Warna 45
2.6.1 *Red Green Blue* (RGB) 46
2.6.2 *Cyan Yellow Magenta* (CYM) 48
2.6.3 *Hue Saturation Intensity* (HIS) 48
2.6.4 Transformasi RGB ke HSV 49
2.6.5 Transformasi HSV ke RGB 52

2.7 *Pulse With Modulation* (PWM) 53
2.8 Motor DC 56

BAB III PERENCANAAN DAN PEMBUATAN ALAT
3.1 Perencanaan Sistem 60
3.2 Modul Kamera 62
3.3 Central Processing Unit 63
3.4. Perencanaan Software 64
3.4.1 Mode Deteksi Benda 65
3.4.2 Mode Menentukan Target 65
3.4.3 Mode Path Tracking dan Obstacle Avoidance 66
3.5 Perencanaa Hardware 66
3.5.1 Motor DC 66
3.5.2 Driver Motor DC 67
3.5.3 Perencanaan PWM 68

BAB IV ANALISA DAN PENGUJIAN
4.1 Pengujian Jarak Benda 71
4.2 Pengujian Gerakan Robot 76

BAB V PENUTUP 80

Daftar Pustaka 81

DAFTAR GAMBAR

Halaman

2.1 Diagram Unsur *Trajectory Planning* 10
2.2 Jalur roda melalui sebuah putaran 14
2.3 Kecepatan roda yang berbeda 17
2.4 *Ackerman Steering* 22
2.5 Kinematik Kemudi Diferensial 23
2.6 *Follow-the-carrot* 27
2.7 Pendekatan *Pure Pursuit* 29
2.8 Efek dari jarak *look-ahead* 34
2.9 Proses Pencitraan Digital 35
2.10 Digitisasi Spasial 36
2.11 Resolusi gambar 37
2.12 Binerisasi citra *greyscale* 39
2.13 Operasi Konvolusi 42
2.14 Operasi Korelasi 42
2.15 Operasi Dilasi 43
2.16 Operasi *Erotion* 44
2.17 Koordinat Kubus warna sistem RGB 47
2.18 Koordinat Kerucut sistem warna HIS 49
2.19 Sistem warna HSV 50
2.20 Sinyal PWM 53
2.21 Sinyal arus yang melewati Daya 54
2.22 Sinyal PWM dan arus jangkar 55
2.23 Struktur mekanis motor DC magnet permanen 57
2.24 Model Listrik Motor DC 57

3.1 Diagram Blok Sistem 60
3.2 Diagram Alir Sistem 61
3.3 Logitech QuickCam Pro 4000 63
3.4 Nilai piksel dengan Saturation 64
3.5 Implementasi Robot 67
3.6 Rangkaian Driver Motor DC 68
3.7 Sinyal masukan dan output sinyal PWM 69
3.8 Diagram blok Pembangkit PWM 69
3.9 Rangkaian Pembangkit Sinyal PWM 70

4.1 Pengujian Jarak Benda 71
4.2 Posisi Obstacle, Target dan Robot..... 77

DAFTAR TABEL

Halaman

4.1 Data Pengujian Jarak Warna Biru 77
4.2 Data Pengujian Jarak Warna Kuning 78
4.3 Data Pengujian Jarak Warna Hijau 79
4.4 Data Pengujian Jarak Warna Orange 80
4.5 Data Pengujian Jarak Warna Merah 81
4.6 Data Pengujian Gerak robot 78

BAB I
PENDAHULUAN

BAB I
PENDAHULUAN

1.1 Latar Belakang

Pesatnya perkembangan ilmu pengetahuan khususnya dibidang teknologi mekatronika telah banyak mendisain berbagai produk yang memungkinkan manusia lebih nyaman, aman dan effisien dalam melakukan aktifitasnya. Sebagai salah satu produk rekayasa teknologi mekatronika adalah robot yang banyak difungsikan di bidang industri. Disiplin ilmu pengetahuan yang mempelajari robot disebut Robotika. Didalam mengimplementasikan robot ini maka diperlukan gabungan dari beberapa disiplin ilmu matematik, mekanik, elektronik (sensor, mikroprosesor/mikrokontroler) dan teknik informatik khususnya komputer.

Dewasa ini ilmu robot makin berkembang, seiring dengan majunya ilmu komputer, seperti dengan kecepatan dari mikroporsesor yang cepat dan ilmu pengendalian secara otomat, sehingga meningkatkan produktivitas yang tinggi, effisiensi waktu dan tingkat ketelitian yang baik dari robot yang dihasilkan.

Dengan pesatnya inovasi rekayasa teknologi robot dan tuntutan effisiensi kerja, maka robot menjadi salah satu instrumen alternatif untuk membantu tugas-tugas manusia. Karena robot memiliki kemampuan dan ketahanan spesifik sesuai dengan situasi dan kondisi lingkungan tempat robot difungsikan. Salah satu jenis robot yang sering difungsikan untuk melakukan tugas-tugas tertentu adalah yang digunakan untuk mengangkut barang atau benda pada daerah yang sulit, seperti yang digunakan untuk mengangkut kayu di hutan [3]. Dimana pada robot ini dituntut untuk bisa

menghindari rintangan yang terdapat pada jalur yang telah ditentukan sebelumnya. Untuk mendeteksi adanya rintangan didepan robot itu maka pada robot diberikan sebuah kamera.

Pada tesis ini lebih menitikberatkan pada salah satu elemen dasar sebuah robot, yaitu pengolahan citra, yang dapat dianggap seperti sepasang mata pada robot. Mengapa robot pada proyek *mobile robotics* memerlukan mata atau dalam hal ini pengolahan citra ? Jawaban yang umum dan singkat adalah karena robot pada proyek ini sifatnya *mobile,* yakni bergerak dari satu titik ke titik berikutnya, sehingga perlu dideskripsikan bentuk *trajectory* yang akan ditempuhnya (dengan bantuan kamera), bentuk *obstacle* (penghalang) maupun objek yang ditemukan.

Perancangan pengolahan citra sebuah *mobile robot* tidak bisa dilepaskan dari perangkat sensor dan kecerdasan robot. Fitur umum semua robot yang akan dibahas lebih jauh mempunyai kemampuan untuk menerjemahkan sekuen dari rangkaian operasi penangkapan gambar selama alat itu dibutuhkan. Bagaimanpun, semua robot tersebut tidak mampu merasakan dan memberikan tanggapan terhadap perubahan sesaat pada lingkungannya.

Level yang lebih tinggi dari hirarki sebua sensor adalah kemampuan untuk merasakan perubahan sesaat pada lingkungan tersebut yang kemudian diasosiasikan dengan antarmuka dan *artificial intelligent.* Ada dua metode dasar dalam penggunaan sensor, yaitu sensor statik dan sensor siklus tertutup. Umumnya sensor –sensor yang digunakan dalam sistem robotika adalah yang dapat merasakan dan memanipulasi keadaan dengan fungsi matrik dan algoritma *artificial intelligent* (AI). Manipulator tersebut akan terus dalam keadaan statik ketika proses sensing telah dilaksanakan.

Kemudian dapat diawasi gerak yang telah dikerjakan tanpa referensi yang lebih jauh dari sensor. Metode ini mengacu pada sensor statik.

Sebaliknya, dalam sensor siklus tertutup robot dikendalikan oleh alat sensing selama pergerakan menipulator. Kebanyakan sistem *vision* beroperasi pada metode siklus tertutup ini. Disini, sistem *vision* mengawasi faktor koreksi antara posisi aktual robot dengan posisi yang diinginkan. Faktor eror ini kemudian digunakan untuk menggerakkan penggerak robot. Dengan sensor siklus tertutup, bahkan ketika objek masih dalam pergerakan, sebuah robot pasti mampu melakukan transfer aktual yang diinginkan.

Kebanyakan sistem *vision* dilengkapi dengan satu atau lebih kamera yang dihubungkan dengan prosesor *vision*. Prosesor ini kemudian mendigitalisasikan citra yang ditangkap kamera dan manganalisanya untuk mendefenisikan objek yang ditangkap oleh kamera tersebut.

1.2 Ruang Lingkup Permasalahan

Di dunia industri telah banyak difungsikan robot untuk membantu menyelesaikan pekerjaan-pekerjaan yang tidak dapat dikerjakan oleh manusia baik ditinjau dari faktor ekonomis maupun keselamatan kerja. Hal ini dikarenakan robot didisain agar memiliki kemampuan dan ketahanan spesifik sesuai dengan situasi dan kondisi tempat robot difungsikan. Misalnya untuk pekerjaan pengelasan, pemotongan plate logam, pengecatan dan navigasi.

Salah satu robot yang banyak dikembangkan pada jenis robot *outonomous mobile robot* adalah pada sistem visualnya yang dapat melakukan pemetaan secara

langsung dilingkungan yang sedang dijelajahinya serta dapat mendeteksi dan menghindari rintangan yang dilaluinya.

Untuk menghindari rintangan yang ditemukan oleh robot pada saat menjelajah, maka dalam penelitian ini akan diterapkan dua buah algoritma, yaitu *follow-the-carrot* dan *pure pursuit.* Dimana kedua algoritma ini dibandingkan, mana yang lebih baik.

1.3 Permasalahan

Realisasi pengendalian robot meliputi beberapa bagian yaitu pengendalian penggerak dari robot dan pengendalian penghindaran rintangan yang ditemukan pada lintasan yang telah ditentukan serta menemukan objek. Banyak algoritma yang telah digunakan pada sistem robot navigasi, diantaranya algoritma *follow-the-carrot* dan *pure pursuit.* Pada robot berjalan yang hanya menggunakan satu buah sensor kamera, dimana sensor kamera ini digunakan untuk sebagai penentu jalur yang akan dilalui oleh robot, juga sebagai penentu posisi target dan rintangan. Permasalahanya adalah bagaimana dengan hanya menggunakan satu buah sensor ini, robot dapat menentukan jalur jalannya, menghindari rintangan dan menemukan objek.

1.4 Tujuan Penelitian

Tujuan dari penelitian ini adalah untuk mengemplementasikan robot berjalan dengan menggunakan satu buah sensor (kamare), dan membandingkan dua buah algoritma yaitu *follow-the-carror* dan *pure pursuit.* Diharapkan dari hasil penelitian ini dapat ditentukan algoritma mana yang cocok digunakan untuk robot berjalan ini.

1.5 Metodologi Penelitian

Tahapan penyusunan tesis ini dilakukan dengan langkah – langkah sebagai berikut :

- Studi literatur yang meleiputi konsep dasar bentuk model matematik robot berjalan, konsep dasar akuisisi citra digital, algoritma deteksi objek, algoritma tracking serta pengaturan motor DC.
- Realisasi perangkat keras, yang meliputi pembuatan robot berjalan dan kamera sebagai pendukung dari sistem deteksi objek, jalur dan rintangan. Robot berjalan yang direalisasikan adalah robot dengan sistem kemudi diferensial (dua buah motor penggerak tidak saling berhubungan)
- Perancangan dan realisasi perangkat lunak yaitu teknik deteksi dan *path tracking* dengan memanfaatkan kamera digital. Pada realisasi perangkat lunak ini, penulis menggunakan bahasa pemrograman Visual Basic 6, yang digunakan untuk mengolah data *image* .
- Pengujian sistem robot berjalan yang telah dibuat
- Penulisan laporan.

1.6 Sistematika Laporan

Penyusunan laporan tugas akhir ini diawali dengan : Bab I Pendahuluan yang berisi tentang latar belakang, permasalahan , serta metodologi dan ruang lingkup permasalahan. Bab II berisi dasar robot, pemodelan matematik robot berjalan, proses akuisisi data citra digital, algoritma *path tracking* robot berjalan dan pengaturan putaran motor DC. Bab III berisi perancangan perangkat keras dan lunak yang

menyangkut dengan diagram blok dan algoritma pemrograman. Pengujian sistem dan analisis serta batasan kemampuan dari sistem yang diperoleh dari hasil pengujian di tempatkan di Bab IV dan bab V penutup berisi kesimpulan yang diperoleh dari pembuatan tesis, serta saran – saran untuk pengembangan lebih lanjut.

BAB II
TEORI PENUNJANG

BAB II

TINJAUAN PUSTAKA

2.1. Defenisi dan Pengertian Robot

Istilah robot pertama kali dikenalkan oleh Karel Kapeck, yang berarti kerja. Sejak itulah istilah robot banyak digunakan untuk peralatan-peralatan mekanik seperti manipulator lengan robot, mesin-mesin perkakas otomat, robot *medical*. Dalam kamus Webster's mendefinisikan robot sebagai piranti otomatik yang penampilannya berfungsi seperti kemampuan manusia.

Institut Robot Amerika (*Robotics Institut of America*) mendefinisikan robot adalah sebuah manipulator yang dapat diprogram dan mempunyai banyak kegunaan diantaranya untuk memindahkan benda, peralatan atau suatu sistem khusus dengan bermacam-macam gerakan untuk menyelesaikan tugas tertentu.

Berdasarkan sistem koordinatnya, robot diklasifikasikan menjadi empat macam yaitu: *robot Cartesian, robot Spherical , robot Cylindrical.* Ada tiga macam gerakan dasar robot yaitu: *Gerakan Swipe* adalah gerakan rotasi terhadap sumbu longitudinal dari *link* antara dua *joint*, *Gerakan Bending* gerakan rotasi terhadap sumbu potong pada *joint*, *Gerakan Prismatik* adalah gerakan *linier* pada arah *sumbu longitudinal.*

Secara umum komponen utama robot terdiri dari enam bagian yang saling berhubungan satu sama lainnya yaitu: manipulator, aktuator, sensor, sistem kontrol, sistem transmisi dan komputer.

1. **Manipulator,** suatu rangkaian mekanik yang berfungsi untuk melakukan gerakan kinematik robot yang terdiri atas *joint* dan *link*. Gerakan translasi dapat diwujudkan dengan menggunakan pasangan ulir, piston hidrolik atau pneumatik, mekanik teleskopik dan gerakan sepanjang rel. Sedang gerakan rotasi dapat diwujudkan dengan menggunakan motor listrik melalui transmisi roda gigi.
2. **Aktuator,** merupakan peralatan yang memberikan daya kepada manipulator sehingga manipulator dapat melakukan gerakan. Sumber daya untuk menggerakan manipulator dapat diperoleh dari tenaga listrik, hidrolik ataupun pneumatik.
3. **Sensor,** suatu komponen pengindra besaran fisis dari robot. Besaran fisis ini berupa informasi posisi, kecepatan dan rotasi. Hal ini diperlukan agar manipulator dapat bergerak sesuai dengan kondisi yang di-inginkan dengan *error* yang seminimal mungkin.
4. **Transmisi,** *instrument* yang digunakan untuk memindahkan daya dan putaran dari suatu poros ke poros yang lain apabila posisi aktuator tidak satu sumbu dengan manipulator. Sistem transmisi ini diperlukan juga untuk mentransformasikan daya atau putaran sesuai dengan spesifikasi robot yang dikehendaki. Transmisi dapat dilakukan secara mekanik dengan menggunakan pasangan roda gigi atau pasangan *pulley* dan sabuk.
5. **Komputer,** *instrument* yang dianggap sebagai otak dari sebuah robot, berfungsi mengolah data yang digunakan untuk mengendalikan gerakan manipulator berdasarkan algoritma yang diterapkan sebelumya.

6. **Sistem kontrol,** suatu metoda koordinasi dan pengendalian komponen-komponen robot untuk melakukan suatu aktifasi berdasarkan respon input dan output secara simultan dari mulai kondisi awal sampai dengan kondisi akhir.

Berdasarkan metoda kontrolnya robot diklasifikasi menjadi :

1. *Robot control servo* yaitu robot yang menggunakan umpan balik (*feed back*) untuk pengendalian gerakannya (*system control loop*) tertutup.
2. *Robot control non-servo* yaitu robot yang tidak menggunakan umpan balik dalam pengendalian gerakannya (*system control loop*) terbuka.

Dan berdasarkan lintasan (*path*) yang dilalui, robot dapat dibedakan menjadi :

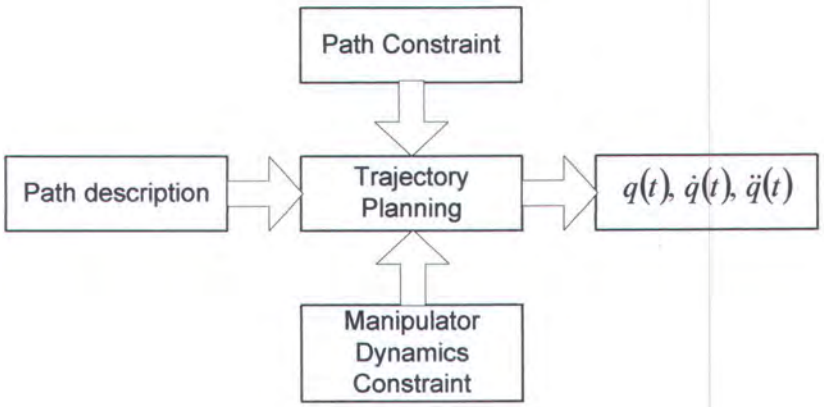
1. *Robot servo point to point* adalah robot servo yang dalam melakukan gerakannya lebih mengutamakan posisi akhir yang dihasilkan dari pada lintasannya.
2. *Robot servo continous path* yaitu robot servo yang memiliki manipulator dapat bergerak mengikuti lintasan tertentu dengan kecepatan sesuai dengan yang dikehendaki.

2.2. Trajectory Planning

Trajectory Planning, perencanaan gerakan manipulator untuk melalui jalur dari titik awal ke titik tujuan dengan *profile* waktu sepanjang jalur.

Trajectory dapat direncanakan pada *joint space* (menentukan waktu evolusi sudut *joint*) dan *cartesian space* (menentukan posisi dan orientasi dari *end frame*).

Secara prinsip, dapat dimengerti bahwa variabel input untuk suatu algoritma perencanaan *trajectory* berupa gambaran lintasan (*path description*), batasan lintasan (*path constraints*), dan batasan kinematika manipulator (*Manipulator kinematics constraints)*. Sedangkan outputnya merupakan *trajectory end-effector* dalam bentuk sebuah fungsi garis lurus dalam domain waktu. Dari turunan fungsi ini dapat diperoleh posisi, kecepatan dan percepatan. Kontribusi dari unsur-unsur di atas dapat diilustrasikan pada Gambar 2.1



Gambar 2.1 Diagram Unsur *Trajectory Planning*

Untuk menghindari kesalahan persepsi didalam mengartikan istilah *path* dan *trajectory*, maka dapat penulis terangkan bahwa *path* adalah menyatakan kedudukan titik-titik (koordinat) yang berurutan dalam *work space* yang harus dilalui lengan robot dalam melakukan gerakan dari konfigurasi awal

sampai dengan konfigurasi akhir. Sedangkan *trajectory* adalah sebuah bentuk gerakan robot dalam *joint space* yang harus dilalui dalam domain waktu secara spesifik, misalnya dalam istilah kecepatan (*velocity*), dan percepatan (*acceleration)* pada setiap titik

Ada dua cara pengendalian lintasan yaitu pengendalian lintasan dari titik ke titik (*point to point path*) dan pengendalian lintasan secara kontinyu *(continous path*). Untuk sistem pengendalian dari titik ke titik cukup ditentukan variabel sendi dari titik awal (*initial point*) dan titik akhir (*final point*). Dengan demikian hanya diperlukan pengaturan kecepatan dan percepatan lintasan. Pada lintasan kontinyu diperlukan pola diskritisasi pola lintasan yang mana sebuah kurva lintasan dibagi menjadi sejumlah titik yang akan dilalui vektor posisi robot.

Model perencanaannya sangat banyak dikarenakan pesatnya perkembangan teknologi saat ini, sebagian diantaranya yang paling umum adalah [9]:

- Model Statik Deterministik: model ini memerlukan informasi umum tentang bentuk jalur dan posisi target yang telah didefenisikan sebelumnya, sehingga dapat digunakan untuk menelusuri jalur dan target yang telah diketahui.
- Model Dinamik Deterministik: model ini memerlukan sedikit informasi lokal yang tidak didefinisikan sebelumnya, sehingga

dapat digunakan untuk menelusuri jalur dan target yang belum diketahui.

- Model Stokastik: model ini memerlukan sedikit informasi lokal yang tidak didefenisikan sebelumnya, model ini dapat mengindentifikasikan objek target yang bergerak dan dapat menelusuri jalur yang berubah.

2.3. *Mobile Robot.*

Mobile robot adalah salah satu jenis robot yang dapat bergerak atau berpindah dari posisi awal ke posisi akhir, oleh karena itu, pada *mobile robot* dilengkapi oleh *end effector* berupa roda yang memungkinkan robot untuk bergerak (*maneuver*) [10]. Dengan mengatur gerakan roda dari *mobile robot* untuk melakukan gerakan yang diinginkan, akan dejelaskan pada bagian berikutnya. Dan juga dari sistem gerakan dari roda ini akan dapat kita menentukan kenematik dari *mobile robot.*

Dalam merencanakan *mobile robot* hal yang harus diperhatikan adalah :

1. *Maneuverability* (gerakan translasi dan perubahan arah); gerakan dari *mobile robot,* yang biasanya disebut dengan maneuver, sangat banyak mendapatkan perhatian dalam merencanakan sebuah *mobile robot,* yaitu untuk mendapatkan gerakan robot yang cepat dan akurat dalam melintasi suatu jalur, baik gerakan translasi maupun perubahan arah.
2. *Controllability ;* kemampuan pengontrolan gerak robot sehingga robot dapat lebih mudah dioperasikan sesuai dengan keinginan.

3. *Traction*; kemampuan *mobile robot* untuk mencengkram sehingga dapat meminimalisasi slip.
4. *Climb*; kemampuan dari *mobile robot* untuk memanjat bidang tertentu.
5. *Stability*; kestabilan dari *mobile robot* mutlak diperlukan untuk pekerjaan yang membutuhkan keakuratan tinggi.
6. *Efficiency*; dalam merencanakan *mobile robot* seharusnya dapat meningkatkan efesiensi dari suatu pekerjaan sehingga dapat mempermudah pekerjaan manusia.
7. *Maintenance*; perawatan yang mudah dan murah, cukup menjadi pertimbangan yang menarik dalam perencanaan sebuah *mobile robot.*
8. *Environment Impact*; dalam merencanakan *mobile robot* diharapkan tidak merusak lingkungan dari areal yang dilalui oleh *mobile robot.*
9. *Navigational Consideration; mobile robot* yang dapat mengemudikan diri sendiri, tanpa memerlukan operator.

2.3.1. Pengemudian

Pengemudian (*steering*) pada *mobile robot* merupakan bagian utama dari *mobile robot* untuk melakukan perpindahan. Jenis – jenis pengemudian pada *mobile robot* adalah sebagai berikut : *differential steering*, *ackerman steering*, *synchro drive*, *tricycle drive* dan *omni directional drive.*

Yang sering digunakan pada *mobile robot* pada umumnya adalah *differential steering dan ackerman steering*, sedangkan untuk jenis yang

lainnya biasa digunakan pada *mobile robot* penelitian yang membutuhkan manuver yang lebih bebas.

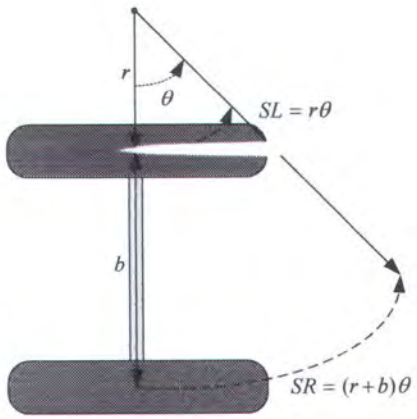
2.3.1.1. Differential Steering.

Sistem kemudi deferensial ini sudah biasa dalam kehidupan sehari – hari, sebab pengaturan pada kursi roda menggunakan sistem ini. Dua buah roda kemudi menjulang pada poros dengan bebas sebagai tenaga dan kontrol, dengan begitu keduanya menyediakan pengarah dan kemudi. Kabanyakan dari kita mempunyai sebuah gerakan intuitif untuk melakukan genggaman dan ini merupakan perilaku dasar dari sistem kemudi deferensial. Jika roda penggerak kedua-duanya berputar sama, maka robot bergerak lurus. Jika roda memutar pada kecepatan sama, tetapi pada arah kebalikan, robot berputar. Mengemudikan robot adalah sekedar bermacam-macam tentang perihal kecepatan roda penggerak itu. Tatapi bagaimana persisnya memilih kecepatan itu, sehingga sedemikian rupa robot akan bergerak dimana kita ingin kan robot itu untuk pergi.

Sebelum mengerjakan model sistem kemudi deferensial, mari untuk mempertimbangkan permasalahan dalam sebuah robot untuk membuat gerakan 90^0 pada suatu gang atau jalan. Sebuah pendekatan sederhana adalah robot tersebut mempunyai kemudi pada saat persimpangan, berhenti dan berputar. Yang dengan jelas, meskipun demikian pendekatan seperti itu tidaklah efesien dan sedikit kaku. Suatu pendekatan yang lebih rapi pada robot untuk mengitari sebuah sudut, mengikuti sebuah busur lingkaran perlahan

lahan sampai persimpangan dengan mempertahankan pada kecepatan yang tetap. Untuk memenuhi cita-cita ini, dengan meningkatkan kecepatan roda terluar dari lingkaran dan memperlambat bagian roda dalamnya.

Apakah jalur yang diikuti robot sungguh-sungguh lingkaran, atau apakah itu merupakan semacam kurva. Akan menunjukkan bahwa jika roda kedua-duanya berputar tetap, tetapi berbeda kecepatannya, tentu robot mengikuti sebuah lintasan lingkaran. Sekarang, baru saja berasumsi bahwa jalur yang diikuti adalah lingkaran. Lalu dengan melihat kecepatan roda itu dipilih berdasarkan pada bagaimana keliling dari sebuah putaran yang diinginkan untuk dilalui robot tersebut.



Gambar 2.2. Jalur roda melalui sebuah putaran.

Dari gambar 2.2, dan mengamati hubungannya didapat rumus :

$$SL = r\theta$$

$$SR = (r+b)\,\theta$$

$$SM = (r + b/2)\theta \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (1)$$

Dimana SL, SR menunjukkan perpindahan/jarak (jarak yang ditempuh selama perjalanan) karena secara berturut roda kanan dan kiri adalah radius putaran untuk roda bagian dalam (kiri), jarak antara roda (dari roda kanan ke roda kiri), dan sudut putar pada lingkaran $\theta radian = \theta derjat\ (\pi/180)$. SM adalah kecepatan titik tengah pada poros sumbu utama. Pada masalah ini, akan diperlukan titik pusat poros sumbu lain sebagai kerangka asal acuan dari simulasi robot sebagai referensi.

Pada permasalahan diatas, mempertimbangkan permasalahan dalam usaha untuk memilih kecepatan roda yang berdasarkan pada suatu jalur yang diinginkan. Sekarang dengan mempertimbangkan kebalikan masalah itu: bagaimana cara menemukan *trajectory* dari robot jika diberikan kecepatan dari roda apa yang akan berputar? Jawaban dari pertanyaan ini mengarahkan kearah pengembangan dari model tersebut dan ternyata juga perhitungan sebenarnya secara langsung untuk dapat dipergunakan. Yang sangat utama, adalah sebuah masalah didalam apa yang disebut dengan bagian kenematika, teknik prediksi sebuah perilaku sistem dasar mekanik pada masukan untuk sistem itu.

Mengembangkan sebuah model sistem kemudi deferensial tidaklah sulit, tetapi memerlukan penggunaan kalkulus dan persamaan deferensial. Untuk memulai, bahwa dengan mempertimbangkan apa yang tertarik akan bagaimana koordinat x dan y yang akan berubah terhadap waktu. Pada saat tertentu, koordinat x dan y titik pusat robot berubah berdasarkan pada

kecepatan dan orientasi dari robot tersebut. Perlakukan orientasi sebagai suatu sudut yang diukur dalam radian, yang berlawanan arah dengan putaran jarum jam pada sumbu x. Ingat bahwa garis panah vektor memberikan arah gerakan maju untuk robot akan disederhanakan dengan cos θ, sin θ. Koordinat x dan y untuk titik pusat dari robot akan berubah tergantung pada kecepatan gerakannya sepanjang garis vektor.

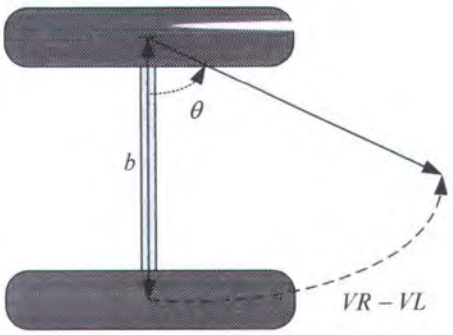
Pengamatan ini menyatakan bahwa, dengan mengambil $m(t)$ *dan* $\theta(t)$ yang tergantung fungsi waktu untuk kecepatan dan orientasi dari robot, diperoleh rumus :

$$dx / dt = m(t) \cos(\theta(t))$$

$$dY / dt = m(t) \sin(\theta(t)) \quad \text{......} \quad (2)$$

Dari persamaan (2) diperoleh fungsi untuk *trajectory* dari robot tersebut. Sekarang, akan jadi lebih mudah untuk berpikir tentang percepatan dan hanya berasumsi bahwa kecepatan roda adalah tetap. Jika kedua rodanya sedang bergerak pada kecepatan yang sama, maka robot akan bergerak lurus. Maka, sebagai gantinya, kita akan mempertimbangkan kasus itu jika roda berputar pada kecepatan yang berbeda.

Dari gambar 2.3 diperoleh, ketika posisi perubahan robot, semua titik – titik pada robot mungkin sedang bergerak. Untuk mengembangkan sebuah persamaan kinematik untuk gerakan robot maju pada sistem kemudi deferensial, diawali dengan menetapkan sebuah kerangka acuan dimana sebuah titik dipilih untuk digunakan sebagai titik awal pada keperluan ini.



Gambar 2.3. Kecepatan roda yang berbeda.

Dan titik yang lainnya didalam sistem diperlakukan sebagai titik yang bergerak seperti yang ditunjukan oleh acuan. Robot dianggap sebagai suatu benda yang kaku. Dengan pendekatan masalah didalam cara ini, akan memperoleh pengertian yang mendalam untuk dapat diberlakukan semakin umum permasalahannya dalam memperagakan gerakan robot itu dalam sebuah kerangka yang mutlak sebagai acuan.

Titik yang dipilih sebagai referensi adalah titik pusat roda yang kiri. Ini adalah titik dimana suatu roda diasumsikan menyentuh lantai. Semua gerakan pada kerangka ini untuk direferensi yang menunjukkan sesungguhnya relative terhadap gerakan roda kiri. Sebab roda yang kanan tegak lurus terhadap poros sumbu, gerakannya dalam kerangka referensi yang mengikuti suatu busur lingkaran dengan radius yang sesuai dengan panjang poros sumbu.

Sekarang titik tengah sendiri mungkin sedang bergerak, sehingga jalur yang pasti dari roda yang kanan tidak akan perlu sesuai dengan busur lingkaran tertentu tersebut. Tetapi perubahan didalam orientasi tidaklah

terbatas pada kerangka referensi robot itu. Sebab diperlakukan robot sebagai benda yang kaku, semua titik-titk didalam sistem mengalami perubahan yang sama didalam orientasi. Jika kita putar robot tersebut sebesar 10 derajat kearah kiri dari roda robot (titik tengah roda kiri), maka semua titik-titik juga mengalami perubahan 10 derajat sesuai dengan perubahan orientasinya. Dan dimanapun perubahan didalam orientasi kerangka referensi yang khusus merupakan hal yang sesungguhnya, dan merupakan kasus yang semakin umum.

Berdasarkan pada pengamatan ini, dapat diperoleh sebuah persamaan diferensial yang menggambarkan perubahahan itu didalam orientasi ketika perubahan tersebut terhadap waktu. Defenisi dari sebuah sudut dalam radian adalah panjang sebuah busur lingkaran dibagi dengan radius lingkaran tersebut. Kecepatan relatif roda bagian kanan adalah panjang busur per unit waktu. Panjangnya dari roda kemudi ketitik pusat adalah radius. Kombinasi dari fakta ini, diperoleh persamaan :

$$d\theta / dt = (VR - VL)/b \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (3)$$

Integral dari persamaan (3) dan mengambil orientasi awal dari robot sebagai $\theta\,(0) = \theta 0$, diperoleh sebuah fungsi untuk menghitung orientasi robot sebagai fungsi kecepatan roda dan waktu :

$$d(t) = (VR - VL)t/b + \theta 0 \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (4)$$

Seperti diatas, perubahan ini didalam orientasi berlaku juga bagi kerangka referensi mutlak. Perhatian pergeseran kembali ke bingkai yang *absolute*, perlu mengingatkan persamaan (2) diatas yang mana robot secara keseluruhan bergerak tergantung pada kecepatan titik pusatnya (titik tengah dari sumbu). Kecepatan rata-rata untuk dua buah roda, atau kombinasi fakta ini dengan apa yang diketahui sekitar orientasi sebagai fungsi waktu, dan diperoleh persamaan diferensial yang berikut ini :

$$dy/dt = [(VR + VL)/2]\cos(\theta(t))$$

$$dy/dt = [(VR + VL)/2]\sin(\theta(t)) \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (5)$$

Dimana pada persamaan (5) adalah dalam format yang sama dengan persamaan yang ditunjukkan pada persamaan (2). Dengan pengintegrasian dan menetapkan posisi awal robot itu $x(0) = x0$, $y(0) = y0$, diperoleh :

$$x(t) = x0 + \frac{b(VR+VL)}{2(VR-VL)}\left[\sin\left((VR-VL)t/b + \theta 0\right) - \sin(\theta 0)\right]$$

$$y(t) = y0 + \frac{b(VR+VL)}{2(VR-VL)}\left[\cos\left((VR-VL)t/b + \theta 0\right) - \cos(\theta 0)\right] \quad \ldots\ldots\ldots\ldots \quad (6)$$

Persamaan (6) memberikan konfirmasi pernyataan yang lebih awal, ketika roda memutar pada kecepatan tetap, maka robot akan mengikuti sebuah lintasan lingkaran. Dimana, $(b/2)(VR+VL)/(VR-VL)$ adalah radius putaran untuk *trajectory* dari pusat lingkaran robot.

Dari persamaan (6) bisa digunakan dengan mengganti SR, SL dengan istilah VR, VL, kemudian pemecahan untuk nilai – nilai x, y, dan θ. Dengan

menggunakan persamaan itu, diperlu secara seksama pada kasus dimana robot begerak hampir mendekati garis lurus dan jarak SR, SL menjadi hampir sama yang menghasilkan nilai-nilai yang mendekat nol pada penyebutnya.

Untuk menghindari kesulitan dalam menggunakan persamaan (6), Johan Borenstein, et al., mengusulkan persamaan (7) cocok digunakan untuk aplikasi robot kecil.

$$\begin{aligned} \bar{s} &= (SR + SL)/2 \\ \theta &= (SR - SL)/b + \theta 0 \\ x &= \bar{s}\cos(\theta) + x0 \\ y &= \bar{s}\sin(\theta) + y0 \end{aligned} \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (7)$$

Persamaan (7) bisa bekerja dengan baik sepanjang algoritma yang digunakan. Yang sangat utama, persamaan (7) menghitung total perputaran robot dan jarak tempuh yang dilalui oleh robot.

Penambahan percepatan pada model ini adalah hal yang mudah yaitu dengan mensubsitusikan persamaan *time-dependent* sesuai dengan percepatan dari roda ke dalam persamaan (3) dan (5). Dengan memberikan percepatan pada kedua roda, dimana ini memberikan nilai-nilai untuk sebuah percepatan tetap (atau turunnya kecepatan) dan memberikan kecepatan awal. Dengan mensubsitusikan persamaan (3) dan (5) menghasilkan :

$$A = (\alpha R + \alpha L)/2$$
$$B = (WR + WL)/2$$
$$C = (\alpha R - \alpha L)/2b$$
$$D = (WR - WL)/b \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (8)$$
$$\theta(t) = Ct^2 + Dt + \theta 0$$
$$dx/dt = (At + B)\cos(Ct^2 + Dt + \theta 0)$$
$$dy/dt = (At + B)\sin(Ct^2 + Dt + \theta 0)$$

2.3.1.2. *Ackerman Steering*

Pada *Ackerman steering*, sudut belok robot (θ_A) dapat dihitung sebagai berikut:

$$Cot\theta_R - Cot\theta_L = \frac{d}{l} \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (9)$$

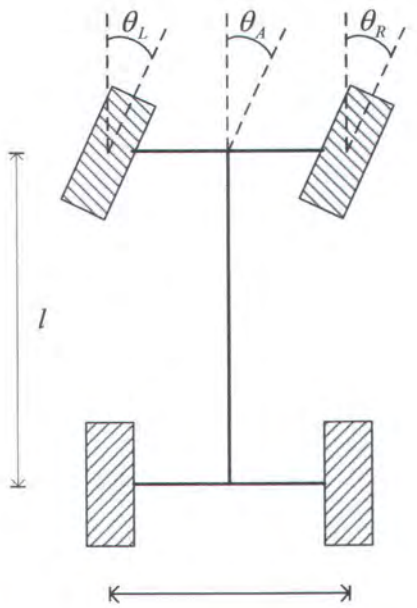
$$Cot\theta_A = \frac{d}{2l} + Cot\theta_R \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (10)$$

$$Cot\theta_A = Cot\theta_L - \frac{d}{2l} \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (11)$$

Dimana:

θ_R = Sudut kemiringan antara roda sisi dalam dan garis normal bodi robot
θ_L = Sudut kemiringan antara roda sisi luar dan garis normal bodi robot
θ_A = Sudut belok *Ackerman Steering*
d = Jarak antara roda sisi luar dan roda sisi dalam
l = Jarak antara roda depan dan roda belakang

Untuk lebih jelasnya dapat dilihat pada gambar 2.4 berikut:

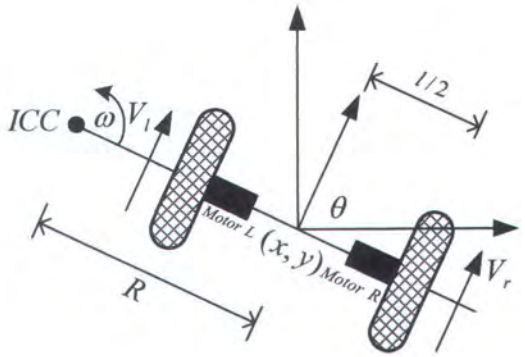


Gambar 2.4. *Ackerman Steering*

2.3.2. Kinematik Kemudi Diferensial

Teori kinematik kemudi diferensial adalah sungguh sederhana. Setiap roda mobile robot dalam keadaan gerakan harus selalu berputar pada suatu titik yang berada di suatu tempat pada porosnya, yang umumnya dua buah roda [6]. Titik ini sering disebut dengan pusat lengkungan seketika *(instantaneous centre of curvature* (ICC)), atau pusat perputaran seketika (*instantaneous centre of rotation* (ICR)). Berubahnya posisi ICC hanya dengan berubahnya kecepatan dari kedua roda. Dengan menggunakan motor penggerak yang terpisah, masing-masing motor dengan satu buah roda, dengan demikian dapat diberikan kecepatan sendiri untuk mengendalikan roda yang kanan dan yang kiri. Dengan memiliki ini bisa memungkinkan dari robot untuk memilih *trajectory* dan jalan alur yang berbeda. Sebuah gambar yang menggambarkan

kinematika pada suatu kemudi diferensial dari *mobile* robot dapat dilihat pada gambar 2.5.



Gambar 2.5. Kinematik Kemudi Diferensial

Masing – masing roda mengikuti suatu alur di lantai pada ICC dengan tingkat sudut ω yang sama, seperti :

$$\omega\left(R + 1/2\right) = Vr \quad \text{..... (12)}$$

$$\omega\left(R - 1/2\right) = Vl \quad \text{..... (13)}$$

dimana l adalah jarak antara kedua roda, roda yang kanan bergerak dengan kecepatan VR dan roda yang kiri bergerak dengan kecepatan VL. R adalah jarak dari ICC ke *Midpoint* antara roda. Semua parameter kontrol ini adalah fungsi waktu, diperoleh :

$$R = 1/2 * \left(Vl + Vr\right)/\left(Vr - Vl\right) \quad \text{..... (14)}$$

$$\omega = (Vr - Vl)/l \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (15)$$

Dengan mempunyai dua kasus khusus yang datang dari persamaan diatas. Jika V*l* = Vr, kemudian radius R adalah tanpa batas dan robot berpindah dengan lurus. Jika V*l* = - Vr, kemudian radius adalah nol dan robot berputar pada tempatnya. Dalam semua kasus lain robot berpindah pada *trajectory* disekitar ICC dengan beberapa tingkat sudut ω.

Persamaan kinematik gerakan maju dapat diperoleh dengan mudah. Fokusnya adalah pada bagaimana koordinat x dan y dan dengan orientasi yang berubah berdasarkan dengan waktu [6]. Misalkan sudut θ sebagai sudut orientasi, pengukuran dalam radian, dan searah jarum jam dari sumbu X. Jika dimisalkan m(t) dan θ(t) dijadikan dengan fungsi waktu yang mana mewakili kecepatan dan orientasi dari robot, sehingga diperoleh rumus dibawah ini :

$$dx/dt = m(t)\cos(\theta(t)) \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (16)$$

$$dy/dt = m(t)\sin(\theta(t)) \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (17)$$

Perubahan orientasi yang berkenaan dengan waktu adalah sama seperti ketika nilai sudut ω. Oleh karena itu :

$$d\theta/dt = \omega = (Vr - Vl)/l \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (18)$$

Penggabungan persamaan ini menghasilkan sebuah fungsi untuk orientasi robot dengan respon waktu. Orientasi awal robot adalah θ(0) dan digantikan dengan θ_0 :

$$\theta(t) = (Vr - Vl)t/l + \theta_0 \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (19)$$

Karena fungsi dari kecepatan pada persamaan (16) dan (17) diatas sama dengan kecepatan rata-rata untuk kedua roda, adalah $m(t) = (Vr + Vl)/2$, dengan menggabungkan dengan persamaan (16) dan (17) diperoleh :

$$dx/dt = [(Vr - Vl)/2]\cos(\theta(t)) \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (20)$$

$$dy/dt = [(Vr - Vl)/2]\sin(\theta(t)) \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (21)$$

Langkah terakhir adalah untuk menggabungkan persamaan (20) dan (21), dan dengan mempertimbangkan posisi awal $x(0) = x_0$ dan $y(0) = y_0$ diperoleh :

$$x(t) = x_0 + 1/2(Vr + Vl) \;/\; (Vr - Vl)\;[\sin((Vr - Vl)\,t/l + \theta_0) - \sin(\theta_0)]$$
$$\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (22)$$

$$y(t) = x_0 + 1/2(Vr + Vl) \;/\; (Vr - Vl)\;[\cos((Vr - Vl)\,t/l + \theta_0) - \cos(\theta_0)]$$
$$\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (23)$$

Dimana, $1/2(Vr + Vl) \;/\; (Vr - Vl) = R$, radius berputar dari robot, dan bahwa $(Vr + Vl)/l = \omega$, persamaan (22) dan (23) bisa disederhanakan menjadi :

$$x(t) = x_0 + R\;[\sin(\omega t + \theta_0) - \sin(\theta_0)] \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (24)$$

$$x(t) = y_0 - R\;[\cos(\omega t + \theta_0) - \cos(\theta_0)] \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (25)$$

Ini merupakan teori yang berbeda dari yang diterapkan pada pada roda *mobile* robot dengan menggunakan kemudi diferensial. Satu-satunya yang harus dilakukan adalah untuk menggantikan istilah Vr dan V*l* dengan Sr dan

Sl, ini menandakan perhitungan jarak bukan untuk mempercepat, dan sebagai hasilnya juga dalam nomain waktu t. Disini Sr dan S*l* adalah jarak tempuh perjalanan dari roda kanan dan roda kiri. Akhirnya persamaan (22) dan (23) menjadi :

$$x(t) = x_0 + 1/2(Sr + Sl) / (Sr - Sl) \left[\sin\left((Sr - Sl)/l + \theta_0\right) - \sin(\theta_0)\right] \quad \ldots\ldots (26)$$

$$y(t) = y_0 - 1/2(Sr + Sl) / (Sr - Sl) \left[\cos\left((Sr - Sl)/l + \theta_0\right) - \cos(\theta_0)\right] \quad \ldots\ldots (27)$$

yang merupakan persamaan kenematik untuk bergerak maju yang digunakan oleh robot dengan kemudi diferensial ketika memutar.

2.4. Path Tracking

Path tracking adalah proses yang berhubungan dengan bagaimana cara menentukan kecepatan dan menentukan arah kemudi pada waktu tertentu pada saat robot mengikuti sebuah *path* yang telah ditentukan. Sebuah *path* terdiri dari satu set titik-titik yang mewakili posisi yang tergantung pada rute koordinat. Sering ketika menerapkan suatu algoritma *path tracking*, satu instrumen yang lainnya juga digunakan untuk merekam *path* tersebut pada semua titik koordinat yang dilalui.

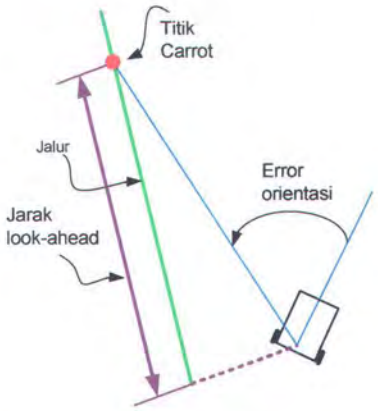
Algoritma *path tracking* harus bisa untuk mengatasi penyimpangan atau yang tidak direncanakan dari *path*, seperti penyimpangan dapat disebabkan oleh kesalahan odometrik atau oleh rintangan yang baru terjadi pada *path* yang harus

dihindarkan. Ada banyak jenis algoritma yang digunakan untuk *path tracking* yang tersedia saat ini. Diantaranya; *follow-the-carrot* dan *pure pursuit*.

2.4.1. Follow-the-carrot

Algoritma ini berdasarkan pada sebuah gagasan yang sangat sederhana. Dengan memperoleh suatu titik tujuan, kemudian mengarahkan *mobile* robot ke arah yang dituju tersebut [8]. Gambar 2.6 menguraikan latar belakang dasar dari metoda ini.

Sebuah garis ditarik dari pusat sistem koordinat dari *mobile* robot yang tegak lurus terhadap *path*. Titik *carrot* (titik tujuan), kemudian didefenisikan sebagai titik yang terdapat pada *path*, suatu jarak kemuka (*look-ahead)* yang menjauh dari titik persimpangan dari garis ini. Parameter yang paling utama adalah kesalahan orientasi, yang didefenisikan dengan sudut antara depan robot dan garis dari pusat sistem koordinat robot pada titik *carrot*.



Gamabar 2.6. *Follow-the-carrot*

Sebuah kendali proporional digunakan untuk memperkecil kesalahan orientasi itu antara robot dan titik carrot. Pada saat kesalahan orientasi nol maka robot sedang menuju persis ke arah titik carrot. Besarnya putaran φ didefenisikan dengan :

$$\varphi = k_p * e_o \quad \text{..... (28)}$$

dimana k_p adalah nilai penguatan proposional dan e_o adalah kesalahan orientasi. Satu hal mempunyai kemungkinan meningkatkan ketelitian pengontrol, barangkali dengan menambahkan fungsi derevativ atau integrativ untuk hal ini. Bagaimanapun, kontrol yang sederhana masih paling sering digunakan untuk algoritma ini.

Walapun pendekatan *follow-the-carrot* mudah untuk dipahami dan sangat sederhana dalam penerapannya, tetapi masih mempunyai sepasang kelemahan utama. Yang pertama, robot mempunyai suatu kecendrungan untuk secara alami memotong sudut. Ini terjadi disebabkan robot yang dengan seketika mencoba untuk memutus ke arah masing-masing titik *carrot* yang baru. Kelemahan yang lain dengan teknik *path tracking* ini adalah bahwa robot bisa bergerak–gerak pada *path*, terutama sekali didalam kasus pada saat jarak *look-ahead* yang kecil atau pada saat kecepatan sedang tinggi.

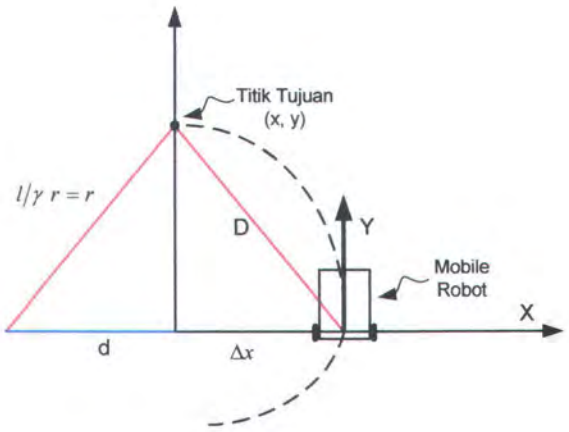
Bagaimana modifikasi yang dapat dibuat pada algoritma ini untuk meningkatkan ketelitian dan efesiensinya. Pemilihan sudut kemudi dapat dimodifikasi sebagai dasar untuk mengatasi masalah keduanya, penggantian jarak kesalahan yang tergantung pada posisi yang tegang lurus pada *path* dan

kesalahan orientasi. Namun kerugian–kerugian dari metoda ini biasanya besar, membuat sia-sia untuk diimplementasi didalam proyek yang memerlukan kemampuan *path tracking* yang baik. Meski demikian adalah pantas untuk tujuan bidang pendidikan, atau untuk perbandingan dengan algoritma *path tracking* yang lainnya.

2.4.2. Pure Pursuit

Konsep dari pure pursuit adalah untuk mengkalkulasi lengkungan yang akan diambil robot itu dari posisi yang sekarang ke posisi tujuannya [2],[3],[7]. Titik tujuan ditentukan sama dengan metoda algoritma *follow-the-carrot*. Kemudian sebuah lingkaran yang digambarkan sedemikian sehingga kedua-duanya lewat melalui titik tujuan dan posisi dari robot saat sekarang. Akhirnya sebuah algoritma kontrol memilih sebuah sudut kemudi dalam hubungan dengan lingkaran ini. Sesungguhnya, robot merubah lengkungannya ini dengan berulang kali mengukur busur lingkaran dengan baik, yang selalu dengan mendorong maju kearah titik tujuan itu.

Adalah hal yang penting untuk dicatat bahwa uraian algoritma *pure pursuit* seperti yang ditunjukkan pada gambar 2.7 pada sistem koordinat robot. Sistem koordinat robot, dimana sumbu y didefenisikan sebagai arah gerakan maju dari robot, sedangkan sumbu z adalah kearah bawah dan sumbu x adalah arah kiri dan kanan pada sistem koordinat. Oleh karena itu semua koordinat yang digunakan pertama harus diubah ke koordinat robot terlebih dahulu supaya algoritma dapat bekerja dengan baik.



Gambar 2.7. Pendekatan *Pure Pursuit*

Sungguh beruntung agak lurus kedepan kearah lokasi konversi koordinat dalam sebuah sistem ke dalam reperesentasi pada sistem yang lainnya. Misalkan (xr,yr) adalah posisi robot sekarang, dan (xg,yg) adalah titik tujuan yang akan dirubah kedalam koordinat robot. Dimana :

$$xg_v = (xg - xr)\cos(\Phi) + (yg\text{-}yr)\sin(\Phi) \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (29)$$

$$yg_v = -(xg - xr)\sin(\Phi) + (yg\text{-}yr)\cos(\Phi) \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (30)$$

dimana (xgv,ygv) adalah titik tujuan koordinat robot dan Φ adalah depan dari robot yang sekarang. Pada gambar diatas D didefenisikan sebagai jarak *mobile robot* saat ini dan titik tujuan. Δx adalah selisih nilai x dari titik tujuan ke titik saat ini, dan $1/\gamma_r$ adalah radius lingkaran yang dilewati pusat robot dan titik tujuan. Lengkungan yang diperlukan *mobile robot* dihitung dengan :

$$\gamma_r = 2\Delta x/D^2 \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (31)$$

Rumusan derevativ ini didasarkan hanya pada dua persamaan sederhana (1) :

1) $x^2 + y^2 = D^2$

2) $x + d = r$

(x,y) disebut koordinat dari titik tujuan seperti yang ditunjukkan gambar 2.7 diatas. Persamaan yang pertama adalah suatu hasil penerapan dalil *Pythagoras* pada segitiga siku-siku pada gambar yang sama, dan persamaan yang kedua datang dari penjumlahan segmen garis yang terdapat pada sumbu X.

Persamaan (32) berikut ini diperoleh dari gambar 2.7. :

$d = r - x$ ..... (32a)

$(r - x)^2 + y^2 = r^2$ ..... (32b)

$r^2 - 2rx + x^2 + y^2 = r^2$ ..... (32c)

$2rx = D^2$ ..... (32d)

$r = D^2/2x$..... (32e)

$\gamma_r = 2x/D^2$ ..... (32f)

Langkah yang terakhir berasal dari fakta matematik suatu lingkaran mempunyai suatu lengkungan tetap yang mana berbanding terbalik dengan radiusnya.

Nama *pure pursuit* berasal dari gambaran dari sistem metoda ini. Dengan melukiskan sebuah gambaran didalam pikiran tentang robot yang mengejar titik tujuan yang ditunjukkan oleh *path* pada suatu jarak yang digambarkan berada didepannya (yang selalu mengejar titik tujuan itu). Ini dapat juga diabalogikan dengan cara manusia dalam mengemudikan mobilnya. Dimana pada umumnya memperhatikan beberapa jarak didepan dari mobil, dan akan mengejar tujuan tersebut.

Dengan mengasumsikan titik-titik yang dibuat oleh *path* itu dan posisi yang berurtan dari robot yang mempunyai sistem koordinat yang sama, algoritma *pure pursuit* dapat diuraikan dengan langkah – langkah sederhana berikut ini :

1. Memperoleh posisi sekarang dari robot.
2. Menemukan titik tujuan:
    2.1 Menghitung titik pada *path* terdekat ke robot (xc, yc)
    2.2 Menghitung jarak *look ahead D*
    2.3 Memperoleh titik tujuan dengan memindahkan jarak D diatas *path* dari titik (xc,yc)
3. Mengubah titik tujuan pada koordinat robot.
4. Menghitung lengkungan yang diinginkan dari robot $\gamma = 2\Delta x/D2$.
5. Menggerakkan robot ke arah titik tujuan dengan lengkungan yang diinginkan.
6. Memperoleh posisi yang baru. (Kembali ke 2)

Teknik *pure pursuit* menampilkan hasil yang lebih baik dibandingkan dengan metoda *follow-the-carrot* seperti yang diuraikan lebih awal. Satu peningkatan adalah lebih sedikit kasus osilasi pada posisi yang besar dan sedikit kesalahan, yang lainnya ditingkatkan ketelitian *path tracking* pada kurva. Oleh karena keuntungan metoda ini maka lebih sering digunakan untuk aplikasi yang sebenarmya dibandingakn dengan algoritma *follow-the-carrot*. Optimalisasi metoda yang lain boleh menunjukkan hasil *path tracking* yang

lebih baik dalam beberapa area, tetapi secara keseluruhan adalah algoritma yang terbaik yang dimanfaatkan

2.4.3. Jarak *look-ahead*

Umumnya semua metoda *path tracking* menggunakan titik *look-ahead*, yang mana terdapat suatu titik pada jalur dengan jarak tertentu sebesar L yang menjauh dari proyeksi orthogonal dari pada posisi *mobile* robot pada jalur tersebut. Teknik *path tracking* yang menggunakan titik *look-ahead* ini disebut dengan algoritma geometris. Dengan mengubah jarak *look-ahead* dapat menghasilkan suatu efek yang penting pada pencapaian algoritma itu. Dengan menaikkan nilai L cenderung untuk mengurangi banyaknya goyangan, dengan begitu memastikan *path tracking* dengan halus untuk mencapai tujuan.

Bagaimanapun juga ini akan menyebabkan *mobile robot* itu untuk memotong sudut, yang menyebabkan mengurangi ketepatan dari *path tracking* pada jalur. Efeknya dengan mengubah parameter ini pada akhirnya akan menjadi suatu pertanyaan didalam kontek apa yang diinginkan dari suatu alat mengikuti jejak untuk menghasilkan yang baik. Ada dua permasalahan yang perlu untuk dipertimbangkan :

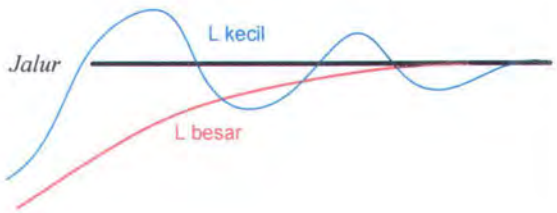
1. Memperoleh kembali suatu jalur
2. Mempertahankan jalur.

Masalah yang pertama adalah terjadi ketika *mobile robot* memulai jalan pada jalur, dengan begitu tergantung pada kesalahan posisi dan orientasi

yang besar, dan sedang berusaha untuk kembali ke jalur yang sebelumnya. Dalam keadaan ini agak jelas efek yang akan terjadi jika dilakukan perubahan terhadap jarak *look-ahead.*

Nilai yang besar dari *look-ahead* akan menyebabkan *mobile robot* akan berhasil dengan halus dan dengan lebih sedikit goyangan. Disisi lain, pada saat kembali lagi ke pada jalur dengan nilai *look-ahead* yang besar menyebabkan hasil yang lebih buruk, terutama pada kasus jalur yang mempunyai belokan dengan tiba-tiba. Maka sesuatu hal yang sulit untuk harus memilih jarak *look-ahead* pada sebuah algoritma *path tracking* tertentu.

Faktor lain yang harus dipertimbangkan ketika memilih jarak *look-ahead* L adalah kecepatan dari *mobile robot*. Pada saat peningkatan kecepatan pada *mobile robot* juga memerlukan jarak L yang ditingkatkan. Alasannya adalah kecepatan yang lebih tinggi memerlukan start awal untuk langkah berbelok atau menghindar.



Gambar 2.8. Efek dari jarak *look-ahead.*

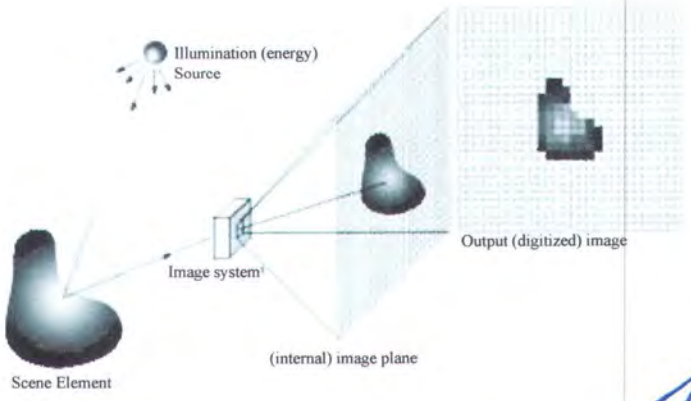
Yang biasanya *mobile robot* selalu mulai memutar sebelum kurva pada setiap kurva dari jarak *look-ahead* yang lebih besar dari nol. Ini

merupakan kualitas yang positif menggantikan kerugian waktu yang diperlukan dalam tahap pelaksanaan putaran. Kemampuan *path-tracking* yang baik pada kecepatan tinggi menjadi penting sekali, dan oleh karena itu suatu prioritas yang sangat diperhatikan untuk aplikasi *mobile robot* yang sebenarnya.

2.5. Akuisisi Citra

Proses akuisisi citra digital adalah proses yang paling penting pada pencitraan karena proses ini sangat menentukan kualitas dari citra digital yang akan diperoleh. Agar proses akuisisi citra digital dapat terjadi diperlukan tiga komponen utama yang harus dipenuhi yaitu sumber cahaya, obyek atau benda yang akan diamati dan sensor peka cahaya (kamera). Sedangkan proses akuisisi citra digital tersebut dapat dijelaskan sebagai berikut, cahaya yang mengenai permukaan suatu benda atau obyek 3 dimensi akan dipantulkan ke segala arah, seperti yang terlihat pada gambar 2.9.

Pantulan cahaya ini sebagian ditangkap oleh sensor peka cahaya pada kamera. Intensitas cahaya yang diterima oleh sensor merepresentasikan kondisi obyek tersebut. Sehingga citra digital yang diperoleh merupakan informasi tentang obyek yang terbentuk dari pantulan cahaya atau refleksi pada permukaan obyek [11].



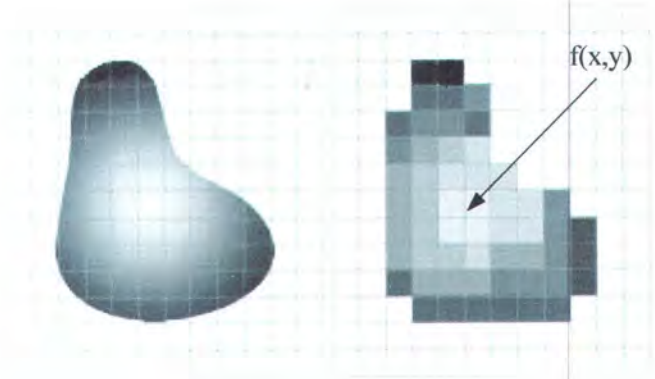
Gambar. 2.9. Proses Pencitraan Digital

2.5.1. Representasi Citra Digital

Kalau diperhatikan gagasan sebuah citra digital dari sudut pandang yang sederhana, citra sebagai fungsi dua dimensi pada koordinat spasial (x,y), harga-harga tersebut mendefinisikan suatu ukuran intensitas pada titik tersebut.

a. Digitisasi Spasial

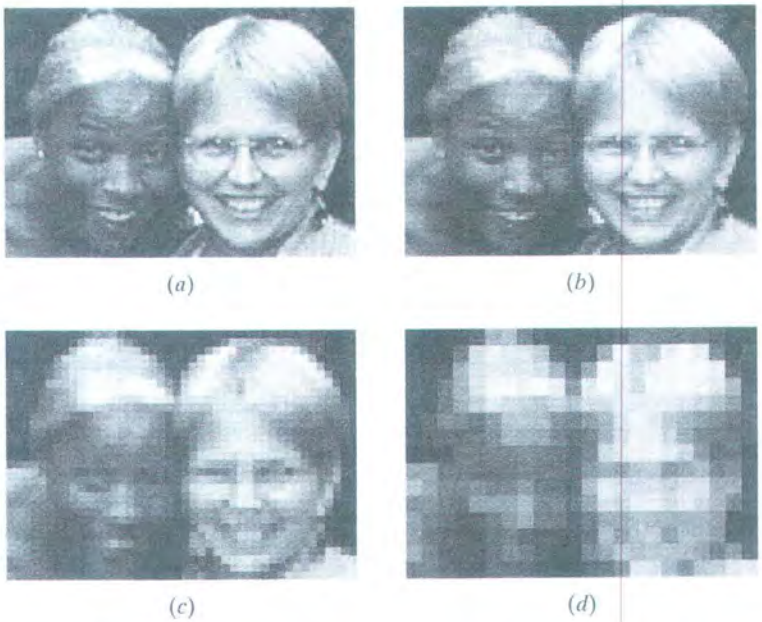
Pada digitisasi spasial citra analog disampling pada titik diskrit. Setiap sampel disebut sebagai *picture cell* atau *pixel.* Citra digital tersebut disampling menurut kriteria Nyquist. Dalam hal ini frekuensi sampling menentukan frekuensi spasial maksimum yang dapat direkonstruksi ulang secara akurat pada citra digital [12].



Gambar 2.10. Digitisasi Spasial

Citra digital merupakan hasil sampling dari citra analaog yang besar samplingnya ditentukan dalam deret dua dimensi. Elemen dari citra digital dinyatakan dalam satuan *pixel*. Standard sampling kamera digital memenuhi ratio 4:3. Saat ini ukuran citra digital yang dihasilkan oleh kamera digital tipe *low-end* mulai dari 80 x 60, 160 x 120, 320 x 240, hingga 640 x 480 *pixel*. Besarnya sampling menentukan tingkat resolusi dari suatu gambar [13].

Dari gambar 2.11 dapat dilihat bahwa semakin besar sampling maka semakin jelas informasi dari suatu citra. Namun semakin besar sampling maka memori yang diperlukan sebagai media penyimpanan juga bertambah besar. Oleh karena itu perlunya menentukan besarnya sampling sekecil mungkin, tetapi informasi tidak hilang.



Gambar 2.11. Resolusi gambar (a) 176 x 127 (b) 88 x 63 (c) 44 x 31 (d) 22 x 15

b. Digititasi Amplitudo

Pada digititasi amplitudo setiap piksel harus diberikan sebuah kode angka yang merepresentasikan intensitas cahaya pada titik tersebut. Tingkat intensitas sebuah citra ditunjukkan oleh sebuah tingkat keabuan atau *grey level* dan keseluruhan rentang tingkatan intensitas yang ada pada citra digital disebut sebagai skala keabuan atau *grey scale.* Sebuah piksel akan mempunyai tingkat keabuan g, dimana:

$$0 \leq g \geq 2^{l} - 1 \quad \text{..... (33)}$$

dengan l adalah bilangan bulat dan terdapat 2^{l} tingkat keabuan pada sebuah skala keabuan [12].

c. Pemilihan Parameter Digitisasi

Metode untuk merepresentasikan citra digital harus dipilih sehingga memenuhi kedua persyaratan berikut:

- metode tersebut harus memudahkan serta memperbesar effisiensi proses yang dilakukan oleh komputer.
- metode tersebut harus dapat mencakup semua informasi mendefinisikan karakteristik citra yang relevan.

Untuk memenuhi kreteria (b) maka nilai m, n dan 1 harus sebesar mungkin agar dapat mencakup semua informasi yang terdapat dalam citra, akan tetapi hal ini bertolak belakang dengan kriteria yang menitikberatkan pada effisiensi perhitungan. Hal ini dikarenakan setiap citra digital akan mempunyai dimensi 1 x m x n bit, sehingga effisiensi perhitungan akan dapat ditingkatkan dengan menjaga nilai 1, m serta 1 adalah tetap rendah.

d. Citra Biner

Suatu kasus khusus pada digitasi amplitude bilamana terjadi skala keabuan yang hanya terdiri dari dua tingkat. Harga mungkin yaitu $l = 1$, sehingga citra yang dihasilkan terdiri dari titik array yang setiap titik tersebut dapat seluruhnya hitam atau seluruhnya putih. Citra digital yang terdiri dari array dengan dua macam harga demikian disebut dengan citra biner.

Menghasilkan sebuah citra biner dari suatu citra yang memiliki skala keabuan lebih umum dan sangat mudah dilakukan dalam praktek. Hanya diperlukan suatu nilai ambang batas *threshold* yang tepat untuk merubah citra menjadi piksel-piksel dengan hanya salah satu dari dua harga. Jadi jika,

f(x,y) = harga intensitas pada koordinat (x,y) pada citra

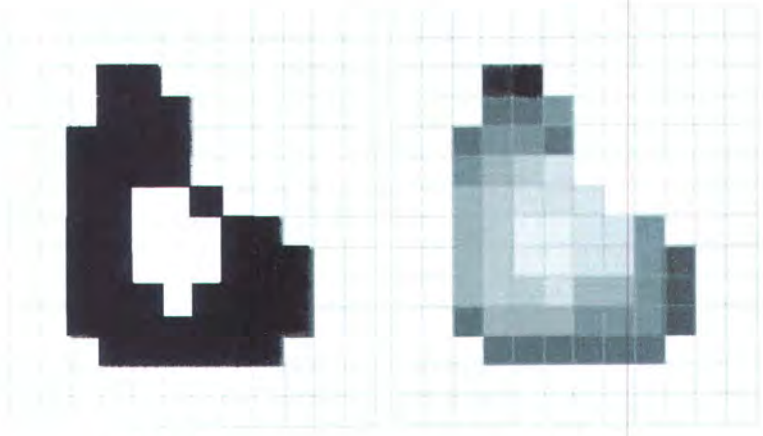
T = nilai ambang

maka, sebagaimana diperilihatkan pada gambar. 2.12, citra yang dinyatakan oleh f(x,y) dapat diubah menjadi biner dengan menerapkan aturan bahwa:

Jika $f(x,y) \geq T$, maka $f'(x,y) = 1$

Jika $f(x,y) < T$, maka $f'(x,y) = 0$

Dimana f '(x,y) menyatakan versi biner dari f (x,y).



black white *gray scale*

Gambar 2.12. Binerisasi citra grayscale

2.5.2. Pemrosesan Citra Digital

2.5.2.1. *Thresholding*

Citra biner umumnya lebih mudah untuk dianalisa dari citra grayscale. Oleh karena itu seringkali diperlukan *threshol* untuk mengubah citra grayscale menjadi citra biner, sehingga dapat dengan mudah dipisahkan antara daerah *background* dan *foreground*. *Thresholding* mengubah nilai pixel menjadi bernilai 1 jika lebih besar daripada nilai *threshold* tertentu dan bemilai 0 jika lebih kecil daripada nilai *threshold* [12].

$$g(i,j) = \begin{cases} 1; f(i,j) \geq T \\ 0; f(i,j) < T \end{cases} \qquad (34)$$

Pemilihan nilai *threshold* T adalah sangat penting. Biasanya digunakan histogram untuk menentukan pemilihan nilai *threshold*. Sebuah citra *grayscale* biasanya terdiri dari dua buah puncak atau lebih yang dipisahkan oleh lembah yang jelas.

2.5.2.2. Greyscaling

Grayscaling adal teknik yang digunakan untuk mengubah citra berwarna menjadi bentuk *gray scale*. Pada dasarnya semua pemrosesan citra digital dalam bentuk *balck and white* atau citra biner, sehingga citra berwarna harus diubah dahulu ke bentuk gray scale, kemudian dilakukan proses *thresholding* untuk mendapatkan citra biner. Pengubahan dari citra berwarna ke bentuk *gray scale* mengikuti aturan :

$$I(i,j) = \frac{R(i,j) + G(i,j) + B(i,j)}{3} \quad \text{.....(35)}$$

2.5.2.3. Filter

Filter adalah teknik untuk modifikasi atau memperbaiki kualitas dari citra. Jika kita berbicara mengenai frekuensi audio maka kita mengacu pada pembahan sinyal audio yang terjadi untuk setiap satuan waktu. Akan tetapi jika kita berbicara mengenai frekuensi pada citra digital maka pada perubahan warna yang terjadi pada sebuah citra digital. Frekuansi spasial diukur dari seberapa cepat kecerahan atau warna berubah pada sebuah citra digital. Citra digital yang mempunyai kecerahan atau warna yang berubah secara perlahan-lahan dikatakan bahwa citra tersebut mempunyai frekuensi spasial yang rendah. Sedangkan citra dengan perubahan kecerahan atau warna yang mendadak, tekstur yang kuat atau detail yang tinggi dikatakan bahwa citra tersebut mempunyai frekuensi spasial yang tinggi.

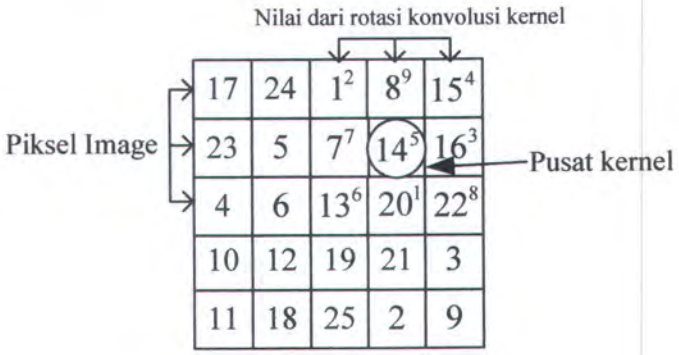
Filtering merupakan operasi yang melibatkan piksel tetangga atau disebut dengan *neighborhood.* Nilai piksel pada citra outoput ditentukan oleh nilai piksel tetangga pada citra input. *Linear filtering* adalah filter yang menggunakan operasi linear antara piksel tetangga. Operasi linear mencakup operasi korelasi dan konvolusi yang merupakan konsep dasar. Hampir semua pemrosesan citra digital menggunakan operasi korelasi dan konvolusi, seperti *masking, averaging, dilation, erosiaon, edge detectio,*dan *template matching.*

2.5.2.3.1. Konvolusi

Konvolusi adalah operasi dasar dalam proses filter. Pada operasi konvolusi, nilai piksel ditentukan oleh jumlah nilai dari piksel tetangga dengan bobot tertentu. Matrik bobot disebut sebagai kernel atau dikenal juga dengan sebutan filter [14].

$$A = \begin{pmatrix} 17 & 24 & 1 & 8 & 15 \\ 23 & 5 & 7 & 14 & 16 \\ 4 & 6 & 13 & 20 & 22 \\ 10 & 12 & 19 & 21 & 3 \\ 11 & 18 & 25 & 2 & 9 \end{pmatrix} \qquad h = \begin{pmatrix} 8 & 1 & 6 \\ 3 & 5 & 7 \\ 4 & 5 & 2 \end{pmatrix}$$

Sebagai contoh matrik A adalah dari sebuah citra dan matrik h sebagai kernel. Jika akan dihitung oiksel output pada baris 2 kolom 4, maka dengan cara menghitung : pertama putar matrik kernel sebesar 180^0. Geser matrik kernel sampai pusat kernel berhimpit dengan posisi elemen (2,4). Hasil output merupakan jumlah perkalian elemen kernel dengan matrik A, Seperti terlihat pada gambar 2.13.



Gambar.2.13. Operasi Konvolusi

Maka hasil konvolusi pada elemen (2,4) :

1x2 +8x9 + 15x4 + 7x7 + 14x5 + 16x3 + 13x6 + 20x1 +22x8 = 575

2.5.2.3.2. Korelasi

Operasi korelasi pada prinsipnya hampir sama dengan Operasi konvolusi hanya saja pada operasi korelasi matriks kernel tidak diputar 180°. Seperti pada matrik A dan h sebelumnya akan dihitung pixel ouput pada elemen (2,4) maka hasilnya sebagai berikut:

Gambar 2.14. Operasi Korelasi

2.5.2.3.3. Dilation dan Erosion

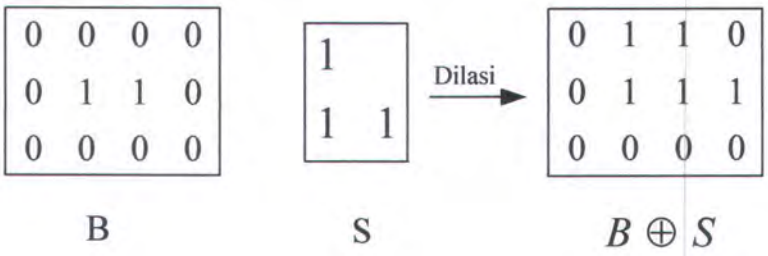
Dua operasi dasar untuk analisa morfologi adalah *dilation* dan *erosion.* Hampir semua operasi morfologi dapat dinyatakan dalam kedua operasi dasar ini.. Matematika morfologi adalah bagian dari pemrosesan citra digital yang berkaitan dengan bentuk dan struktur citra yang bertujuan untuk memperbaiki kualitas citra ditinjau dari bentuk dan struktur.

Dilation adalah operasi perluasan dari kumpulan pixel yang bernilai 1. *Dilation* dapat digunakan untuk memperluas area suatu bentuk- *Dilation* juga dapat digunakan untuk menutup lubang yang ada dalam suatu bentuk.

Dilation dirumuskan sebagai berikut:

$$A \oplus B = \left\{ Z \mid \left(\hat{B}\right)_Z \cap A \neq 0 \right\} \quad \text{..... (36)}$$

Rumus diatas menyatakan bahwa hasil *dilation* adalah himpunan titik dalam B yang dirotasi 180° dan ditranslasikan sebesar z dalam A. Implementasi dalam logika yaitu sebagai berikut, *dilation* (B,S) mengambil citra biner B dan menempatkan *origin* dari elemen S pada pixel yang bernilai 1 serta melakukan operasi OR elemen S dengan citra biner B.

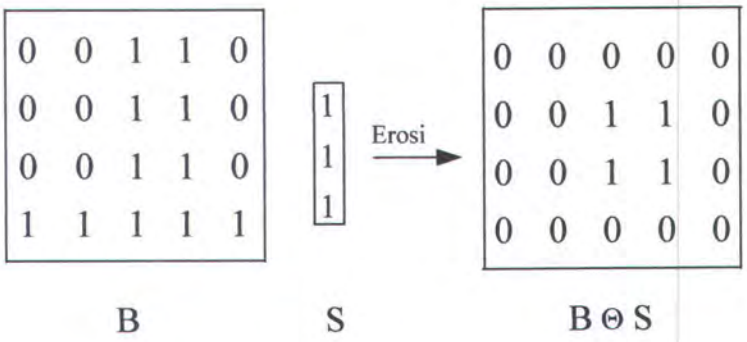


Gambar 2.15. Operasi Dilasi

Erosion adalah operasi penyusutan dari kumpulan pixel yang bernilai 1. *Erosion* dapat digunakan untuk mempersempit area suatu bentuk. *Erosion* juga dapat digunakan untuk menghilangkan tonjolan kecil yang ada dalam suatu bentuk atau menghilangkan percabangan. *Erosion* dirumuskan sebagai berikut:

$$A \ominus B = \left\{ z \mid \left(\hat{B}\right)_z \subseteq A \right\} \quad \text{.....(37)}$$

Rumus diatas menyatakan bahwa hasil *erosion* adalah himpunan titik didalam B dan ditranslasikan sebesar z ada dalam A. Implementasi dalam logika yaitu sebagai berikut, *erosion* (B,S) mengambil citra biner B dan menempatkan *origin* dari elemen S pada setiap piksel serta melakukan operasi OR citra biner B dengan biner 1 jika semua posisi dalam S tertutup oleh pixel bernilai 1 pada B.



Gambar 2.16. Operasi *Erotion*

2.5.2.4. Color Matching

Color matching digunakan untuk segmentasi warna. Piksel berwama sama dengan warna referensi dalam range error tertentu akan diubah menjadi piksel biner bernilai 1. Teknik ini biasanya digunakan untuk pengenalan obyek dengan cepat. Hal yang pertama dalam proses *color matching* adalah dengan menormalisasikan warna obyek dan warna referensi setiap komponen RGB.

Harga konstanta k mewakili besar toleransi error. Semakin besar nilai k maka toleransi terhadap error akan semakin besar juga artinya ketelitian

color matching akan semakin rendah. Setelah melakukan perhitungan nilai korelasi, langkah selanjutnya adalah melakukan proses *thresholding* nilai korelasi pada tiap-tiap channel. Perhitungan tersebut mengikuti aturan:

$$r = \frac{R}{(R + G + B)}$$
$$g = \frac{g}{(R + G + B)} \quad \text{..... (38)}$$
$$b = \frac{B}{(R + G + B)}$$

Setelah melakukan normalisai warna obyek dan warna referensi, selanjutnya adalah mengitung nilai korelasi piksel terhadap nilai rata – rata warna referensi. Persamaan (39) dapat digunakan untuk menentukan nilai korelasi pada R :

$$r = \exp\left(\frac{-\left(r_{obj} - r_{mean}\right)^2}{k}\right) \quad \text{..... (39)}$$

Nilai pixel akan bernilai 1 jika ketika nilai korelasi rgb lebih besar atau sama dengan nilai *threshold.*

2.6. *Space* Warna

Monitor komputer CRT secara khas mempunyai tiga buah elektron *gun* yang berfungsi untuk memancarkan tiga macam fospor yang berbeda pada layar. Fospor ini memancarkan cahaya (warna) dasar utama, yaitu merah (R), hijau (G) dan biru (B). Dari warna dasar ini warna yang dihasilkan oleh layar monitor. Sebuah kombinasi nyata secara pisik menentukan karaktristik sistem *vision*

manusia yang merasakan warna tersbut. Suatu ruang warna selalu menampilkan tiga dimensi. Ada beberapa ruang warna yang didefenisikan [15] :

- Digital *Image* sering menggunakan *red/green/blue* sebagai ruang warnanya, yang secara sederhana disebut RGB.
- Ruang warna yang digunakan untuk mencetak digunakan *Cyan/Yellow/Magenta*, yang dikenal dengan CYM
- *Hue, Saturation* dan *intensity* (HIS), biasanya ruang warna ini digunakan oleh artis.

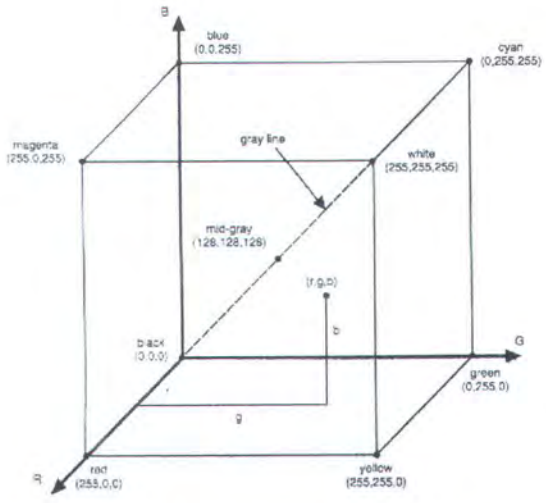
Meskipun demikian untuk proses digital *image* menggunakan RGB, yang mana hasilnya nanti yang ditampilkan ke layar monitor. Banyak aplikasi pengolahan citra digital memerlukan perubahan bentuk ke ruang warna yang lainnya. Pada umumnya program komputer akan melakukan perubahan bentuk ruang warna ini.

2.6.1. *Red Green Blue* (RGB)

Semua ruang warna terdapat dalam sistem koordinat ortogonal tiga dimensi, artinya ada tiga buah sudut (dalam hal ini adalah intensitas warna merah, hijau dan biru) yang tegak lurus terhadap yang lainnya. Seperti yang ditunjukkan gambar 2.17. Intensitas merah mulai dari nol dan meningkat sepanjang sisinya, dan dengan cara yang sama juga untuk warna hijau dan biru. Masing – masing warna hanya dapat mempunyai nilai-nilai antara nol dampai nilai intensitas maksimumnya (255 untuk 8-bit), sehingga struktur ini

akan menghasilkan kubus. Koordinat ini pada umumnya menampilkan tiga buah (merah, hijau dan biru) yang terdapat didalam tanda kurung.

Semua warna bisa dipetakan kedalam kubus RGB atau ruang warna. Warna hitam mempunyai intensitas nol untuk masing – masing warna merah, hijau dan biru. Maka warna hitam mempunyai titik koordinat (0,0,0). Dan sebaliknya warna putih mempunyai intensitas warna maksimum untuk merah, hijau dan biru, sehingga titik koordintanya (255,255,255). Untuk warna merah maksimum mempunyai komponen warna hijau dan biru yang nol, maka titik koordinatnya adalah (255,0,0). Kuning mempunyai titik koordinat (255,255,0). Dan untuk semua warna yang lainnya dapat ditempatkan didalam koordinat kubus ini. Ruang warna RGB asalnya dimulai dengan warna hitam, dan semua warna lainnya diperoleh dengan manambahkan sejumlah warna dasar tersebut.



Gambar 2.17. Koordinat Kubus warna RGB

2.6.2. *Cyan Yellow Magenta* (CYM)

Ruang warna CYM jika dihubungkan dengan ruang warna RGB adalah kebalikannya. Yang untuk ruang warna CYM dimulai dari putih, dan sudut asal dari sistem koordinatnya tidaklah merah, hijau dan biru tetapi cyan, kuning dan meganta. Ruang merah ini diperoleh dari kombinasi dari kuning dan magenta, sedangkan hijau dari kuning dan cyan dan biru dari cyan dan magenta. Persamaan berikut ini menggambarkan bagaimana cara untuk mengkonversikan ruang RGB ke ruang CYM atau sebaliknya [15].

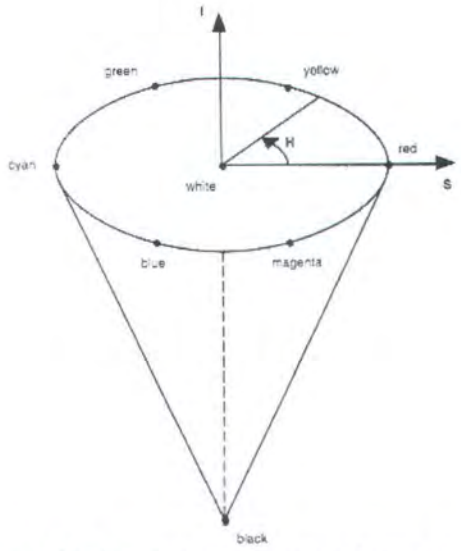
c= max-r m= max-g y= max-b

r= max-c g= max-m b= max-y ..... (40)

dimana max adalah nilai intensitas maksimum (untuk 8-bit nilai maksimum adalah 255).

2.6.3. *Hue Saturation Intensity* (HSI)

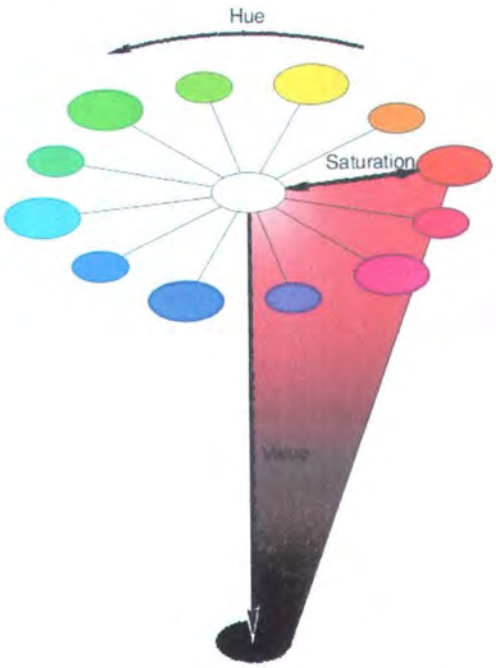
Sistem koordinat ortogonal tiga dimensi HSI berbeda dengan RGB atau CYM. Gambar 2.18 menampilkan bentuk dari koordinat dari HSI. Bentuk kerucut yang mempunyai satu poros pusat yang mewakili intensitas warna. Sepanjang poros menunjukkan nilai *gray*, dengan warna hitam diakhir kerucut dan warna putih pada dasar kerucut. Semakin jauh dari garis tegah, maka intensitas warna semakin tinggi.



Gambar 2.18. Koordinat Kerucut sistem warna HSI

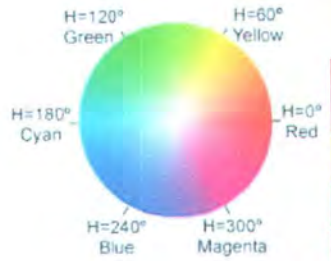
Jika kerucut ini dipandang dari atas, akan terlihat seperti lingkaran. Warna yang berbeda, atau *hue*, yang terdapat pada lingkaran ini. *Hue* ditentukan oleh penempatan sudut yang terdapat pada lingkaran ini. *Saturation,* atau kesempurnaan warna digambarkan sebagai jarak yang tegak lurus terhadap poros intensitas. Warna yang dekat dengan pusat poros mempunyai *saturation* yang rendah. Sedangkan dekat dengan permukaan kerucut mempunyai tingkat saturation yang tinggi.

Sistem *Hue, Saturation dan Value (HSV)* adalah suatu variasi dari sistem HSI Model warna HSV. Model warna HSV, dengan menggunakan nilai *hue* dari 0 derajat sampai 360 derajat, dengan warna merah pada 0 derajat. *Saturation* berkisar pada nilai 0 dan 1, dengan nilai 0 menjadi tidak ada warna (sepanjang pusat poros) dan 1 pada tepi atas luar kerucut. Nilai (variasi dari intensitas) juga mempunyai nilai 0 dan 1, dimana 0 adalah hitam dan 1 adalah putih. Seperti yang terlihat pada gambar 2.19.



(a)

RGB Color space equivalence to HSV Channels



Color	HSV Channels values			RGB Channels values		
	Hue Degree	Saturation %	Value %	Red	Green	Blue
Red	0° = 360°	100	100	255	0	0
Yellow	60°	100	100	255	255	0
Green	120°	100	100	0	255	0
Cyan	180°	100	100	0	255	255
Blue	240°	100	100	0	0	255
Magenta	300°	100	100	255	0	255

(b)

Gambar 2.19. Sistem warna HSV (a) Sistem Koordinat HSV (b) Konversi RGB ke HSV [16]

2.6.4. Transformasi RGB ke HSV

Jika sebuah gambar yang digambarkan dengan (R,G,B), dimana R,G, dan B bernilai antara 0,0 dan 1,0, dengan 0,0 adalah nilai yang minimum dan 1,0 adalah nilai maksimum dari warna tersebut, suatu padanan warna (H,S,V) dapat ditentukan dengan rumusan ini. Nilai maksimum pada sistem RGB sama dengan nilai maksimum HSV, maka didapat rumus dibawah [19]:

$$H = \begin{cases} \left(0 + \dfrac{G-B}{Max-Min}\right) x\,60, & jika\ R = \max \\ \left(2 + \dfrac{B-R}{Max-Min}\right) x\,60, & jika\ G = \max \\ \left(4 + \dfrac{R-G}{Max-Min}\right) x\,60, & jika\ B = \max \end{cases} \quad \cdots\cdots\cdots\cdots\cdots\cdots \quad (41)$$

$$S = \frac{Max - Min}{Max}$$

$$V = Max$$

Hasil nilai yang dibentuk oleh sistem warna HSV, dimana H bervariasi dari nilai 0,0 sampai 360,0, nilai ini menunjukkan nilai sudut di dalam lingkaran dimana *Hue* berada. S dan V bernilai dari 0,0 sampai 1,0, dengan 0,0 adalah nilai minimum dan 1,0 nilai maksimum dari nilai *Saturation* atau *Value*. Sebagai sebuah koordinat, H dapat diputar balik dari 360 ke 0, maka semua nilai dari H berkisar dari 0,0 sampai 360,0 sehingga H dapat dipetakan dengan nilai 360.

Rumusan dibawah ini memberikan beberapa sifat dari sistem warna HSV:

- ✓ Jika MAX = MIN (S = 0), kemudian H adalah tak tergambarkan. Ini jelas kelihatannya, dengan melihat diagram dari ruang HSV, jika S = 0 warna berada sepanjang garis pusat dari gray, yang secara alami tidak mempunyai *Hue*, dan koordinat sudut tidak digunakan.
- ✓ Jika MAX= 0 (jika V= 0), kemudian S adalah tak tergambarkan. Ini merupakan gambaran diagram terbaik yang terbaik dari diagram kerucut diatas. Jika V = 0, kemudian warnya adalah hitam murni, sehingga warna ini tidak mempunyai *hue dan saturation.*

2.6.5. Transformasi HSV ke RGB

Sebuah warna yang digambar menggunakan sistem HSV, dengan nilai H berkisar antara 0,0 sampai 360,0 yang menandakan sudut dalam derajat dalam lingkaran warna dimana *hue* berada. Dan dengan nilai S dan V antara 0,0 sampai 1,0. suatu warna yang berseuaian dengan sistem warna RGB dapat ditentukan dengan rumusan berikut ini [19] :

Pertama, jika S sama dengan 0,0, maka akan menghasilkan warna *achromatis* atau grey. Pada kasus khusus ini, R,G dan B hanya sama dengan V. Seperti kasus diatas, maka H tidak relevan pada situasi ini.

Jika S tidak sama dengan nol, rumusan berikut ini dapat digunakan :

$$H_i = \left\lfloor \frac{H}{60} \right\rfloor$$

$$f = \frac{H}{60} - H_i$$

$$p = V \; x \; (1 - S) \qquad \text{..... (42)}$$

$$q = V \; x \; (1 - (S \, x \, f))$$

$$t = V \; x \; (1 - (S \, x \, (1 - f)))$$

Jika Hi = 0,R = v,G = t,B = p

Jika Hi = 1,R = q,G = v,B = p

Jika Hi = 2,R = p,G = v,B = t

Jika Hi = 3,R = p,G = q,B = v

Jika Hi = 4,R = t,G = p,B = v

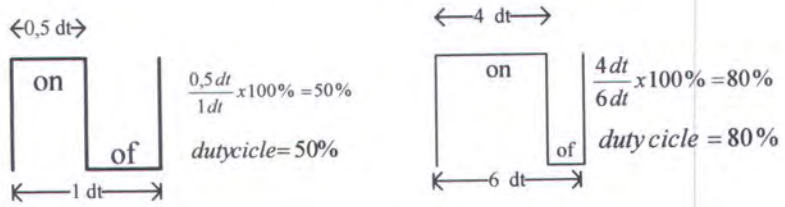
Jika Hi = 5,R = v,G = p,B = q

2.7. Pulse Width Modulator (PWM)

Salah satu bagian yang perlu diperhatikan untuk mengatur kecepatan dari motor DC adalah *Pulse Width Modulation* (PWM). PWM merupakan sinyal kontrol untuk menghidup-matikan saklar. PWM menghasilkan sinyal berbentuk kotak dengan frekeunsi yang tetap, tetapi *duty cycle* dapat berubah. *Duty cycle* merupakan periode waktu hidup (*high* atau hidup) suatu pulsa dalam persen dengan rumus :

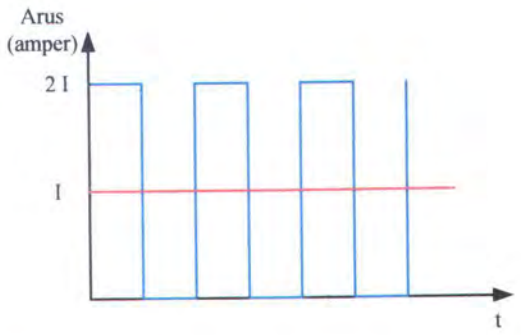
$$D = \frac{waktu \;\; On}{waktu \; perioda \; on \,/\, off} \; x \; 100\% \qquad \text{..... (43)}$$

Gambar dibawah ini (gambar 2.20) menjelaskan persamaan (43), yang menunjukkan dua buah sinyal PWM dengan *duty sycle* yang berbeda.



Gambar 2.20. Sinyal PWM

Proses *switching* pada MOSFET menghasilkan rugi daya yang kecil. Karena itu semakin besar waktu yang diperlukan untuk *switching* semakin besar daya yang hilang. Semakin besar frekuensi PWM semakin stabil sinyal arus yang melewati motor. Pada frekuensi rendah sinyal arus ini merupakan sinyal *spiky*, tapi pada frekuensi tinggi induktansi dari motor akan menghaluskannya menjadi nilai rata-rata arus DC. Hal tersebut dapat dijelaskan melalui gambar dibawah :



Gambar. 2.21. Sinyal arus yang melewati Daya

Untuk sinyal yang berwarna biru merupakan hasil proses *switching* dengan frekuensi yang rendah sedang yang berwarna merah atau disebut dengan

istilah *switching case* menunjukkan proses sebaliknya sehingga menghasilkan sinyal DC *steady* atau disebut dengan istilah DC *case*. Keduanya mempunyai nilai rata-rata arus yang sama, tapi jika kita menghitung disipasi daya yang terjadi karena tahanan pada motor, maka:

- ✓ Untuk Motor DC *case* :

$$P = I^2 R \quad \text{.......................................} \quad (44)$$

- ✓ Untuk *switching case* :

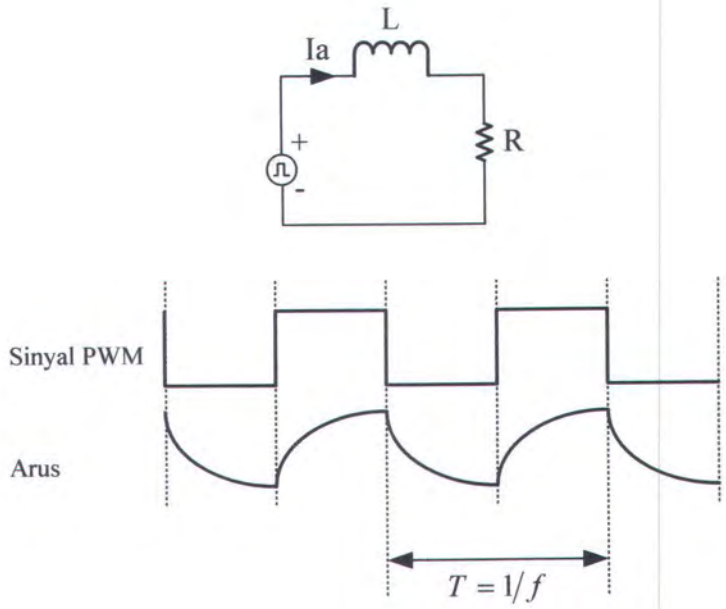
$$P = \frac{(2I)^2 R}{2} + \frac{0^2 R}{2} \quad \text{.......................................} (45)$$
$$= 2I^2 R$$

Jadi pada *switching,* disipasi daya yang hilang dua kali lebih besar dari DC *case*. Pada kenyataannya terlihat pada gambar 2.21, tapi tetap *switching* rugi dayanya lebih besar daripada DC.

Salah satu cara menentukan frekuensi sinyal PWM yang benar adalah dengan mempertimbangkan besarnya kestabilan dari arus sebesar p dalam persen. Gambar dibawah ini (gambar 2.22) menunjukkan sebuah rangkaian ekivalen dari motor DC, sinyal PWM 50 % dan sinyal arus pada pada keadaan *switching.*

T adalah periode *switching* dimana berbanding terbalik dengan frekuensi. Persamaan sinyal arus di atas adalah sebagai berikut:

$$i = I e^{-t/\tau} = I e^{-tR/L} \quad \text{..... (46)}$$



Gambar. 2.22. Sinyal PWM dan arus jangkar [19]

dimana τ adalah *time konstan* dari rangkaian, sebesar *L/R*. Arus pada saat t = T/2 tidak boleh kurang sebesar p% dibandingkan saat t = 0 (io). Ini berarti ada kondisi pembatasan arus pada rangkaian.

$$i_1 = \left(1 - \frac{p}{100}\right) i_0 \quad \text{..... (46)}$$

sehingga :

$$I e^{-\frac{TR}{2l}} = \left(1 - \frac{p}{100}\right) I e^{0}$$

$$e^{-\frac{TR}{2L}} = 1 - \frac{p}{100}$$

$$-\frac{TR}{2L} = \ln\left(1 - \frac{p}{100}\right)$$

$$T = \frac{2L}{R}\ln\left(1 - \frac{p}{100}\right)$$

Bila $f = 1/T$, maka

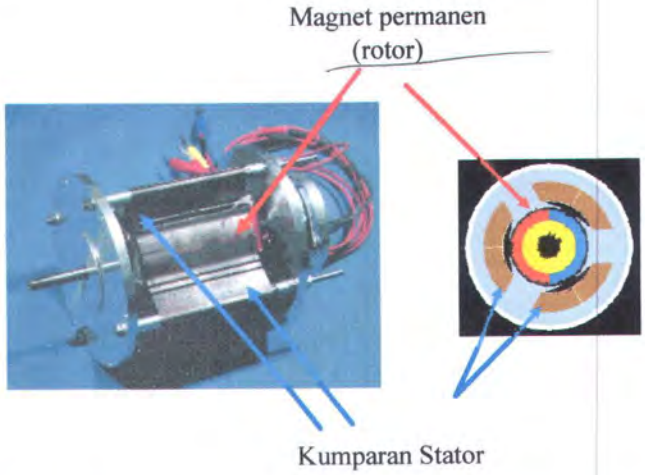
$$f = \frac{R}{-2L\ln\left(1 - \frac{p}{100}\right)} \qquad \text{.. (47)}$$

Persamaan (47) digunakan untuk memilih besarnya frekuensi dari PWM yang digunakan untuk pengaturan kecepatan motor DC.

2.8. Motor DC

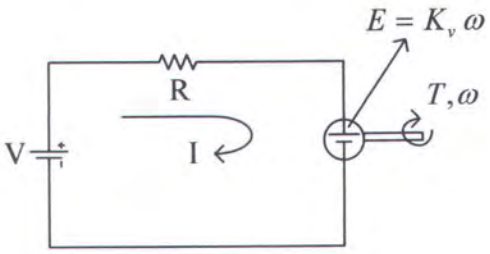
Motor merupakan mesin listrik yang dapat merubah energi listrik menjadi energi mekanik dalam bentuk putaran. Motor DC magnet permanent merupakan salah satu jenis motor yang sumber energi listriknya didapatkan dari sumber arus searah (DC). Struktur mekanisnya terdiri dari dua bagian, yaitu : stator, merupakan bagian yang diam yang dibentuk dari magnet permanen dan rotor merupakan bagian yang bergerak yang dibentuk dari lilitan kawat yang dialiri arus dari sumber tegangan luar.

Arus yang mengalir pada kumparan jangkar (rotor) yang berada diantara medan magnet stator akan menimbulkan gaya sebesar F, gaya inilah yang menimbulkan Torsi pada kumparan jangkar, sehingga menyebabkan kumparan jangkar berputar.



Gambar. 2.23. Struktur mekanis motor DC magnet permanen

Motor DC permanen magnet dapat di dimodelkan sebuah rangkaian elektronik sederhana sebagai berikut :



Gambar 2.24. Model Listrik Motor DC

V merupakan sumber tegangan DC dan R merupakan resistansi kumparan jangkar, dalam hal ini komponen induktansi dari kumparan jangkar diabaikan, karena dalam kondisi *steady state* induktansi bernilai nol dalam sebuah

rangkaian DC. Harga resistansi kumparan jangkar tidak dapat diukur ketika motor dalam keadaan diam, karena nilainya berubah terhadap kecepatannya. E merupakan tegangan induksi pada kumparan jangkar motor.

Hubungan diatas dapat dinyatakan dalam sebuah persamaan tegangan kirchof:

$$V = I\ R + E \quad \text{...} \quad (48)$$

Pengaruh medan magnet yang dihasilkan oleh magnet permanent stator dengan besar fluknya konstan, menghasilkan dua persamaan yang berhubungan dengan kecepatan dan torsi motor:

$$E = K_v\ \omega \quad \text{...} \quad (49)$$

$$T = K_m\ I \quad \text{...} \quad (50)$$

dimana Kv sebagai konstanta kecepatan motor dan Km sebagai konstanta mekanis motor. Persamaan ini lazim disebut sebagai persamaan tranduser motor. Besaran Km dan Kv berbeda untuk tiap jenis motor.

Hubungan antara kecepatan dengan torsi dirumuskan sebagai:

$$\omega = \frac{V}{K} - \frac{R\,T}{K^2} \quad \text{...} \quad (51)$$

Nilai konstanta K dapat dengan dicari dengan memberikan input tegangan V pada motor dan mengukur kecepatan motor tanpa beban ketika torsi yang bekerja sama dengan nol.

$$K = \frac{V}{\omega_0} \quad \text{...} \quad (52)$$

Sehingga nilai torsi yang bekerja pada motor jika dibebani dapat diketahui dengan :

$$T = K I = \frac{V I}{\omega_0} \quad \text{..... (53)}$$

dalam hal ini torsi merupakan torsi total yang dihasilkan termasuk torsi yang bekerja untuk melawan gaya gesek pada poros motor.

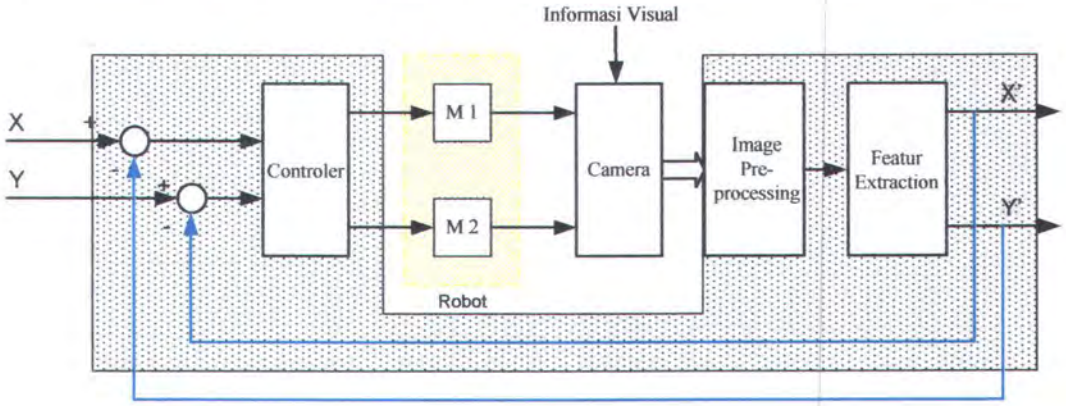
BAB III
PERENCANAAN DAN PEMBUATAN ALAT

BAB III

PERENCANAAN DAN PEMBUATAN ALAT

3.1 Perencanaan Sistem

Sistem *path tracking* dan *obstacle avoidance* ini dirancang dalam blok diagram seperti gambar dibawah ini :



Gambar 3.1. Diagram Blok Sistem

Sistem ini menggunakan prinsip *image based visual servoing* (IBVS). Sehingga sistem ini menggunakan *feature point* dari citra digital secara langsung sebagai *feedback.*

Dari diagram blok diatas dapat dijelaskan sebagai berikut. Kamera digital yang ditempat didepat *mobile robot* difungsi sebagai *end effector* akan menangkap gambar obyek. Kemudian gambar tersebut dikirimkan ke CPU (komputer) untuk diproses lebih lanjut. Proses pertama kali dilakukan oleh komputer adalah proses *image pre-processing.* Proses ini dilakukan sebagai proses awal yang bertujuan untuk pengkondisian sinyak gambar. Proses ini terdiri dari proses transfromasi ke sistem warna SHV, dimana pada sistem ini akan dilakukan pemisahan gambar

obyek dengan gambar lainnya. Karena dilakukan pemilihan sistem SHV ini, disebabkan dinding yang digunakan sebagai pembatas berwarna putih dan lantai berwarna hitam. Maka hanya dengan mengambil nilai *saturation* saja akan terjadi pemisahan antara gambar obyek dengan dinding dan lantai. Sehingga akan diketahui posisi dari obyek. X dan Y adalah titik koordinat dari robot, sedangkan X' dan Y' merupakan nilai koordinat baru, dan nilai ini dibandingkan dengan nilai X dan Y. Diperoleh eror dari posisi, sinyal eror ini yang diumpankan ke driver motor.

Diagram alir dari software sistem dapat dilihat pada gambar 3.2 berikut ini;

Gambar 3.2. Diagram Alir Sistem

Diagram alir dari sistem pada gambar 3.2, dapat dijelaskan sebagai berikut :

- Kamera digital akan menangkap citra analog obyek dan kemudian mengubahnya menjadi sinyal digital dan ditransmisikan ke CPU.
- CPU memproses sinyal citra digital tersebut untuk mendeteksi adanya target dan penghalang pada citra digital tersebut. Jika tidak ada target dan penghalang maka kamera akan mengambilnya kembali.
- Apabila CPU menemukan target dan penghalang, maka proses selanjutnya adalah pembentukan lintasan.
- Pembentukan lintasan ini tergantung pada algoritma *path tracking* yang digunakan. Pada penelitian ini penulis menggunakan algoritma *follow-the-carrot* dan algoritma *pure pursuit.*
- Kemudian *mobile robot* berjalan mengikuti lintasan yang telah ditentukan tersebut. Didalam mengikuti lintasan ini, sinyal PWM digunakan untuk mengatur kecepatan dari *mobile robot* untuk melakukan manuver. Jika target telah ditemukan, maka sistem berhenti.

3.2 Modul Kamera

Kamera sebagai sensor dalam sistem *path tracking* dan *obstacle avoidance* ini memegang peranan penting dalam menentukan keberhasilan dari pada sistem. Kamera digital yang berfungsi sebagai perangkat akuisisi citra digital harus dapat menghasilkan data digital yang akurat dan sesuai dengan keadaan objek sesungguhnya.

Pada sistem *path tracking* dan *obstacle avoidance* ini dipilih kamera digital jenis WebCam Messenger buatan Logitech. Fitur yang dimiliki oleh kamera ini antara lain [20] :

- High Quality VGA CCD sensor yang menghasilkan gambar lebih tajam.
- Resolusi gambar hingga 640 x 480 piksel
- *Still image capture* sampai 1280 x 960 piksel, 1,3 megapiksel.
- *Frame rate* : 30 *frame per second*
- *Full Auto Modes,* yaitu kemampuan untuk menyesuaikan pencahayaan lingkungan.
- *USB interface*

Gambar 3.3. Logitech QuickCam Pro 4000

3.3 *Central Processing Unit* (CPU)

CPU (*Central Processing Unit*) sebagai perangkat yang bertugas untuk mengolah data citra digital. Kecepatan dalam pengolahan data citra digital ini ditentukan oleh spesifikasi CPU. Pada sistem implementasi penelitian ini menggunakan CPU dengan spesifikasi sebagai berikut :

- Processor Intel Pentium IV 3,0 MHz.
- Memori DDR-SDRAM 512 MB

- Video Grapik Adapter 128 MB.

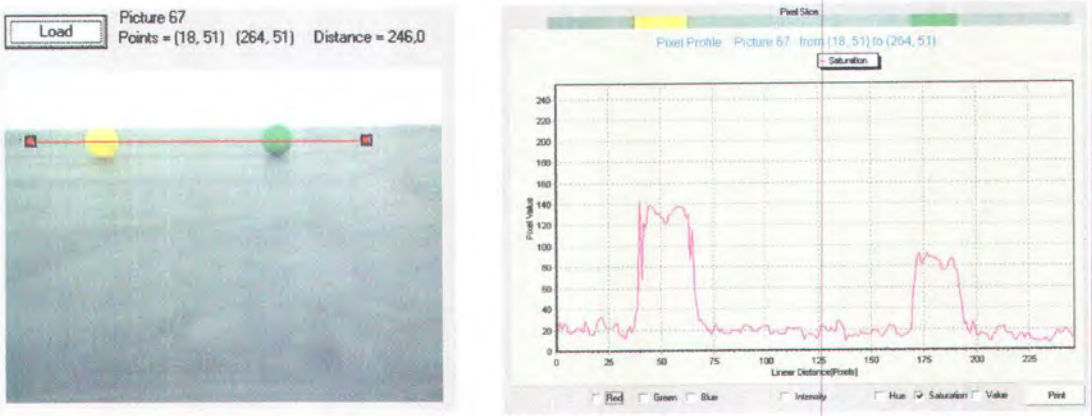
Pemilihan spesifikasi dari CPU ini didasarkan pada pertimbangan bahwa pada sistem yang menggunakan pengolahan citra digital harus memiliki kecepatan yang tinggi dan mempunyai kapasitas memori yang besar.

3.4 Perencanaan Software

Prinsip kerja program sistem *path tracking* dan *obstacle avoidance* digambarkan diagram alir pada gambar 3.2. Berdasarkan dengan diagram alir dapat dijelaskan sebagai berikut, yang pertama *software* ini akan melakukan pencarian posisi target dan posisi penghalang, seterusnya akan menentukan jalur yang akan ditempuh untuk menuju ke target. Pada mode pencarian posisi target dan penghalang, dilakukan dengan sistem warna SHV. Dimana dengan hanya mengambil *saturation* saja maka warna putih dan hitam tidak ditampilkan.

3.4.1 Mode deteksi benda.

Pada mode ini, *software* melakukan pemisahan antara latar (jalan dan dinding). Disini dilakukan dengan menggunakan sistem warna SHV, seperti yang ditunjukkan pada gambar 3.4.



(a) (b)

Gambar 3.4. (a) *Image* (b) nilai piksel dengan menggunakan *saturation.*

Dengan menggunakan *saturation,* dapat diketahui jumlah piksel dari gambar sehingga dengan demikian dapat diketahui lebar dari gambar yang diperoleh. Kemudian dengan melakukan kalibrasi, didapat lebar (diameter) dan posisi semua benda. Berikut ini ditampilkan prosedur untuk mendapatkan diameter, posisi dan jumlah benda.

```
For Y = 2 To Picture1.ScaleHeight - 50
    For X = 2 To Picture1.ScaleWidth - 10
      w = Picture1.Point(X, Y)
      If j > 90 Then Exit Sub
      coba w, X, Y, z
      If z > 6 Then
        For i = 0 To 6
          If (X > x1(i)) And (X < x2(i)) Then
            If (Y > y1(i)) And (Y < y2(i)) Then
              X = x2(i)
              GoTo jarot1
            End If
          End If
        Next i
        wr_benda w, j
        k = k + 1
        If x1(j) > X Then x1(j) = X
        If x2(j) < X Then x2(j) = X
      End If
```

3.4.2 Mode menentukan target

Setelah ditemukan jumlah dan posisi benda, selanjutnya akan ditentukan benda yang digunakan sebagai target. Disini hanya ada satu buah benda sebagai target sedangkan benda yang lainnya disebut dengan penghalang. Untuk menentukan target ini, data yang diperlukan dari benda adalah nilai RGB dari target tersebut. Data RGB yang dibandingkan dengan setiap benda yang ditemukan. Rutin program berikut ini adalah untuk menentukan target.

```
obs = 0
  For j = 0 To Combo1.ListCount - 1
    If j <> tar Then
      i = 0
      If x1(0) = 0 And y1(0) = 0 Then Exit Sub
      hitung_kalibrasi j, hy, hx
      If (tx + 5) > hx And (tx - 5) > hx Then
        ox(obs) = hx
        oy(obs) = hy
        wor(obs) = wr(j)
        wog(obs) = wg(j)
        wob(obs) = wb(j)
        obs = obs + 1
        If obs > 15 Then Exit Sub
      End If
      If Adodc1.Recordset.BOF = True And Adodc1.Recordset.EOF = True Then
        MsgBox " There is no data in the Recordset!", , "Oops"
        Exit Sub
      End If
    End If
```

3.4.3 Mode *path tracking* dan *obstacle avoidance*

Setelah target ditentukan, maka selanjutnya adalah menentukan jalan dari pada robot yang akan menuju target. Jika untuk menuju target tersebut tidak ada penghalang, maka robot akan langsung menuju ke target. Tetapi jika untuk menuju target ada benda yang menghalangi, maka *path tracking* dari robot adalah mendekati penghalang yang berada lebih dekat dari robot, selanjutnya robot ini akan menghindari penghalang ini untuk menuju target. Penghindaran ini menggunakan algoritma *follow-the-carrot* dan *pure pursuit.*

Algoritma penghindaran *follow-the-carrot* mengikuti bagian 2.4.1 seperti yang dijelaskan pada bagian 2.4.2 dan algoritma penghindaran *pure pursuit* mengikuti seperti yang dijelaskan pada bagian 2.4.2

3.5 Perencanaan Hardware

3.5.1 Motor DC

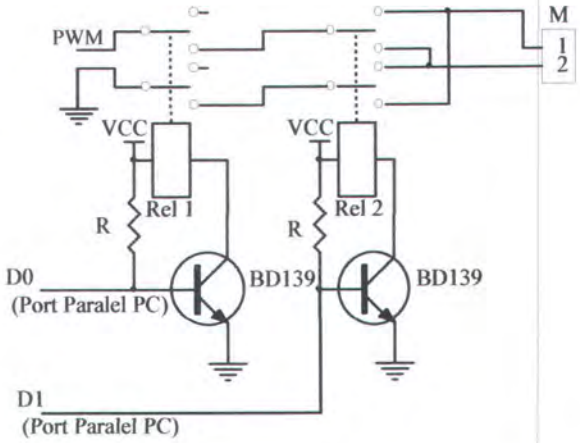
Motor DC yang digunakan pada sistem ini berfungsi sebagai penggerak *end-effector* yang merupakan sebagai penggerak dari pada robot. Robot yang digunakan adalah robot dengan sistem kemudi diferensial. Dengan menggunakan dua buah motor DC, dimana antara motor DC tidak saling berhubungan. Motor DC digunakan sebagai penggerak roda kiri dan kanan. Sehingga dengan mengatur kecepatan dari motor penggerak ini bisa untuk melakukan manuver daripada robot. Gambar 3.5 menampilkan bentuk implementasi dari robot berjalan.

Gambar 3.5. Implementasi Robot.

3.5.2 Driver Motor DC

Driver motor dc digunakan untuk menggerakkan motor DC untuk arah maju dan mundur dari robot, serta untuk berbelok (kiri – kanan). *Driver* motor ini

dirancang dengan menggunakan relay. Dimana disini menggunakan relay jenis DPDT. Relay pertama (Rel 1) berfungsi untuk memutuskan sumber tegangan (ON / Off), sedangkan relay kedua (Rel 2) berfungsi untuk menggerakkan motor DC arah maju atau mundur. Gambar 3.6. manampilkan rangkaian dari driver motor DC.



Gambar. 3.6. Rangkaian Driver Motor DC

Perancangan driver motor DC mengikuti langkah – langkah berikut, pertama ditentukan bahwa ketika input *LOW* maka relay *Open* dan ketika input *HIGH* maka relay *CLODE*. Maka dengan demikian dapat dihitung besarnya tahanan R, dimana basis transistor BD139 harus supaya pada saat input tidak ada tegangan maka arus kolektor harus mampu untuk menghidupkan relay. Dari hasil percobaan yang dilakukan, bahwa harga nilai arus minimum agar relay *close* sebesar 20 mA. Sehingga dengan demikian dapat dihitung besarnya nilai dari tahanan R :

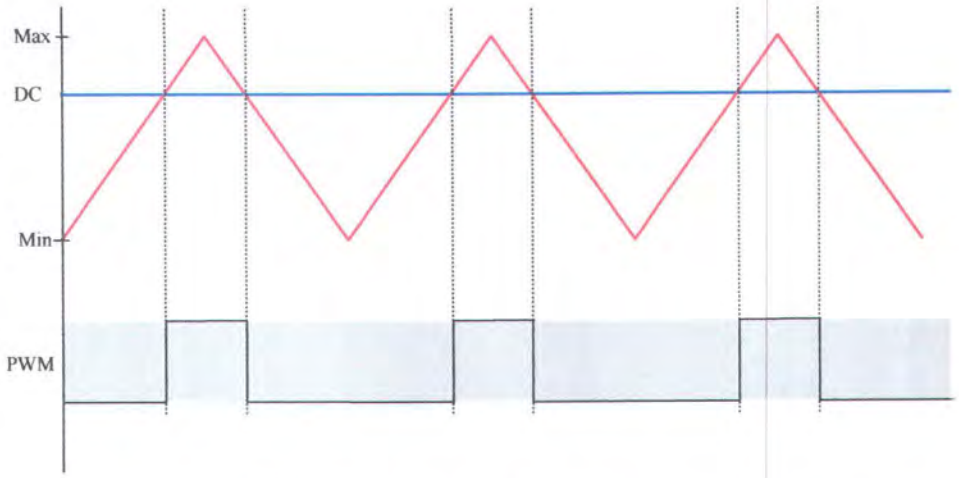
$$Ic = I_{relay} = 20\,mA$$
$$Ib = Ic/\beta dc$$
$$Ib = 20\,mA/40 = 0{,}5\ mA$$

$$R = (Vcc - Vbe) / Ib$$
$$R = (5 - 0{,}6) / 0{,}5\ mA$$
$$R = 8{,}8\,K\Omega \sim 10\ K\Omega$$

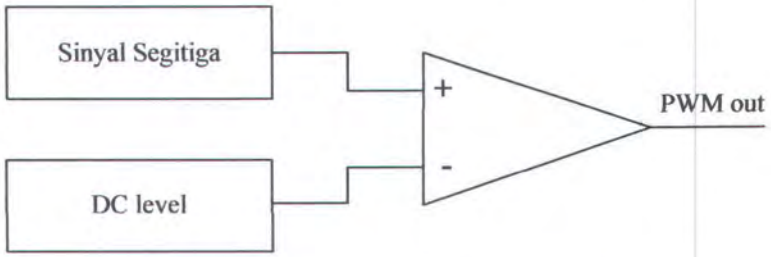
3.5.3 PWM

Sinyal PWM yang dihasilkan merupakan hasil perbandingan antara sinyal segitiga dengan tegangan masukan DC. Nilai masukan tegangan DC berada pada kisaran antara nilai maksimum dan nilai minimum dari sinyal segitiga.

Saat masukan tegangan sinyal segitiga nilainya lebih besar dari masukan DC maka keluaran dari op-amp akan *high*, saat masukan tegangan sinyal segitiga nilainya lebih kecil dari masukan DC maka keluaran dari op-amp akan *low*. Untuk lebih jelasnya proses terjadinya sinyal PWM dapat dilihat pada gambar 3.7. dan gambar 3.8 merupakan diagram blok dari pembangkit sinyal PWM.



Gambar 3.7. Sinyal masukan dan output sinyal PWM



Gambar 3.8. Diagram blok Pembangkit PWM

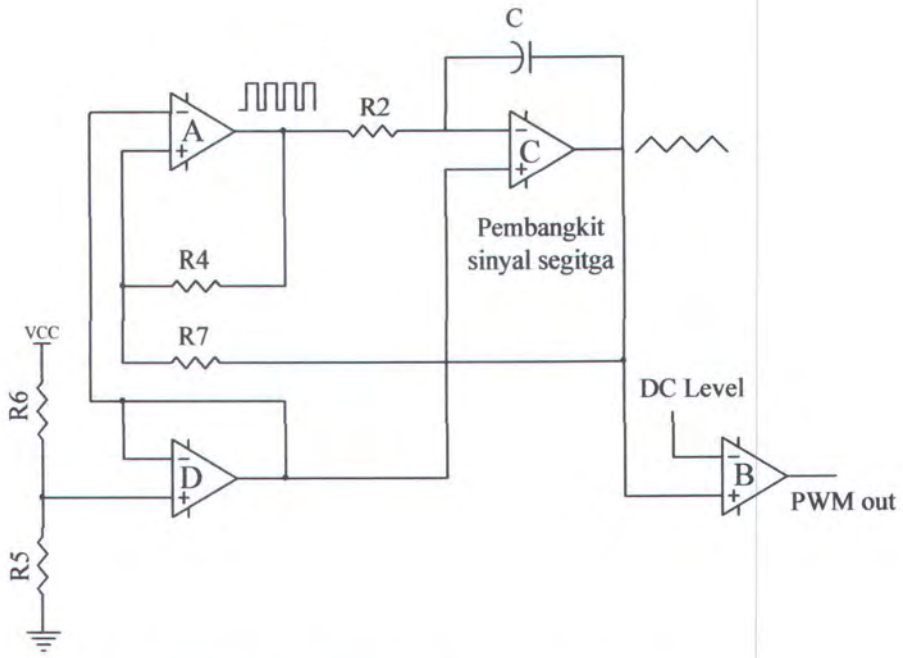
Dari diagram blok tersebut dapat dibuat rangkaian seperti terlihat pada gambar 3.9. Op-amp C berfungsi sebagai bufer yang memberikan tegangan referensi pada op-amp A yang berfungsi sebagai detektor taraf tegangan tak membalik dimana keluarannya berupa sinyal kotak dengan frekuensi sesuai dengan masukan sinyal pada *inverting input.* Sinyal kotak tersebut kemudian menjadi masukan bagi op-amp D yang akan membangkitkan sinyal segitiga.

Sinyal segitiga ini kemudian dimasukkan ke *non-inverting* input dari op-amp B dan kemudian dibandingkan dengan masukan DC (DC level). Keluaran dari op-amp ini merupakan sinyal PWM. Frekuensi dari sinyal PWM dapat dihitung dengan persamaan 3.1.

$$f = \frac{n}{4.Ri.C} \quad \text{.....} \quad (3.1)$$

dimana : Ri = R2 (ohm)

n = R4/R7 (ohm)



Gambar 3.9. Rangkaian Pembangkit Sinyal PWM

Pembangkitan *duty cycle* dari rangkaian pembangkit sinyal PWM tersebut diatur oleh masukan DC pada pembanding (op-amp B). besarnya *high state* sinyal tersebut tergantung dari besarnya nilai Vsat, nilai Vsat sendiri adalag $\pm Vcc$.

BAB IV

PENGUJIAN DAN ANALISA

BAB IV

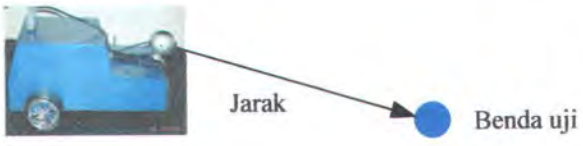
PENGUJIAN ALAT DAN ANALISA

Pengujian alat dilakukan untuk mengetahui apakah sistem yang direncanakan bekerja dengan baik atau tidak. Pengujian alat juga berguna untuk mengetahui tingkat kinerja dari sistem tersebut. Cara yang dilakukan adalah dengan menempatkan benda – benda yang akan diuji.

Sebagai bahan uji disini menggunakan bola dengan diameter 6,5 cm dan terdiri dari lima macam warna : biru, merah, hijau, orange dan kuning. Pengujian – pengujian yang dilakukan adalah :

4.1 Pengujian Jarak Benda.

Pada pengujian ini merupakan pengujian awal. Dimana dengan melakukan pengujian kita bisa menentukan berapa akurat kamera dapat menentukan posisi dan jarak dari masing- masing benda terhapat benda, dan jarak benda terhadap robot. Kita menggunakan kamera yang terdapat pada robot kemudian benda uji ditempatkan didepan dari robot dan benda robot digeser jaraknya. Selanjutnya dilakukan pengambilan data, yaitu dengan pengukuran langsung dan pengukuran yang dilakukan oleh software. Seperti yang terlihat pada gambar 4.1.



Gambar 4.1. Pengujian Jarak Benda

Setalah dilakukan dengan menukarkan benda dengan warna – warna merah, kuning, hijau, orange dan biru. Didapat data – data sebagai berikut :

Tabel 4.1. Pengujian Warna Biru

No	Aktual (cm)	Pengukuran (cm)	Eror	% Eror
1	40	32,750	7,250	18,125
2	50	44,167	5,833	11,666
3	60	53,000	7,000	11,667
4	70	63,530	6,470	9,243
5	80	70,571	9,429	11,786
6	90	87,333	2,667	2,963
7	100	103,611	3,611	3,611
8	110	116,429	6,429	5,845
9	120	119,976	0,024	0,020
10	130	124,750	5,250	4,038
11	140	146,735	6,735	4,811
12	150	147,143	2,857	1,905
13	160	172,778	12,778	7,986
14	170	185,500	15,500	9,118
15	180	197,158	17,158	9,532
16	190	198,444	8,444	4,444
17	200	228,462	28,462	14,231
18	210	248,417	38,417	18,294
Jumlah			184,314	149,285
Rata - Rata			10,240	8,294

Dari data diatas didapat eror rata – rata untuk warna biru adalah : 8,494 %

Tabel 4.2. Pengujian Warna Kuning

No	Aktual (cm)	Pengukuran (cm)	Eror	% Eror
1	40	40,13	0,130	0,325
2	50	50,294	0,294	0,588
3	60	56,02	3,980	6,633
4	70	66,25	3,750	5,357
5	80	81,705	1,705	2,131
6	90	98,429	8,429	9,366
7	100	101,923	1,923	1,923
8	110	125,273	15,273	13,885
9	120	136,296	16,296	13,580
10	130	134,118	4,118	3,168
11	140	152,656	12,656	9,040
12	150	157,589	7,589	5,059
13	160	181,429	21,429	13,393
14	170	179,898	9,898	5,822
15	180	206,667	26,667	14,815
16	190	208,854	18,854	9,923
17	200	217,754	17,754	8,877
18	210	223,667	13,667	6,508
Jumlah			184,412	130,394
Rata - Rata			10,245	7,244

Dari data diatas didapat eror rata – rata untuk warna kuning adalah : 7,244 %.

Tabel 4.3. Pengujian Warna Hijau

No	Aktual (cm)	Pengukuran (cm)	Eror	% Eror
1	40	37,083	2,917	7,293
2	50	47,197	2,803	5,606
3	60	61,429	1,429	2,382
4	70	69,375	0,625	0,893
5	80	86,528	6,528	8,160
6	90	94,394	4,394	4,882
7	100	109	9,000	9,000
8	110	110	0,000	0,000
9	120	132,984	12,984	10,820
10	130	141,253	11,253	8,656
11	140	151	11,000	7,857
12	150	150	0,000	0,000
13	160	164,667	4,667	2,917
14	170	174	4,000	2,353
15	180	191,667	11,667	6,482
16	190	201,5	11,500	6,053
17	200	225,654	25,654	12,827
18	210	236,667	26,667	12,699
Jumlah			147,088	108,878
Rata - Rata			8,172	6,049

Dari data diatas didapat eror rata – rata untuk warna hijau adalah : 6,049 %.

Tabel 4.4. Pengujian Warna Orange

No	Aktual (cm)	Pengukuran (cm)	Eror	% Eror
1	40	37,083	2,917	7,293
2	50	47,197	2,803	5,606
3	60	61,429	1,429	2,382
4	70	69,375	0,625	0,893
5	80	86,528	6,528	8,160
6	90	94,394	4,394	4,882
7	100	109	9,000	9,000
8	110	110	0,000	0,000
9	120	132,984	12,984	10,820
10	130	141,253	11,253	8,656
11	140	151	11,000	7,857
12	150	150	0,000	0,000
13	160	164,667	4,667	2,917
14	170	174	4,000	2,353
15	180	191,667	11,667	6,482
16	190	201,5	11,500	6,053
17	200	225,654	25,654	12,827
18	210	236,667	26,667	12,699
Jumlah			147,088	108,878
Rata - Rata			8,172	6,049

Dari data diatas didapat eror rata – rata untuk warna orange adalah : 6,049 %.

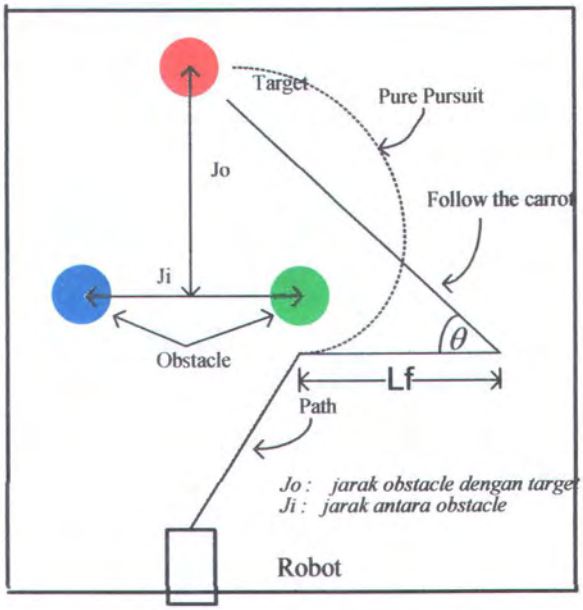
Tabel 4.5. Pengujian Warna Merah

No	Aktual (cm)	Pengukuran (cm)	Eror	% Eror
1	40	37,9	2,100	5,250
2	50	48,615	1,385	2,770
3	60	52,222	7,778	12,963
4	70	70,222	0,222	0,317
5	80	81,094	1,094	1,367
6	90	93,333	3,333	3,703
7	100	105,333	5,333	5,333
8	110	118,788	8,788	7,989
9	120	131,333	11,333	9,444
10	130	136,4	6,400	4,923
11	140	154,219	14,219	10,156
12	150	162,127	12,127	8,085
13	160	174,431	14,431	9,019
14	170	173,333	3,333	1,961
15	180	199	18,981	10,545
16	190	198,651	8,651	4,553
17	200	219,765	19,765	9,882
18	210	236,647	26,647	12,689
Jumlah			165,920	120,951
Rata - Rata			9,218	6,720

Dari data diatas didapat eror rata – rata untuk warna merah adalah : 6,720 %.

4.2 Pengujian Gerakan Robot

Pengujian gerakan robot ini dilakukan dengan meletakkan benda didepan dari robot kemudian robotnya dijalankan. Sehingga robot akan menentukan *path tracking* sendiri, menghindari rintangan dan menemukan target. Pengujian gerakan robot dapat dilihat pada gambar 4.2.



Gambar. 4.2. Posisi *Obstacle,* target dan robot.

Tabel. 4.6. Pengujian Gerak Robot

No	Jarak Benda (cm)		Benda Hijau				Benda Biru				Benda Merah				Benda Kuning				Benda Orange			
			Algoritma				Algoritma				Algoritma				Algoritma				Algoritma			
			Pure Pursuit (%)		Follow-The-Carrot (%)		Pure Pursuit (%)		Follow-The-Carrot (%)		Pure Pursuit (%)		Follow-The-Carrot (%)		Pure Pursuit (%)		Follow-The-Carrot (%)		Pure Pursuit (%)		Follow-The-Carrot (%)	
	Jo	Ji	Berhasil	Gagal	Berhasil	Gagal	Berhasil	Gagal	Berhasil	Gagal	Berhasil	Gagal	Berhasil	Gagal	Berhasil	Gagal	Berhasil	Gagal	Berhasil	Gagal	Berhasil	Gagal
1	30	20	80	20	10	90	80	20	10	90	80	20	10	90	80	20	10	90	80	20	10	90
2	40	20	90	10	10	90	80	20	20	80	90	10	20	80	90	10	20	80	90	10	10	90
3	50	20	90	10	80	20	90	10	80	20	90	10	80	20	90	10	80	20	90	10	90	10
4	60	20	90	10	90	10	90	10	90	10	90	10	90	10	90	10	90	10	90	10	90	10
5	30	25	80	20	10	90	80	20	10	90	80	20	10	90	80	20	10	90	80	20	10	90
6	40	25	90	10	10	90	80	20	20	80	90	10	20	80	90	10	20	80	90	10	10	90
7	50	25	90	10	80	20	90	10	80	20	90	10	80	20	90	10	80	20	90	10	90	10
8	60	25	90	10	90	10	90	10	90	10	90	10	90	10	90	10	90	10	90	10	90	10
9	30	30	80	20	10	90	80	20	10	90	80	20	10	90	80	20	10	90	80	20	10	90
10	40	30	90	10	10	90	80	20	20	80	90	10	20	80	90	10	20	80	90	10	10	90
11	50	30	90	10	80	20	90	10	80	20	90	10	80	20	90	10	80	20	90	10	90	10
12	60	30	90	10	90	10	90	10	90	10	90	10	90	10	90	10	90	10	90	10	90	10

Dari data tebel 4.6, diperoleh bahwa warna tidak terlalu berpengaruh terhadapat pergerakan dari robot. Baik untuk algoritma *pure-pursuit* maupun *follow-the-carrot*. Karena robot dalam menentukan *path* robot mendekati *obstacle* yang terdekat dari posisinya, baru kemudian robot tersebut melakukan gerakan menuju target. Dengan dekatnya jarak robot dengan *obstacle*, maka kesalahn pengukuran dengan kamera dapat diatasi.

Sedangkan jarak antara *obstacle* tidak berpengaruh terhadap penghindaran robot menuju ke target, sedangkan jika jarak antar *obstacle* ini lebih besar dari lebar robot, maka robot akan lewat ditengah-tengah *obstacle* tersebut untuk menuju ke target.

Kegagalan pada algoritma *pure pursuit* dalam menuju ke target, selalu menabrak target, hal ini disebabkan karena perhitungan lengkungan *path* untuk menuju taget dipengaruhi oleh pengukuran dari kamera, sehingga lengkungannya lebih. Sedangkan kegagalan daripada algoritma *follow-the-carrot*, selalu menabrak *obstacle,* ini juga kegagalan dalam pengukuran jarak target dengan *obstacle* oleh kamera.

Jarak antara *obstacle* dan target (Jo), pada algoritma *pure pursuit,* makin dekat maka tingkat keberhasilan semakin tinggi, sedangkan jika untuk *follow the carrot*, makin dekat mengalami kegagalan yang besar. Karena dengan dekatnya jarak Jo, maka jarak Lf akan panjang. Sedangkan untuk referensi (sensor) posisi dari robot tidak ada, sehingga jarak Lf yang akurat tidak didapat.

BAB V
PENUTUP

BAB V
PENUTUP

5.1. Kesimpulan

Dari hasil perancangan dan pengujian seluruh sistem pada tesis ini didapat data dan kemudian kita analisa hasilnya. Sehingga didapat beberapa kesimpulan :

1. Pengukuran jarak terhadapat beberapa target benda berwarna diperoleh hasil eror rata – rata sebagai berikut:
    a. Biru eror rata – rata = 8.294 %
    b. Kuning eror rata – rata = 7,244 %
    c. Hijau eror rata – rata = 6,049 %
    d. Orange eror rata – rata = 6,049 %
    e. Merah eror rata – rata = 6,720 %
2. Untuk melakukan pergerakan penghindaran rintangan, maka algoritma *pure pursuit* lebih baik digunakan dari pada *follow-the-carrot*

5.2. Saran

Untuk peningkatan unjuk kerja robot dalam melakukan pencarian target dan penghindaran *obstacle*, perlu meningkatkan sensitifitas sensor citra (kamera), dan menambah sensor kecepatan.

DAFTAR PUSTAKA

DAFTAR PUSTAKA

[1] Coulter,R C., (1992), ***Implementation of the Pure Pursuit Path Tracking Lagorithm***, Robotics Institute, Carnegie Mellon University, January.

[2] Wit,J S., (2000),***Vector Pursuit Path Tracking for Autonomous Ground Vehicle.*** Ph.D Thesis. University of Florida.

[3] Lundgren, Martin,(2003), ***Path Tracking and Obstacle Avoidance for A Miniature Robot,*** Master Thesis. Umea University.

[4] Suk,Jong & Byung Kook Kim,(2001), ***Near-Time-Optimal Trajectory Planning for Wheeled Mobile Robots with Translation and Rotational Section,*** IEEE Transactions on Robotics and Automation, Vol. 17, No. 1, February 2001, pp 85-90

[6] Lucas, G W.,(2001), ***A Tutorial and Elementary Trajectory Model for the Differential Steering System of Robot Wheel Actuator.*** The Rossum Project.

[7] Wit, J S., Carl D. Crane, David Armstrong, (2004) ***Autonomous Ground Vehicle Path Tracking,*** Journal of Robotic System, 2004, pp. 390-449

[8] "Unmanned vehicle : University of Florida" http://www.me.ufl.edu/~webber/web/pages/reserach areas/vehcile control.htm (tanggal akses : 23 Februari 2005)

[9] Zabarankin, M., (2003), **Vision Based Trajectory Planning For Autonomous MAVs In Urban Environments**, Research Report 2003-001, University Of Florida Research Center, Dec 31, 2003

[10] Everett, H.R., (1995), **Sensors For Mobile Robot (Theory and Application)**, A.K. Peters Ltd. Massachusets.

[11] " Morphology at Oregen State Unversity" http://www.engr.oregonstate.edu/~luca/ece568/Ch2.pdf (tanggal akses 2 Mei 2005)

[12] Awcock,G.J., R. Thomas, (1996), ***Applied Image Processing,*** McGrawHill.

[13] " Imaging and Image Representation at University of Masschusetts" http://www.cis.umassd.edu/~rbalasubrama/CIS465/S04/Slides/chapter2.ppt (tanggal akses 2 Mei 2005)

[14] Stockman, Sapiro, ***Computer Vision,*** Massachusette University, 2003

[15] H. E. Burdick, ***Based on material from Digital Imaging: Theory and Applications***, McGraw-Hill, 1997

[16] " Statistics of image at Bersoft" http://bersoft.com/bimagem/help/statistics image.htm (tanggal akses 30 Mei 2005)

[17] "Color Space at NC State University" http://www.ncsu.edu/scivis/lessons/colormodels/color_models2.html (tanggal akses 2 Mei 2005)

[18]"Transformation RGB to HSV and Transformation HSV to RGB at Definition" http://www.free-definition.com/HSV-color-space .html (tanggal akses 2 Mei 2005)

[19] " Speed Control at WichOnLine Webspace" http://www.homepage.which.net/~paul.hills/SpeedControl.html (tanggal akses 1 Juni 2005)