

**z**

**TUGAS AKHIR – TE 091399**

**PENGEMBANGAN ROBOT RUBIK DENGAN IDENTIFIKASI WARNA MENGGUNAKAN SENSOR KAMERA**

Maulana Rachman

NRP 2206100071

Dosen Pembimbing

Dr.Ir. Djoko Purwanto, M. Eng.

Ir. Karyadi, MSc

JURUSAN TEKNIK ELEKTRO

Fakultas Teknologi Industri

Institut Teknologi Sepuluh Nopember

Surabaya 2012



**FINAL PROJECT – TE 091399**

**DEVELOPMENT OF ROBOT RUBIK WITH IDENTIFICATION OF COLORS USING CAMERA SENSOR**

Maulana Rachman

NRP 2206100071

Supervisor

Dr.Ir. Djoko Purwanto, M. Eng.

Ir. Karyadi, MSc

ELECTRICAL ENGINEERING DEPARTMENT

Faculty of Industrial Technology

Institute Technology of Sepuluh Nopember

Surabaya 2012



**TUGAS AKHIR – TE 091399**

**PENGEMBANGAN ROBOT RUBIK DENGAN IDENTIFIKASI WARNA MENGGUNAKAN SENSOR KAMERA**

Maulana Rachman

NRP 2206100071

Dosen Pembimbing

Dr.Ir. Djoko Purwanto, M. Eng.

Ir. Karyadi, MSc

JURUSAN TEKNIK ELEKTRO

Fakultas Teknologi Industri

Institut Teknologi Sepuluh Nopember

Surabaya 2012

PERNYATAAN KEASLIAN

TUGAS AKHIR

Dengan ini saya menyatakan bahwa isi sebagian maupun keseluruhan Tugas Akhir saya dengan judul “Pengembangan Robot Rubik dengan Identfikasi Warna Menggunakan Sensor Kamera” adalah benar-benar hasil karya intelektual mandiri, diselesaikan tanpa menggunakan bahan-bahan yang tidak diijinkan dan bukan merupakan karya pihak lain yang saya akui sebagai karya sendiri.

Semua referensi yang dikutip maupun dirujuk telah ditulis secara lengkap pada daftar pustaka.

Apabila ternyata pernyataan ini tidak benar, saya bersedia menerima sanksi sesuai peraturan yang berlaku.

Surabaya, 3 Februari 2012

Maulana Rachman

NRP. 2206100071

**PENGEMBANGAN ROBOT RUBIK DENGAN IDENTIFIKASI WARNA MENGGUNAKAN SENSOR KAMERA**

**TUGAS AKHIR**

Diajukan Guna Memenuhi Sebagian Persyaratan

Untuk Memperoleh Gelar Sarjana Teknik

Pada

Bidang Studi Elektronika

Jurusan Teknik Elektro

Fakultas Teknologi Industri

Institut Teknologi Sepuluh Nopember

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Menyetujui: | |  |  |  |
| DosenPembimbing I, | | DosenPembimbing II, | | |
| Dr.Ir. Djoko Purwanto, M.Eng. |  |  | Ir. Karyadi, MSc |  |
| NIP.196512111990021002 | | NIP. 194608121974121001 | | |

**SURABAYA**

**FEBRUARI, 2012**

**ABSTRAK**

Rubik adalah permainan teka-teki berbentuk kubus yang sangat populer di kalangan dunia saat ini. Tugas akhir ini bertujuan untuk mendesain sistem yang dapat mendeteksi secara automatis posisi rubik 3x3 dengan menggunakan *image processing* dan mengidentifikasi warna pada setiap sisi rubik. Informasi data warna akan digunakan sebagai *input* yang diseleksi ke dalam *memory*, hasil dari penyeleksian dibuatmenjadi daftar gerakan-gerakan untuk menyelesaikan rubik kembali ke posisi semula. *Software* menggunakan Visual C++ 2010 Express yang mendukung untuk pembuatan *image processing* pada kamera dan *hardware* menggunakan motor servo yang dikontrol dengan minimumsistem Arduino. Dari hasil pengujian sistem pergerakan robot, pendeteksian dan indentifikasi warna didapatkan hasil dengan proses keberhasilan 70% dan metode penyelesaian algoritma rubik masih perlu dikembangkan sehingga robot dapat menyelesaikan rubik tanpa menggunakan data *memory*.

**Kata kunci :** Rubik,*image processing*, kamera, minimum sistem Adruino.

i

*- Halaman ini sengaja dikosongkan -*

ii

***ABSTRACT***

*Rubik is a puzzle game that shaped cube is very popular around the world. Purpose of this thesis was to design a system which is able to automatically detect the scrambled state of a Rubik’s Cube 3x3 using image processing and analysis methods. The state information is used as an input that is selected into the memory. The output of the solving algorithm is a list of necessary movements to bring the cube into its initial ordered state. The software is used Visual C++ 2010 Express thats supports for making image processing with camera and the hardware is used servo motor to be controlled by minimum system Arduino. From the result of robot movement system, detection dan color identifiacation obtained with the success of 70% and the method of solving Rubik algorithms still need to be developed so the robot can solve Rubik without using a data memory.*

***Keywords:*** *Rubik’s cube, image processing, camera, minimum system**Arduino.*

.

iii

*- Halaman ini sengaja dikosongkan -*

iv

**KATA PENGANTAR**

Puji syukur kepada Tuhan Yang Maha Esa atas kasih dan rahmat-

Nya penulis dapat menyelesaikan Tugas Akhir ini dengan judul :

**Pengembangan Robot Rubik dengan Identifikasi Warna Menggunakan Sensor Kamera**

Tugas Akhir ini merupakan salah satu persyaratan yang harus dipenuhi dalam menyelesaikan pendidikan program Strata-Satu di Jurusan Teknik Elektro, Fakultas Teknologi Industri, Institut Teknologi Sepuluh Nopember Surabaya.

Pada kesempatan ini, penulis ingin mengucapkan terima kasih kepada semua pihak yang telah membantu dalam pembuatan tugas akhir ini, khususnya kepada:

1. Bapak, Ibu, kakak serta seluruh keluarga yang telah memberikan dukungan moril dan materiil.
2. Dr.Ir. Djoko Purwanto, M Eng. selaku dosen pembimbing 1 dan Ir. Karyadi, MSc selaku dosen pembimbing 2 atas bimbingan dan arahan selama penulis mengerjakan tugas akhir ini.
3. Seluruh dosen pengajar dan staff Jurusan Teknik Elektro yang telah banyak memberikan ilmu dan pengalaman selama penulis menempuh kuliah.
4. Teman-teman laboratorium Elektronika yang tidak dapat disebutkan satu persatu telah membantu penulis dalam proses pengerjaan tugas akhir ini.

Akhirnya penulis berharap agar Tugas Akhir ini dapat memberikan manfaat yang sebesar-besarnya bagi semua pihak dalam upaya untuk mengembangkan ilmu pengetahuan tentang *image processing*. Selain itu penulis juga berharap supaya Tugas Akhir ini dapat dikembangkan semaksimal mungkin untuk aplikasi yang lebih bermanfaat.

Surabaya, Februari 2012

Penulis

v

*- Halaman ini sengaja dikosongkan -*

vi

**DAFTAR ISI**

**ABTRAK** **i**

***ABSTRACT*** **iii**

**KATA PENGANTAR** **v**

**DAFTAR ISI** **vii**

**DAFTAR GAMBAR** **ix**

**DAFTAR TABEL** **xi**

**DAFTAR GRAFIK** **xii**

**BAB I PENDAHULUAN** **1**

1.1 Latar Belakang 1

1.2 Permasalahan 1

1.3 Tujuan 2

1.4 Batasan Masalah 2

1.5 Metodologi 2

1.6 Sistematika Penulisan 3

1.7 Relevansi 4

**BAB II TEORI PENUNJANG** **5**

2.1 Citra Digital 5

2.2 Pengolahan Citra Digital 6

2.2.1 *Greyscaling* 7

2.2.2 *Smoothing* 8

2.2.3 *Laplace* 10

2.3 Pengenalan Pola Citra Digital 11

2.3.1 *Hough Tranform* 12

2.4 Skala Warna HSV 13

2.5 Motor Servo 15

2.6 Konfigurasi Rubik 16

**BAB III PERENCANAAN SISTEM** **19**

3.1 Kamera Sebagai Sensor Visual 20

3.2 *Central Processing Unit* 21

3.3 Perancanaan Perangkat Keras 22

3.3.1 Pengontrolan Motor Servo Horizontal 23

3.3.2 Pengontrolan Motor Servo Vertikal 24

3.3.3 Minimum Sistem Arduino Duemilanove 25

3.3.4 *Liquid Crystal Display* 28

vii

3.4 Pengolahan dan Pengenalan Pola Citra Digital 31

3.5 Pembacaan Warna 42

**BAB IV PENGUJIAN ALAT** **45**

4.1 Pengujian Gerak Robot 45

4.2 Pengujian Perangkat Lunak 45

4.2.1 Pengujian Deteksi Rubik 46

4.2.2 Pengujian Pengidentifikasi Warna Rubik 48

4.3 Pembahasan Permasalahan 53

**BAB V PENUTUP** **57**

5.1 Kesimpulan 57

5.2 Saran 57

**DAFTAR PUSTAKA**

**LAMPIRAN**

**BIODATA PENULIS**

viii

**DAFTAR GAMBAR**

Gambar 2.1 Citra Lena dan Citra Rubik 6

Gambar 2.2 Blok Sistem Pengolahan Citra Digital 7

Gambar 2.3 Citra Lena yang Mengandung Derau dan Hasil dari

Penapisan Citra 7

Gambar 2.4 Gambar Berwarna dan Gambar *Greyscale* 8

Gambar 2.5 Kernel Operator *Gaussian* dengan ** = 1,4 10

Gambar 2.6 Sebelum dan Sesudah 10

Gambar 2.7 Efek *Laplcae Filter* yang Kedua (-1,8,-1) 11

Gambar 2.8 Blok Sistem Pengenalan Pola Citra Digital 11

Gambar 2.9 Citra Karakter ‘A’ yang Digunakan Sebagai Masukan

untuk Pengenalan Huruf 12

Gambar 2.10 Representasi Hough Transform 13

Gambar 2.11 Hough Transform dalam Koordinat Polar 13

Gambar 2.12 Gambar *Cone* yang Merepentasikan Nilai Skala Warna

HSV 14

Gambar 2.13 Teknik PWM Untuk Mengatur Motor Servo 15

Gambar 2.14 *Pin Out* Kabel Motor Servo 16

Gambar 2.15 Bagian Tepi 16

Gambar 2.16 Bagian Sudut 16

Gambar 2.17 Bagian Tengah 17

Gambar 2.18 Setiap Langkah Dipresentasikan dengan Huruf 17

Gambar 2.19 Huruf dengan ‘i’ adalah Inverse dari Pergerakan

Tersebut 18

Gambar 3.1 Diagram Blok Sistem 19

Gambar 3.2 Logitech Webcam C500 20

Gambar 3.3 Posisi Kamera Sebagai Sensor Visual Terhadap Rubik 21

Gambar 3.4 Laptop Aspire 4736G 22

Gambar 3.5 Perangkat Keras Berupa Dua Motor Servo, Minimum

sistem Arduino dan LCD 23

Gambar 3.6 Blok Diagram Sistem Pergerakan Motor Servo 24

Gambar 3.7 Servo Motor Pada Saat Dalam Posisi Normal, Kekiri dan

Kekanan 24

Gambar 3.8 Pergontrolan Motor Servo Vertikal 25

Gambar 3.9 Arduino Main Board Berbasis Atmega328 dengan

Bootloader 26

Gambar 3.10 Contoh Program Arduino Menggunakan Java 27

ix

Gambar 3.11 Breadboard Rangkaian Robot Rubik Menggunakan

Arduino 28

Gambar 3.12 Hellow World di 2x16 LCD 28

Gambar 3.13 *Schematic* Robot Rubik Menggunakan Fritzing 30

Gambar 3.14 Contoh Program LCD di Arduino 30

Gambar 3.15 Citra Dalam 320x240 pixels 31

Gambar 3.16 Design dan Implementasi 32

Gambar 3.17 Setelah Mengalami Lapalcian Filter 32

Gambar 3.18 Tepi-tepi yang Terdeteksi *Hough Transform* Setelah

Menggunakan *Adaptive Threshold* 33

Gambar 3.19 Contoh Dua Garis yang Saling Berpotongan dan Tidak

Berpotongan 35

Gambar 3.20 Hasil Citra Setelah Dicari Persamaan Garis 36

Gambar 3.21 Trigonometri 37

Gambar 3.22 Kuadran Dalam Grafik Pixel 38

Gambar 3.23 Hasil Citra Setelah Dicari Sudut yang Tegak Lurus 38

Gambar 3.24 Hasil Seleksi Garis Setelaj Dicari Garis dan Sudut 39

Gambar 3.25 Visualisasi dari Sistem Koordinat Dalam Grid 40

Gambar 3.26 Posisi Rubik Terhadap Sistem Koordinat 40

Gambar 3.27 Rubik yang Terdeteksi Citra 41

Gambar 3.28 Bekerja dengan Kedalaman Warna yang Berbeda 43

Gambar 3.29 Diagram Sistem Kerja Program Alat 44

Gambar 4.1 Contoh Rubik yang Jatuh 54

Gambar 4.2 Contoh Posisi *LK tracking* yang Bergeser 54

x

**DAFTAR TABEL**

Tabel 3.1 Perintah *Library* LCD Pada Arduino 29

Tabel 4.1 Hasil Pengujian Pergerakan Robot 45

Tabel 4.2 Hasil Pengujian Deteksi Rubik Pada Siang Hari 46

Tabel 4.3 Hasil Pengujian Deteksi Rubik Pada Malam Hari 47

Tabel 4.4 Hasil Pengujian Identifikasi Warna Rubik Pada Siang

Hari 49

Tabel 4.5 Hasil Pengujian Identifikasi Warna Rubik Pada Malam

Hari 51

Tabel 4.6 Hasil Pengujian Identifikasi Warna dan Penyelesaian

Rubik 54

xi

*- Halaman ini sengaja dikosongkan -*

xii

**DAFTAR GRAFIK**

Grafik 3.1 Empat Garis Vertikal yang Memiliki jarak yang Berbeda-

beda 34

Grafik 3.2 Dua Garis yang Membentuk Sudut Dalam Pixel 37

xiii

*- Halaman ini sengaja dikosongkan -*

xiv

**BAB I**

**PENDAHULUAN**

**1.1 Latar Belakang**

Rubik, diciptakan oleh Erno Rubik pada tahun 1974, penjualan mainan teka-teki terbaik dalam sejarah. Dalam rubik, terdapat 26 individu kubus kecil (cubies) yang membentuk rubik besar. Setiap lapisan dari sembilan cubies dapat berputar dan tiap layer dapat tumpang tindih. Rubik bisa memiliki 43 triliun posisi yang berbeda-beda. Meskipun mungkin sebagian besar posisi tersebut hanya dapat diselesaikan dalam 29 gerakan atau kurang.

*Rubik’s Cube Solving Robot* (RUCUS) terdiri dari tiga sistem dasar

* image processing dan analisis, *solving-algorithm computation*, dan sistem kontrol pada robot.

Tujuan dari *image processing* dan analisis pada sistem bertujuan untuk mendeteksi posisi yang diacak pada rubik, kamera akan mengambil gambar dan menganalisis untuk mendeteksi posisi dari semua rubik dan warna yang sesuai. Output dari image processing berupa string yang berisi informasi warna dari 54 warna menggunakan format : RGGRYWWR...; setiap huruf menyusaikan warnanya, dan posisi dari huruf menentukan posisi pada rubik.

Informasi ini akan diproses menggunakan solving-algorithm computation. Sistem ini menggunakan algoritma yang sudah ada dan diklasifikasi untuk mencari langkah paling cepat dalam menyelesaikan rubik diantara semua algoritma-algoritma yang diperoleh dan input warna berupa string untuk menghitung daftar gerakan yang akan mengubah rubik yang dipecahkan ke dalam *solving state*. Gerakan-gerakan tersebut akan dilakukan oleh robot.

**1.2 Permasalahan**

Permasalahan yang dibahas dalam tugas akhir ini adalah:

1. Bagaimana cara mendeteksi warna pada rubik menggunakan kamera pada OpenCV ?
2. Bagaimana cara mengintegrasikan Visual C++ dengan minimum sistem Adruino pada robot ?
3. Bagaimana mengatur kontrol pada robot ?

1

**1.3 Tujuan**

Penelitian pada tugas akhir ini bertujuan untuk mendesain sistem yang dapat mendeteksi secara automatis keadaan posisi rubik 3x3 yang diacak menggunakan *image processing* dan metode analisa. Informasi data tersebut digunakan sebagai masukan yang akan dikomputasi ke penyelesaian algoritma. Hasil dari penyelesaian algoritma akan dibuat menjadi daftar gerakan-gerakan untuk memutar rubik berubah menjadi kekeadaan awal semula.

**1.4 Batasan Masalah**

Dalam pengerjaan tugas akhir, permasalahan di atas dibatasi dengan asumsi sebagai berikut :

a. Kamera yang digunakan adalah kamera webcam yang mempunyai kecepatan minimal 30 frame per second video dan 1.3 megapixel.

1. Pengambilan data diperlukan pencahayaan yang cukup untuk dapat mendeteksi rubik.

**1.5 Metodologi**

Langkah-langkah yang dikerjakan pada Tugas Akhir ini adalah sebagai berikut :

1. Studi Literatur.
   * Mempelajari prinsip *Image Processing* dalam teknologi *Machine Vision* yaitu akuisisi citra digital dan pemrosesancitra digital.
   * Mempelajari pemrograman visual menggunakan Microsoft Visual C++ 2010 Exprees dan *Open Source Computer* *Vision Library* (OpenCV).
2. Perencanaan dan pembuatan perangkat keras
   * Menyediakan kamera webcam sebagai sensor visual.
   * Menyediakan minimum sistem Adruino.
   * Merancang desain mekanik robot.
   * Menyediakan dua motor servo.
3. Perencanaan dan pembuatan perangkat lunak, meliputi :
   * Perencanaan dan pembuatan program kontrol pergerakan robot menggunakan servo motor pada Adruino.

2

* + Perencanaan dan pembuatan program *detection* dan *tracking* rubik yang bertujuan untuk mendeteksi warnapada rubik serta penyelesain algoritma rubik.

1. Integrasi perangkat keras dengan perangkat lunak
   * Penggabungan sistem hardware yang mengambil data dari pergerakan rubik dan software yang digunakan untuk menampilkan secara visual.
2. Pengujian dan analisa
   * Pada tahap ini dilakukan pengujian terhadap sistem yang telah dibuat. Dengan parameter keberhasilannya adalah dapat meidentifikasi warna tersebut secara benar.
   * Apabila hasil yang diperoleh dari tahap pengujian terjadi ketidaksesuaian pada alat yang telah dibuat atau mungkin kurang memuaskan secara bentuk dan ketelitian penyelesaian, maka perlu diadakan evaluasi pada software dan hardware serta sistem secara keseluruhan. Kemudian dilakukan pengujian ulang sampai parameter keberhasilan telah dicapai.
3. Penulisan buku Tugas Akhir

**1.6 Sistematika Penulisan**

Dalam buku Tugas Akhir ini, pembahasan mengenai sistem yang dibuat terbagi lima bab dengan sistematika sebagai berikut :

* [BAB I](#page23) : [PENDAHULUAN](#page23)

Pada bagian ini menjelaskan beberapa sub bagian yang antara lain berisi [Latar Belakang, Permasalahan,](#page23) [Tujuan, Batasan Masalah,](#page24) [Metodologi,](#page24) [Sistematika Penulisan,](#page25) dan [Relevansi](#page26) penulisan Tugas Akhir ini.

* [BAB II](#page27) : [TEORI PENUNJANG](#page27)

Pada bagian ini berisi tentang landasan teoru yang digunakan dalam pelaksanaan Tugas akhir yang meliputi Citra Digital, Pengolahan Citra Digital, Skala Warna HSV, Konfigurasi Rubik dan mikrokontroler Arduino. Bagian ini memaparkan tentang beberapa teori penunjang dan beberapa literatur yang berguna bagi pembuatan Tugas Akhir ini.

3

* [BAB III](#page41) : [PERENCANAAN](#page41) SISTEM

Pada bagian ini berisi tentang alogritma kerja program dan pemodelan sistem robot rubik ke dalam perangkat lunak yang berguna dalam menyelesaikan Tugas Akhir ini.

* [BAB IV](#page67) : [PENGUJIAN ALAT](#page67)

Pada bagian ini akan menjelaskan tentang cara kerja sistem robot rubik dan hasil uji coba sistem beserta analisanya.

* [BAB V](#page79) : [PENUTUP](#page79)

Bagian ini merupakan bagian akhir yang berisikan kesimpulan

yang diperoleh dari pembuatan Tugas Akhir ini, serta saran-saran untuk pengembangannya.

**1.7 Relevansi**

Pembuatan Robot Rubik dengan sensor kamera merupakan salah satu langkah pengembangan dari aplikasi kamera digital pada bidang robotik, sehingga dapat membantu perkembangan aplikasi robot dalam menirukan kemampuan manusia lainnya dalam hal visualisasi sensor robot, dan dapat dikembangkan dalam aplikasi rubik yang dimainkan oleh robot.

4

**BAB II**

**TEORI PENUNJANG**

**2.1 Citra Digital**

Data atau informasi tidak hanya disajikan dalam bentuk teks, tetapi juga dapat berupa gambar, audio (bunyi, suara, musik), dan video. Keempat macam data atau informasi ini sering disebut multimedia. Era teknologi informasi saat ini tidak dapat dipisahkan dari multimedia. Situs web (website) di Internet dibuat semenarik mungkin dengan menyertakan visualisasi berupa gambar atau video yang dapat diputar. Beberapa waktu lalu istilah SMS *(Short Message Service)* begitu populer bagi pengguna telepon genggam (*handphone* atau HP). Tetapi, saat ini orang tidak hanya dapat mengirim pesan dalam bentuk teks, tetapi juga dapat mengirim pesan berupa gambar maupun video, yang dikenal dengan layanan MMS (*Multimedia Message Service*).

Citra (*image*) sebagai salah satu komponen multimedia memegang peranan sangaat penting sebagai bentuk informasi visual. Citra mempunyai karakteristik yang tidak dimiliki oleh data teks, yaitu citra kaya dengan informasi. Ada sebuah peribahasa yang berbunyi “sebuah gambar bermakna lebih dari seribu kata” (*a picture is more than a* *thousand words*). Maksudanya tentu sebuah gambar dapat memberikaninformasi yang lebih banyak daripada informasi tersebut disajikan dalam bentuk kata-kata (tekstual).

Citra (*image*) adalah gambar pada bidang dwimatra (dua dimensi). Gambar 2.1 adalah citra seorang gadis model yang bernama Lena, dan gambar disebelah kanannya adalah citra sebuah rubik. Ditinjau dari sudut pandang matematis, citra merupakan fungsi menerus (*continue*) dari intensitas cahaya pada bidang dwimatra. Sumber cahaya menerangi objek, objek memantulkan kembali sebagian dari berkas cahaya tersebut. Pantulan cahaya ini ditangkap oleh alat-alat optik, misalnya mata pada manusia, kamera, pemindai (*scanner*), dan sebagainya, sehingga bayangan objek yang disebut citra tersebut terekam.

Citra sebagai keluaran dari sistem perekaman data dapat bersifat :

1. Optik berupa foto,
2. Analog berupa sinyal video seperti gambar pada monitor televisi,
3. Digital yang dapat langsung disimpan pada suatu pita magnetik.

5

Citra yang dimaksudkan di dalam keseluruhan isi buku ini adalah “citra diam” (*still images*). Citra diam adalah citra tunggal yang tidak bergerak. Gambar 2.1 adalah dua buah citra diam. Untuk selanjutnya, citra diam kita sebut citra saja.



(a) (b)

**Gambar 2.1** (a) Citra Lena dan (b) Citra Rubik[1]

Citra bergerak (*moving images*) adalah rangkaian citra diam yang ditampilkan secara (sekuensial) sehingga memberi kesan pada mata kita sebagai gambar yang bergerak. Setiap citra di dalam rangkaian itu disebut *frame*. Gambar-gambar yang tampak pada film layar lebar atau televisi pada hakikatnya terdiri atas ratusan sampai ribuan *frame*.

**2.2 Pengolahan Citra Digital**

Pengolahan citra bertujuan memperbaiki kualitas citra agar mudah diinterpretasi oleh manusia atau mesin (dalam hal ini komputer). Teknik-teknik pengolahan citra mentransformasikan citra menjadi citra lain. Jadi, maksudnya adalah citra dan keluaranya juga citra, namun citra keluaran mempunyai kualitas lebih baik daripada citra masukan. Termasuk kedalam bidang ini juga adalah pemampatan citra (image compression).

6

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
| citra | Citra |  | citra |  |
|  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |



**Gambar 2.2** Blok sistem pengolahan citra digital

Pengubahan kontras citra seperti pada Gambar 2.2 adalah contoh operasi pengolahan citra. Contoh operasi pengolahan citra lainnya adalah penghilangan derau (*noise*) pada citra Lena (Gambar 2.3). Citra Lena yang disebelah kiri mengandung derau berupa bintik-bintik putih (derau). Operasi penapisan *(filtering)* yang akan dibahas adalah *grayscaling*, *smoothing*, dan *Laplace*.



(a) (b)

**Gambar 2.3** (a) Citra Lena yang mengandung derau, (b) Hasil daripenapisan derau[1]

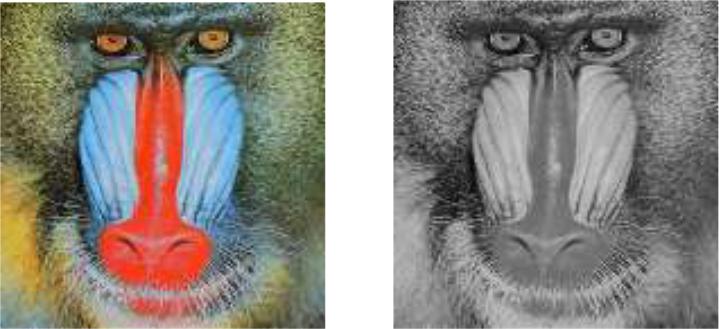
**2.2.1** ***Greyscaling***

*Greyscaling* adalah teknik yang digunakan untuk mengubah citraberwarna menjadi bentuk citra monokrom (hitam - putih). Pada citra berwarna yang terdiri dari 3 komponen warna R,G,B sedangkan pada citra monokrom hanya terdiri dari 1 komponen warna. Pengubahan citra berwarna menjadi citra *greyscale* mengikuti kaidah berikut :



(2.1)

7



(a) (b)

**Gambar 2.4** (a) Gambar berwarna dan (b) Gambar*Grayscale*[1]

**2.2.2** ***Smoothing***

Perbaikan citra dengan *smoothing filter* bertujuan untuk menghilangkan noise, memperjelas sisi (*sharp edge*) dalam sebuah image. *Smoothing filter* dibagi lagi menjadi empat jenis tipe : *blur,* *gaussian, median,* dan *bilateral*. Metode yang saya gunakan adalah *gaussian filter*. *Gaussian Filtering* adalah filter yang ideal yangmampu mengurangi besarnya frekuensi spasial yang tinggi dalam foto sebanding dengan frekuensi mereka. Artinya, metode ini mampu mengurangi besarnya frekuensi spasial yang tinggi lagi. *Gaussian* *Filtering* mampu meluas hingga ke segala arah, tetapi karenamendekati nol secara eksponensial, dapat dipotong tiga atau empat standar deviasi dari pusat tanpa mempengaruhi hasilnya.

*Gaussian filtering* mampu bekerja lebih cepat dari metode meandan median dengan memisahkan sebuah *Gaussian* 2-D menjadi dua *Gaussian* 1-D, G (x, y) = G (x) G (y), dan melakukan *Filtering* dalam1-D, baris demi baris dan kemudian kolom ke kolom. Dasar dari *Gaussian filtering* dapat dirumuskan sebagai berikut:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| *h**x*, *y* *g* | | 2*D* | | *x*, *y* | | |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | |  |  | 1 |  |  |  | 2 |  |  |   | | | 1 |  |  |  | 2 | 2 |  |  |  |  |  |
|  |  |  |  *x* | |  | 2 |  |  |  |  |  |  | 2 |  |  |
|  |  |  |  |  |  |  |  | 2** |  |  |   | |  |  |  |  |  |  |  | 2** |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|   | |  | 2** | | | *e* |  |  |  |  |  |   | | 2** | | *e* |  |  |  |  |  |  |  |  |
|  | |  |  |  |  |  |  |  |  | |  |  |  |  |  |  |  |  |  |
|  *g*1*D* *x* *g*1*D* *y* | | | | | | | | |  |  |  |  |  |  |  |  |  |  |  |  |  |  | (2.2) |  |



8

dimana,

h(x, y)

g2D (x, y)

**

g1D (x)

g1D (y)

* Gaussian filter 2-D pixel (x, y)
* Gaussian filter 2-D pixel (x, y)
* *Bandwith Kernel*, Standar Populasi Kernel
* Gaussian filter 1-D line (x)
* Gaussian filter 1-D line (y)

Ada beberapa cara yang berbeda untuk mengimplementasikan *Gaussian filtering*, salah satunya menggunakan *Spatial Filter*,konvulusi ini menggunakan Operator *Gaussian Smoothing* 2-D yang mirip dengan filter 1-D, tetapi menggunakan kernel yang berbeda yang mewakili bentuk *Gaussian* 1-D untuk nilai X dan nilai *Gaussian* 1-D untuk nilai Y. Sehingga konvulusi ini termasuk tipe non linear kernel. Dalam *Spatial Filter* digunakan persamaan :

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  | 2 |  *y* | | 2 |  |
|  | *x*, *y* | 1 |  |  |  | *x* |  |  |
| *G* |  | *e* | 2** | | 2 |  |  |
|  | 2 |  |  |  |  |
| 2 *D* |  | 2** |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |



(2.3)

Ide *Gaussian smoothing* ini adalah dengan menggunakan distribusi ini 2-D sebagai fungsi ‘titik-menyebar’ atau lebih dikenal dengan ‘*point-spread*’, dan ini dicapai dengan konvolusi. Gambar disimpan sebagai koleksi *discrete pixels*, sehingga perlu dihasilkan pendekatan diskrit ke fungsi *Gaussian* sebelum kita dapat melakukan konvolusi tersebut. Secara teori, distribusi *Gaussian* adalah nol-nol di mana-mana, yang akan membutuhkan sebuah *kernel* konvolusi besar tak berhingga, tetapi dalam prakteknya nol efektif lebih dari sekitar tiga standar deviasi dari mean, dan sehingga dapat dilakukan pemotongan *kernel*.

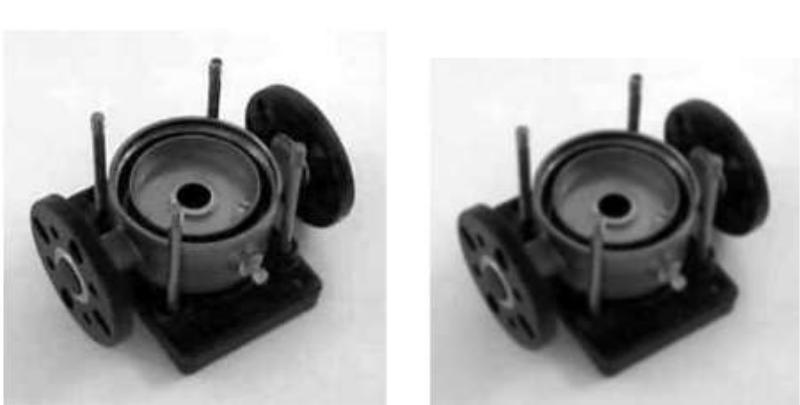
|  |  |  |  |
| --- | --- | --- | --- |
| Operator berikut akan menunjukan sebuah *kernel* konvolusi yang | | |  |
| memiliki nilai integer sebuah *Kernel Gaussian* yang mendekati | ** | = |  |
|  |  |
| 1,4. |  |  |  |

9

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 2 | | 4 | 5 | 4 | 2 | |  |
|  |  | 4 | 9 | 12 | 9 | 4 |  |  |
| 1 |  |  |  |
|  |  |  |  |  |  |
| 5 | | 12 | 15 | 12 | 5 | |  |
|  |  |
| 159  | | 4 | 9 | 12 | 9 | 4 |  |  |
|  |  |  |  |
|  |  | 2 | 4 | 5 | 4 | 2 |  |  |
|  |  |  |  |



**Gambar 2.5 *Kernel Operator Gaussian* dengan** ** **= 1,4**



(a) (b)

**Gambar 2.6** (a) Sebelum dan (b) Sesudah[1]

**2.2.3** ***Laplace***

Secara formal, fungsi 2-D dari laplacian didefinisikan jumlah dari derivatif kedua:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| *laplace**I*  |  | 2 | *I* |  |  | 2 | *I* |  |
|  |  |  |  |  |
|  |  |  |  |  |  |  |
| *x* | | 2 | *y* | | 2 |  |
|  |  |  |
|  |  |  |  |  |  |  |

(2.4)

Dalam bentuk paling sederhana, menggunakan kernel (3x3) berikut ini:

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | 0 | 1 | 0 |  |  | 1 | 1 | 1 |  |
| *H* | 1 |  1 |  4 | 0 | *H* | 2 |  1 | 8 | 1 |  |
|  |  |  |  |  |  |  |  |  |
|  |  | 0 | 1 | 0 |  |  | 1 | 1 | 1 |  |

dalam didekati dengan

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  | 1 |  2 | 1 |  |
| *H* | 3 |   2 | 4 |  2 |  |
|  |  |  |  |  |
|  |  | 1 |  2 | 1 |  |

10



**Gambar 2.7** Efek Laplace Filter yang kedua (-1,8,-1)[12]

*Laplacian filtering* digunakan untuk mencari pinggir atau *outline* dari gambar dalam citra digital. Pada contoh dibawah garis mata, pinggiran baju, bentuk kepala kelihatan jelas. Gambar sebelah kanan adalah hasil laplace filter yang dinegatifkan atau *invert*, disitu terlihat seperti kita menggambar *outline* sebuah foto.

**2.3 Pengenalan Pola Citra Digital**

Pengenalan pola mengelompokkan data numerik dan simbolik (termasuk citra) secara otomatis oleh mesin (dalam hal ini komputer). Tujuan pengelompokan adalah untuk mengenali suatu objek di dalam citra. Manusia bisa mengenali objek yang dilihatnyaa karena otak manusia telah belajar mengklasifikasikan objek-objek di alam sehingga mampu membedakan suatu objek dengan objek lainnya. Kemampuan sistem visual manusia inilah yang dicoba ditiru oleh mesin. Komputer menerima masukan berupa citra objek yang akan diidentifikasi, memproses citra tersebut, dan memberikan keluaran berupa deskripsi objek didalam citra.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
| citra | Citra |  | deskripsi |  |
|  |  | objek |  |
|  |  |  |  |
|  |  |  |  |  |



**Gambar 2.8** Blok sistem pengenalan pola citra digital

Contoh pengenalan pola misalnya citra pada Gambar 2.8 adalah tulisan tangan yang digunakan sebagai data masukan untuk mengenali karakter ‘A’. Dengan menggunakan suatu alogaritma pengenalan pola,

11

diharapkan komputer dapat mengenali bahwa karakter tersebut adalah ‘A’.



**Gambar 2.9** Citra karakter ‘A’ yang digunakan sebagai masukan untukpengenalan huruf

**2.3.1** ***Hough Transform***

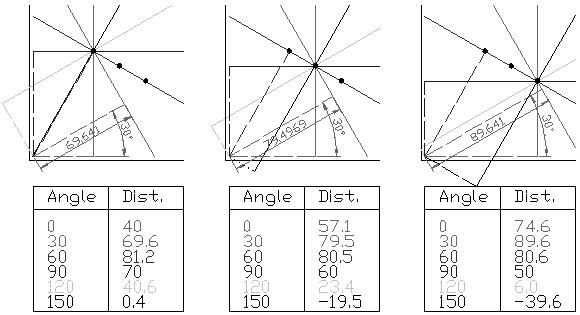
Transformasi Hough memungkinkan kita untuk menemukan (menentukan) bentuk berbagai obyek di dalam citra dengan memanfaatkan tepi-tepi obyek tersebut. Setelah melakukan pendeteksian tepi, dilanjutkan dengan pengoperasian thresholding pada citra maka hanya pixel-pixel yang signifikan saja yang ditampilakn menjadi citra hasil.

Suatu garis lurus merupakan barisan keseluruhan pixel-pixel yang membentuk diekspresikan dengan persamaan:

|  |  |
| --- | --- |
| f(x) = y = m.x + c | (2.5) |

Dan jika kita evaluasi nilai-nilai untuk m dan c, agar diperoleh suatu garis lurus yang diinginkan.

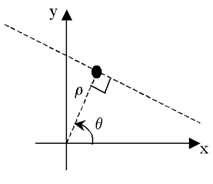
12



**Gambar 2. 10** Representasi Hough Transform[4]

Ada beberapa permasalahan yang timbul dalam penggunaan rumus diatas. Permasalahan yang timbul apabila garis yang direpresentasikan hampir tegak lurus terhadap sumbu x, sehingga nilai m akan menjadi negatif tak terhingga atau positif tak terhingga. Oleh karena itu, representasi garis tidak dalam koordinat kartesian (x,y) tetapi menjadi koordinat polar (r,Ɵ). Rumus persamaan garis dalam koordinat polar adalah sebagai berikut:

|  |  |
| --- | --- |
| x. cos Ɵ + y. sinƟ = r | (2.6) |



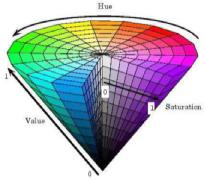
**Gambar 2.11** Hough Transform dalam Koordinat Polar[4]

**2.4 Skala warna HSV**

Skala warna HSV merupakan bentuk lain dalam merepresentasikan data pixel dari suatu image yang berwarna selain skala warna RGB.

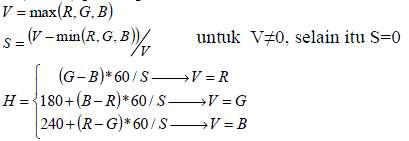
13

HSV merupakan singkatan dari Hue Saturation Value. Hue adalah nilai yang merepresentasikan warna mulai dari merah, kuning, hijau, cyan, biru, magenta dan kembali ke merah. Kisaran nilai Hue antara 0-360. Saturation adalah nilai yang merepresentasikan derajat keabuan dari suatu warna, mulai dari unsaturated sampai fully saturated. Semakin besar prosentase nilai saturation maka semakin besar derajat keabu-abuannya. Kisaran nilai saturation antara 0% - 100%. Value menyatakan tingkat kecerahan suatu warna. Kisaran nilai value antara 0%-100%. Semakin besar nilai value maka semakin terang warna pixel tersebut.



**Gambar 2.12** Gambar*cone*yang merepresentasikan nilai skala warnaHSV[1]

Berikut ini adalah penghitungan nilai H (*Hue*), S (*Saturation*), V (*Value*) pada OpenCV.



(2.7)

14

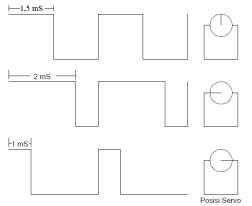
jika H<0 maka H=H+360

Dengan skala output 0≤V≤1, 0≤S≤1, 0≤H≤360. Kemudian nilai tersebut dikonversikan pada nilai HSV yang disediakan oleh OpenCV sesuai tipe data.

**2.5 Motor Servo**

Berbeda dengan motor DC dan motor Stepper, motor servo adalah motor dengan system closed feedback dimana poisi dari motor akan diinformasikan kembali ke rangkaian control yang ada di dalam motor servo. Motor ini terdiri dari sebuah motor, serangkaian gear, potensiometer dan rangkaian control.

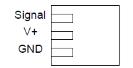
Potensiometer berfungsi untuk menentukan batas sudut dari putaran servo. Sedangkan sudut dari sumbu motor servo diatur berdasarkan lebar pulsa yang dikirim melalui kaki sinyal dari kabel motor. Tampak pada gambar dengan pulsa 1.5 mS sumbu motor akan berada pada posisi tengah. Semakin lebar pulsa OFF maka akan semakin besar gerakan sumbu kearah jarum jam dan semakin kecil pulsa OFF maka akan semakin besar gerakan sumbu ke arah yang berlawanan dengan jarum jam.



**Gambar 2.13** Teknik PWM untuk mengatur servo motor[14]

Motor servo biasanya hanya bergerak mencapai sudut tertentu saja dan tidak kontiyu seperti motor DC maupun motor stepper. Walaupun demikian, untuk beberapa keperluan tertentu, motor servo dapat dimodifikasi agar bergerak kontinyu.

15



**Gambar 2.14** Pin Out kabel Motor Servo[14]

Pada robot, motor ini sering digunakan untuk bagian kaki, lengan atau bagian-bagian lain yang mempunyai gerakan terbatas dan membutuhkan torsi cukup besar.

**2.6 Konfigurasi Rubik**

a. Bagian-bagian rubik.

* Bagian tepi

Bagian ini terdiri dari 2 warna, ada 12 bagian tepi yang terletak di baris tengah.



**Gambar 2.15** Bagian tepi[10]

* Bagian sudut

Bagian ini terdiri dari 3 warna, ada 8 bagian sudut yang terletak disekitar sudut



**Gambar 2.16** Bagian sudut[10]

* Bagian tengah

Bagian ini terdiri dari 1 warna, ada 6 bagian tengah yang terletak di tengah tiap sisinya. Bagian tengah tidak dapat bergerak dan mempresentasikan warna setiap sisinya.

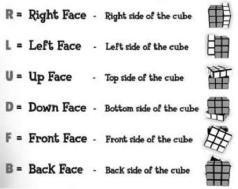
16



**Gambar 2.17** Bagian tengah[10]

b. Langkah pergerakan

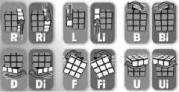
Putar searah atau berlawanan jarum jam maksudnya adalah putar 90 derajat pada arah yang disebutkan



**Gambar 2.18** Setiap langkah dipresentasikan dengan huruf[10]

Huruf dengan awalan “i” maksudnya kebalikan dari pergerakan dari rubik itu sendiri.

17



**Gambar 2.19** Huruf dengan “i” adalahinverse dari pegerakantersebut[10]

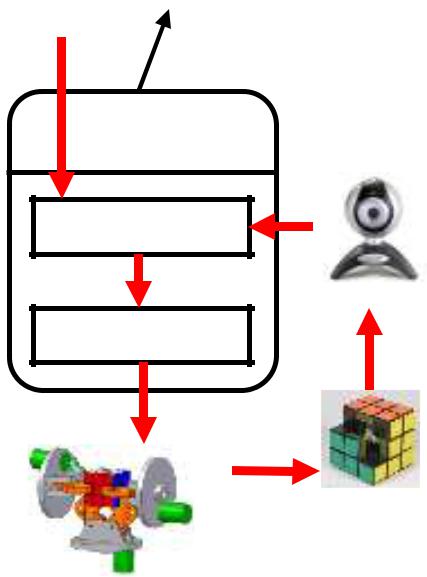
18

**BAB III**

**PERENCANAAN SISTEM**

Pada tugas akhir ini, sistem pendeteksian dan pembacaan warna pada rubik dirancang dengan menggunakan komputer sebagai pusat pengolah data citra dan aktuator robot, robot manipulator sebagai penggerak rubik untuk diputar dan kamera digital sebagai Akusisi citra pada rubik. Diagram blok sistem ditunjukkan pada gambar 3.1.

*Input* posisi



warna Rubik

**PC** Kamera

Computer Vision

Arduino

Robot

Manipulator *Real*

Rubik

**Gambar 3.1** Diagram Blok Sistem

Cara kerja keseluruhan sistem adalah:.

1. Kamera digital sebagai sensor visual menangkap citra dari rubik dan menampilkannya di komputer. Setelah komputer menerima citra, citra tersebut akan diolah dengan menggunakan *image processing*.
2. *Software* akan memulai pendeteksian sisi-sisi tepi rubik yangberwarna hitam dan mempunyai pola persegi. Setelah *software* dapat mengenali rubik, selanjutnya dilakukan metode *tracking* pada rubik untuk diambil warnanya pada sisi yang dideteksi.

19

1. Setelah selesai pengambilan warna pada rubik, selanjutnya komputer akan mengirim data ke minimum sistem Arduino melalui komunikasi serial USB untuk mengarahkan robot manipulator untuk mengganti sisi rubik yang berikutnya.
2. Setelah 6 sisi tiap rubik selesai diambil warnanya (1 sisi = 9 data warna) maka selanjutnya data akan disimpan ke dalam *memory* komputer dan akan dicari *solving algorithm* rubikuntuk menyelesaikan rubik tersebut.

**3.1 Kamera Sebagai Sensor Visual**

Modul kamera yang digunakan sebagai sensor visual dipilih kamera digital jenis *web camera* bermerk Logitech tipe Webcam C500 serta memiliki fitur sebagai berikut :

1. *Image sensor*: *True* 1.3-*megapixel sensor*

|  |  |  |
| --- | --- | --- |
| b. | *Resolution* | : Di atas 1280 x 1024 *pixels* |
| c. | *Frame rate* | : Di atas 30 *frame per second* |
| d. | *Interface* | : Hi-speed USB |



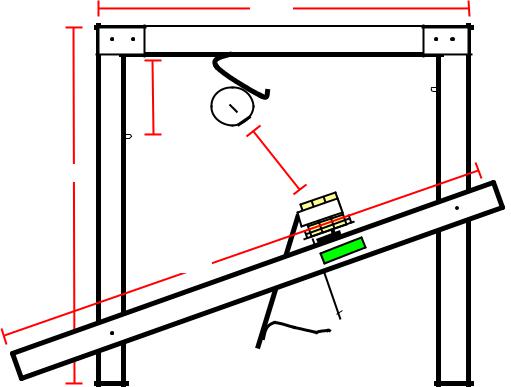
**Gambar 3.2** Logitech Webcam C500

Adapun hal yang terpenting dalam pemilihan sensor visual pada sistem ini yaitu menggunakan kamera yang memiliki *image sensor* 1,3 *megapixel* agar diperoleh gambar yang tajam, serta memiliki *frame rate* yang tinggi*.* Hal tersebut dilakukan untuk membantu proses pengolahan

20

citra agar citra yang dikirim ke komputer memiliki kualitas yang lebih presisi dan lebih cepat.

|  |  |  |  |
| --- | --- | --- | --- |
|  | 40 cm |  |  |
|  | Kamera | 2 cm |  |
|  |  |  |
| 7,5 cm |  | LED |  |
|  |  |  |
| LED | 12 cm |  |  |
| 50 cm |  |  |  |
|  |  | Rubik |  |
| 63 cm |  |  |  |



**Gambar 3. 3** Posisi kamera sebagai sensor visual terhadap rubik

Pada perancangan ini, kamera sebagai sensor visual diatur posisinya terhadap rubik. Dimana jarak antara lensa kamera dengan rubik dapat dengan jarak minimum sebesar 12 cm serta posisi sudut tangkap kamera diatur sebesar 60° terhadap garis normal horisontal. Selain itu ditambahkan tiga buah LED (*Light Emitting Dioda)* yang masing-masing diletakan di sebelah kanan dan kiri kamera. Fungsi dari ketiga lampu tersebut adalah untuk mengatur intensitas cahaya sekitar kamera dan rubik*.*

Selain perancangan posisi untuk kamera sebagai sensor visual, kamera tersebut juga dilakukan perancangan untuk pengaturan resolusi citra yang akan ditampilkan pada monitor. Kamera sebagai sensor visual ini diatur dengan resolusi citra output sebesar 320 x 240.

**3.2 Central Processing Unit**

CPU *(Central Processing Unit*) sebagai perangkat yang bertugas untuk mengolah data. Kecepatan dalam pengolahan data ditentukan oleh spesifikasi CPU. Pada sistem robot rubik ini digunakan CPU dengan spesifikasi sebagai berikut :

1. Intel Core 2 Duo *processor* T6600(2.2 GHz, 800MHz FSB).
2. 2 GB *Memory*
3. *Video Graphic* 512 MB

21

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



**Gambar 3.4** Laptop Aspire 4736G

**3.3 Perancangan Perangkat Keras**

Secara garis besar perancangan perangkat keras merupakan pembuatan robot rubik. Robot ini dirancang dengan spesifikasi sebagai berikut :

1. Robot rubik ini menggunakan dua motor servo. Motor I digunakan untuk memutar rubik dengan arah horizontal dan motor II digunakan untuk memutar rubik dengah arah vertikal serta menahan rubik (bagian atas dan tengah) pada saat rubik diputar secara horizontal.
2. Robot rubik ini dirancang dengan menggunakan minimum sistem Arduino. Arduino adalah pengendali mikro single-board yang bersifat open-source.
3. Robot ini dilengkapi dengan LCD *(Liquid Crystal Display*) yang mempunyai 8-bit kontrol dan 2x16 *display* baris. LCD ini digunakan untuk menampilkan serial komunikasi pada Arduino dan CPU.

22



**Gambar 3.5** Perangkat keras berupa dua motor servo, minimum sistemArduino dan LCD.

**3.3.1** **Pengontrolan Motor Servo Horizontal**

Sistem ini digunakan untuk mengontrol motor servo standar yang dapat diatur posisinya antara 0o sampai dengan 180o yang dipasang pada bagian bawah rubik. Motor servo ini digunakan untuk memutar sisi rubik ke arah kiri atau sebaliknya dengan sudut 90o.

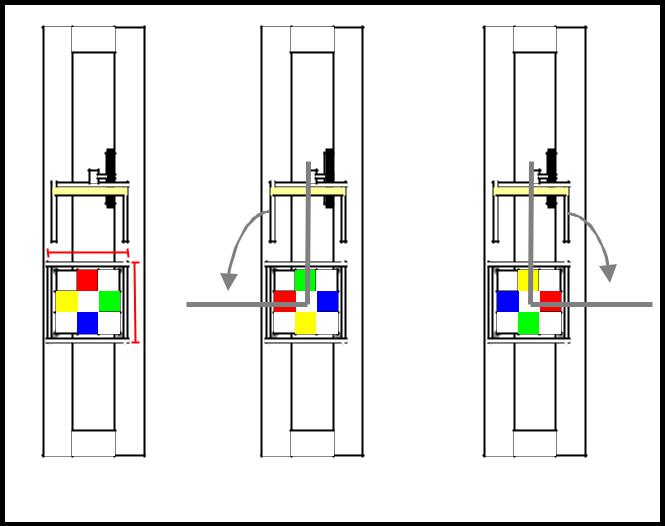
Untuk pengotrolan motor servo yang bergerak 90o, prosesor kedua (Arduino) mengirimkan data/sinyal pulsa (kontinyu) yang lebarnya 1.5 ms selama interval waktu tertentu sehingga sudut yang dicapai motor servo sebesar 90o, pada posisi ini keadaan posisi rubik dalam normal/middle. Setelah itu prosesor kedua akan menunggu data yang dikirim oleh CPU untuk mengetahui posisi mana selanjutnya akan digerakkan (kekanan atau kekiri). Motor servo akan bergerak kekanan bila prosesor utama akan mengirim pulsa 1 ms (sebesar 0o) dan servo akan bergerak kekiri bila diberikan pulsa yang lebarnya 2 ms.

23

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| CPU |  | Adruino |  | Motor Servo |
|  |  |  |  |  |



**Gambar 3.6** Blok diagram sistem pergerakan motor servo



8 cm

8 cm

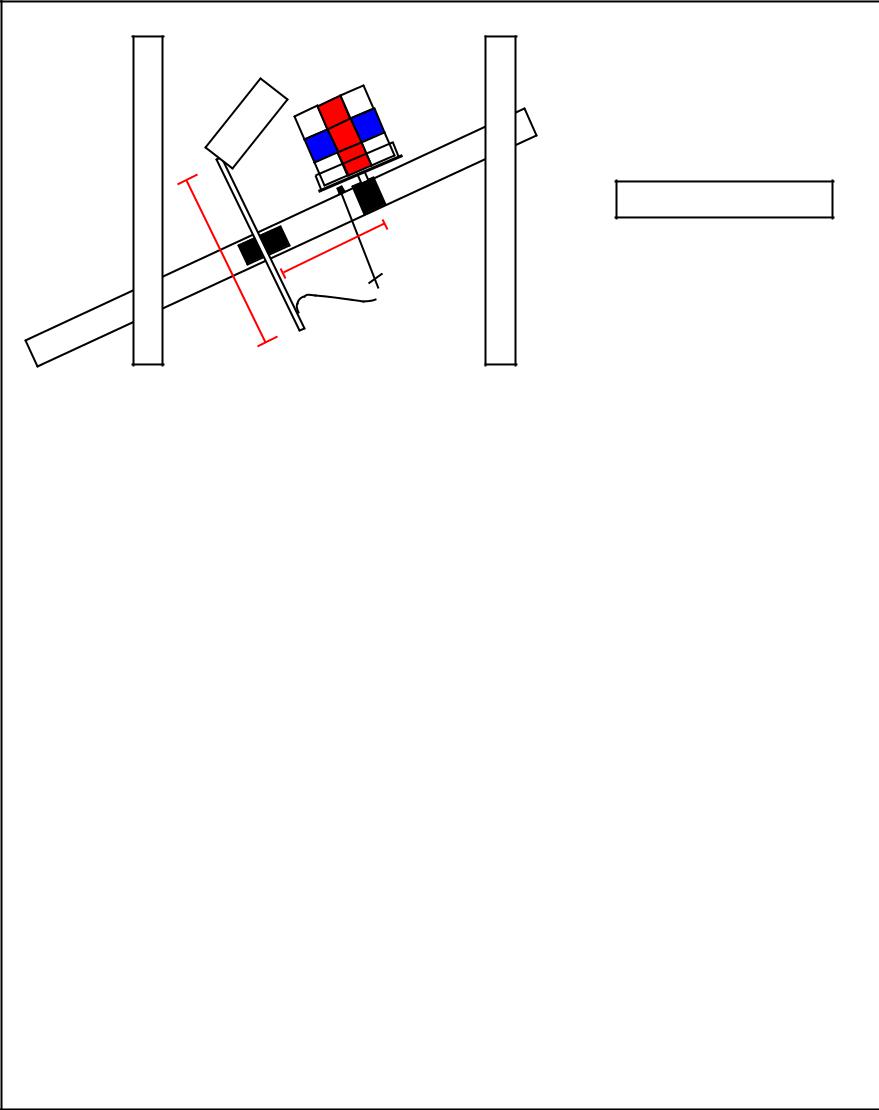
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Pada saat lebar |  | Pada saat lebar |  | Pada saat lebar |
| pulsa 1,5 ms (90o) |  | pulsa 1 ms (0o) |  | pulsa 2 ms (180o) |

**Gambar 3.7** Servo motor pada saat dalam posisi normal, kekiri dankekanan.

**3.3.2** **Pengontrolan Motor Servo Vertikal**

Sistem ini digunakan untuk mengontrol motor servo secara horizontal. Motor ini digunakan untuk memutar sisi rubik secara horizontal sebesar 90o dan menahan rubik pada bagian atas dan tengah. Untuk posisi/keadaan normal motor servo mendapatkan sinyal sebesar 1.7 ms (140o) selama interval waktu tertentu. Untuk pengontrolan motor servo memutar sisi rubik secara vertikal, motor servo bergerak pada posisi 70o dan balik lagi ke posisi normal selama interval waktu tertentu. Pada posisi menahan rubik, motor servo akan bergerak pada posisi 180o (2 ms). Proses ini akan terus berlangsung sesuai dengan kebutuhan robot.

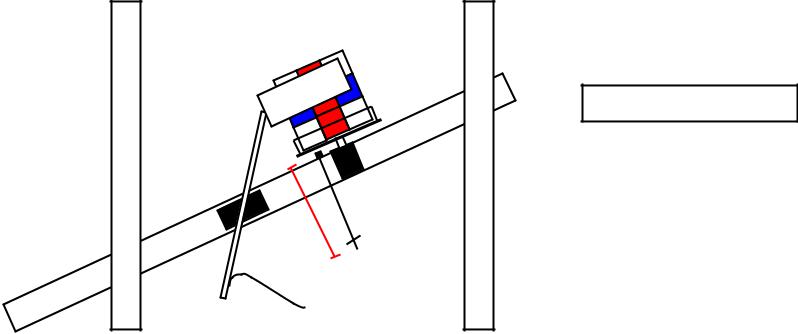
24



Normal

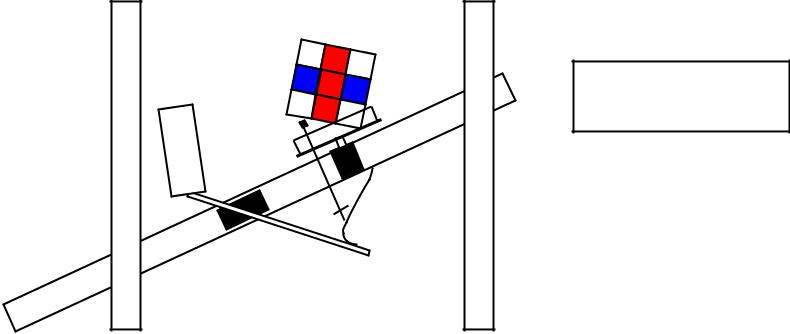
11,5 cm

22 cm



Menggenggam

12 cm



Memutar

vertikal

**Gambar 3.8** Pengontrolan motor servo vertikal

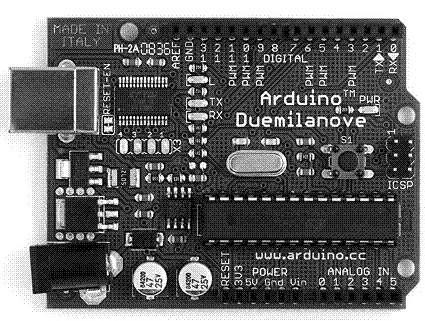
**3.3.3** **Minimum Sistem Arduino Duemilanove**

Arduino board adalah modul yang menggunakan mikrokontroler AVR dan menggunakan seri yang lebih canggih, sehingga dapat digunakan untuk membangun sistem elektronika

25

berukuran minimalis namun handal dan cepat. Berbagai modul dan sensor terkini dapat dipasang pada board ini dilengkapi dengan berbagai kode demo yang memuaskan.

Arduino terdiri dari beberapa board, yang dapat digunakan sesuai kebutuhan dan menggunakan software open source yang dapat dijalankan pada Windows, Mac dan Linux. Beberapa board yang terkenal ditampilkan pada gambar di bawah :



**Gambar 3.9** Arduino Main Board berbasis ATmega328 denganbootloader[14]

**Spesifikasi :**

 Microcontroller : Atmega 328

 Operating Voltage : 5V

* Input Voltage (recommended) : 7-12V

|  |  |  |
| --- | --- | --- |
|  | Input Voltage (limits) | : 6-20V |
|  | Digital I/O Pins | : 14 (yang terdiri dari 6 |
|  |  | PWM output) |
|  | Analog Input Pins | : 6 |
|  DC Current per I/O Pin | | : 40 mA |
|  | DC Current for 3.3V Pin | : 50 mA |
|  | Flash Memory | : 32 KB |
|  | SRAM | : 2 KB |

26

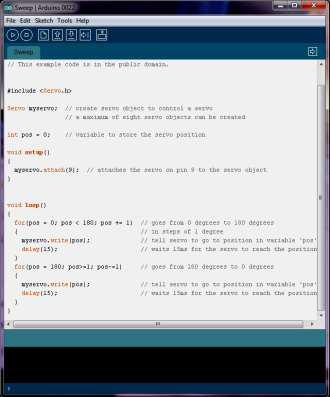
 EEPROM : 1 KB

 Clock Speed : 16 MHz

Mikrokontroler Arduino ini menggunakan Arduino

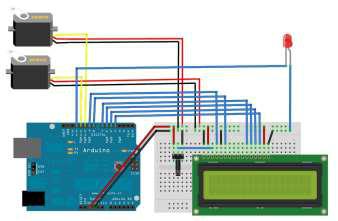
programming language berbasiskan Wiring dan Arduino

development environment berbasiskan Processing. Arduino menggunakan koneksi USB (*Universal Serial Bus*) menggunakan chip FTDI (*Future Technology Devices International*) untuk melakukan pemrograman, dan biasanya pada chip Arduino sudah dimasukkan bootloader, sehingga dapat dilakukan pemrograman langsung ke dalam chip menggunakan software Arduino.



**Gambar 3.10** Contoh program Arduino menggunakan Java.

27



**Gambar 3.11** Breadboard rangkaian robot rubik menggunakan Arduino

**3.3.4** **Liquid Crystal Display (LCD)**

Ini contoh sketsa display “Hello World” di LCD dan memperlihatkan waktu dalam detik sejak Arduino di reset.



**Gambar 3.12** Hello world di 2x16 LCD[14]

LCD mempunyai parallel interface, maksudnya mikrokontroler mempunyai beberapa pin yang dapat mengatur display.

1. **RS (*Register select*) pin** : mengatur didalam memory LCDyang akan kita masukkan data. Kita dapat memilih data register, memegang apa yang akan terjadi di layar, atau instruksi register, yang mana kontroler LCD mencari instruksi apa yang akan dilanjutkan berikutnya.
2. ***Read/Write* (R/W) pin**: memilih reading mode atau writingmode.
3. ***Enable* pin**: mengaktifkan writing ke register.

28

1. **8 data pins (D0 – D7) :** isi dari pins adalah bits(*high*atau*low*)yang mengisi ke register ketika *mode writing*, atau nilai-nilai yang kita baca ketika *mode reading*.

Ada beberapa ***display contrast*** **pin (Vo),** ***power supply*** **pins** **(+5V and Gnd) dan LED *Backlight* (Bklt+ and Bklt-) pins** yangdapat kita gunakan untuk power di LCD, mengontrol kontras display dan menyalakan atau mematikan lampu *background*.

Proses pengontrolan display melibatkan memasukan data yang membentuk gambar dari apa yang kita inginkan ke dalam register data, kemudian menempatkan instruksi kedalam intruksi register. *LiquidCrystal Library* mempermudahkan ini untuk kita maka kitatidak perlu lagi mengetahui intruksi tingkat rendah.

**Tabel 3.1** Perintah library LCD pada Arduino

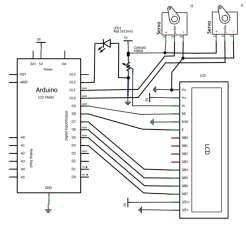
|  |  |  |
| --- | --- | --- |
|  | LiguidCrystal Library | |
| LiquidCrystal() |  | Display() |
| Begin() |  | Nodisplay() |
| Clear() |  | scrollDisplayLeft() |
| Home() |  | scrollDsiplayRight() |
| setCursor() |  | autoscroll() |
| write() |  | noAutoscroll() |
| print() |  | leftToRight() |
| cursor |  | rightToLeft() |
| noCursor() |  | createChar() |
| blink() |  | noBlink() |
| Di schematic pin yang saya gunakan: | | |
|  LCD RS pin | : digital pin 9, | |

* LCD Enable pin : digital pin 8,

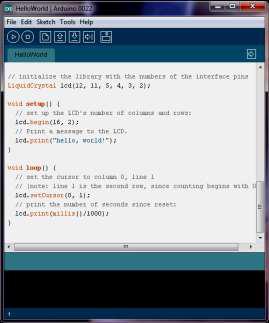
|  |  |  |
| --- | --- | --- |
|  | LCD D4 pin | : digital pin 7, |
|  | LCD D5 pin | : digital pin 6, |
|  | LCD D6 pin | : digital pin 5, |
|  | LCD D7 pin | : digital pin 4, |
|  | LCD R/W pin | : ground, |

* 100K resistor,
* +5V dan ground.

29



**Gambar 3.13** Schematic robot rubik menggunakan Fritzing



**Gambar 3.14** Contoh program LCD di Arduino

30

**3.4 Pengolahan dan Pengenalan Pola Citra Digital**

Pada pendeteksi rubik, citra akan diambil gambarnya dan diolah secara real time, yang akan kita bahas disini menggunakan bahasa Visual C++ dan OpenCV library. Untuk meningkatkan efisiensi, kita mengubah ukuran frame menjadi 320x240 pixels. Standarisasi ukuran gambar adalah ide baik yang tidak hanya untuk efesiensi saja, tetapi juga karena parameter yang kita harus miliki agar dapat bekerja lebih cepat.



**Gambar 3.15** Citra yang diambil dengan kamera

Setelah gagal menemukan titik detektor yang baik untuk tugas akhir ini, keputusan dibuat untuk melanjutkan dengan pendeketan berbasis tepi. Hal ini menunjukan bahwa pendekatan seharusnya bekerja dengan sangat baik karena tepi-tepi garis sejajar pada rubik adalah fitur yang tepat untuk mendeteksi rubik. Tahap pertama dari alogaritma ini adalah untuk mengekstrak tepi dari gambar yang dapat dilakukan dengan menerapkan filter Laplacian. Kemudian menerapkan Hough Transform probabilistik untuk mencari sisi-sisi panjang. Hough Transform mengambil parameter threshold yang menentukan kepekaannya untuk mendeteksi sebuah garis. Untuk menghindari pegambilan garis yang melebihi batas skala untuk setiap pergantian intensitas cahaya, maka saya memberi jumlah segmen tepi yang dideteksi dari gambar kurang lebih 50 threshold setiap iterasinya.

31

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | Detection | | |  | Re-init | |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  | Memory / Learn |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
| Preprocessing |  |  | update |  |  |  | Output |  |
|  |  |  |  |  |  |  |  |  |  |  |
|  |  | Tracking | | |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |
|  |  | **Gambar 3.16** Design dan implementasi | | | | | | |  |  |



Int THR = 50;

if (tracking == 0){

…

if (lastdetected>dects) THR= THR+1;

if (lastdetected<dects) THR= max(2,(THR-1));

li = cvHoughLines2(d2, storage,

CV\_HOUGH\_PROBABILISTIC, 1, 3.1415926/45, THR, 10,

5);

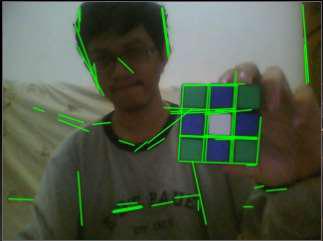
}



**Gambar 3.17** Setelah mengalami Lapacian Filter

32

Ketika Rubik berada dalam citra, algoritma akan menemukan banyak nya tepi-tepi sepanjang garis yang berada di sekitar stiker rubik. Dalam prakteknya tepi garis ini yang ditemukan, ternyata sangat banyak sekali sehingga threshold di Hough Transform akan menjadi sangat besar dan latar belakang yang cenderung memiliki banyak noise.

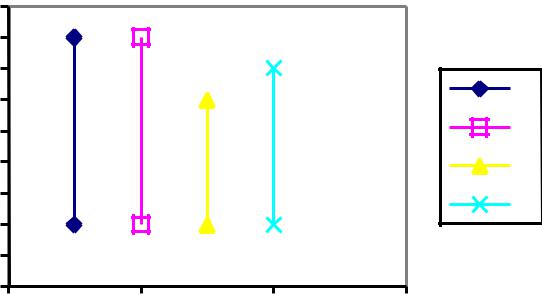


**Gambar 3.18** Tepi-tepi yang terdeteksi Hough Transform setelahmenggunakan Adaptive Threshold

Tujuan langkah selanjutnya adalah mengidentifikasi empat sudut yang menggelilingi permukaan-permukaan kubus. Dengan melihat posisi awal dan akhir dari semua garis, kita akan mencari dua garis yang saling terhubung atau saling berpotongan yang memiliki panjang yang hampir sama. Karena kita mencari sudut kubus dalam langkah ini, maka harus melakukan sedikit beberapa rumus vektor matematika dalam kasus yang terakhir dan mencari posisi sudut yang terdekat. Sepasang garis yang telah terseleksi harus mempunyai panjang kurang lebih 30% yang sama bila tidak dibuang. kita juga akan membuang sepasang garis jika dari dua sudut antara mereka yang lebih rendah dari 0.5 radians (atau kira-kira 30derajat), karena ini menunjukan garis-garis yang hampir sejajar.

33

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **4,5** |  |  |  |  |
| **4** |  |  |  |  |
| **3,5** |  |  | **p** |  |
| **3** |  |  |  |
|  |  |  |  |
| **2,5** |  |  | **q** |  |
| **2** |  |  | **r** |  |
| **1,5** |  |  |  |
|  |  | **t** |  |
| **1** |  |  |  |
|  |  |  |  |
| **0,5** |  |  |  |  |
| **0** |  |  |  |  |
| **0** | **2** | **4** | **6** |  |



**Grafik 3.1** Empat garis vertikal yang memiliki jarak berbeda-beda

Misalkan terdapat empat garis p, q, r dan t. Dimana masing-masing garis mempunyai titik:

* p0(1, 1) dan p1(1, 4)
* q0(2, 1) dan q1(2, 4)
* r0(3, 1) dan r1(3, 3)
* t0(4, 1) dan t1(4, 3.5)

Mencari jarak antara garis :



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *d* |  | 2 | 2 |  |
| *p*1.*x*  *p*0.*x* | *p*1.*y*  *p*0.*y* |  |
| 1 |  |  |  |  |

*d*2 *q*1.*x*  *q*0.*x*2  *q*1.*y*  *q*0.*y*2



|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| *d* |  |  | 2 | 2 |  |
| 3 | *r*1.*x*  *r*0.*x* | *r*1.*y*  *r*0.*y* |  |
|  |  |  |  |  |
| *d* |  |  | 2 | 2 |  |
| 4 | *t*1.*x*  *t*0.*x* | *t*1.*y*  *t*0.*y* |  |
|  |  |  |  |  |



(3.1)

Dengan menggunakan rumus 3.2 kita dapat menseleksi garis yang kita inginkan:

max(*d*1, *dx*) / min( *d*1, *dx*)  1,3

dimana

*x*



2, 3 dan 4

(3.2)

Dengan

|  |  |  |  |
| --- | --- | --- | --- |
| *d* | 4 |  2,5 |  |
|  |  |  |

hasil jarak pada masing-masing garis *d*

1

, maka dapat kita simpulkan bahwa

 *d*1

3 , , *d*

2

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| *d*2 |  3 | | , *d*3  2 dan | |  |
| dan *d* | | 4 | | mempunyai |  |
|  |  |  |  |

34

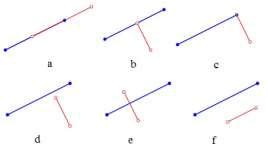
jarak lebih kecil dari 1,3 dan

|  |  |  |
| --- | --- | --- |
| *d* | 3 |  |
|  |  |

mempunyai jarak sebaliknya yang harus

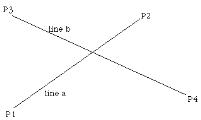
kita buang.

Proses selanjutnya kita mencari dua garis yang saling berpotongan untuk diseleksi (filter) lagi. Pada tahap dua garis yang saling berpotongan akan kita ambil untuk diproses lagi dan garis yang tidak saling berpotongan akan kita buang.



**Gambar 3.19** Contoh dua garis yang saling berpotongan dan tidakberpotongan[2]

Terlihat pada gambar 3.7 bahwa a, b, c dan e mempunyai garis yang saling berpotongan sedangkan d dan f tidak saling berpotongan. Misalkan kita mempunyai dua garis yang saling berpotongan dimana garis Pa mempunyai titik P1 dan P2 dan garis Pb mempunyai titik P3 dan P4.



**Gambar 3.8** Dua garis yang saling berpotongan pada sebuah titik[2]

Persamaan garisnnya adalah

*y*  *mx*  *b*

*Pa*  *P*1 *ua* (*P*2 *P*1)

*Pb*  *P*3 *ub* (*P*4 *P*3)

35

Dimana syarat garis yang saling berpotongan **Pa = Pb** maka dapat kita peroleh

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | *x*1 *u* | | *a* | | (*x*2  *x*1)  *x*3  *u* | *b* | | (*x*4  *x*3) | | (3.3) |  |
|  |  |  |  |  |  |  |  |
|  |  | *y*1 *u* | | | *a* | ( *y*2  *y*1)  *y*3  *u* | | *b* | | ( *y*4  *y*3) |  |
|  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
| Sehingga didapatkan penyelesaian untuk persamaan ua dan ub | | | | | | | | | | |  |  |
| *u* |  |  | *x*4 *x*3*y*1 *y*3*y*4 *y*3*x*1 *x*3 | | | | | | | |  |  |
|  |  | |  |  *y*3*x*2 *x*1*x*4 *x*3*y*2 *y*1 | | | | | |  |  |
|  | *a* |  | *y*4 |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |
| *u* |  |  | *x*2 *x*1*y*1 *y*3*y*2 *y*1*x*1 *x*3 | | | | | | | | (3.4) |  |
| *b* | *y*4 *y*3*x*2 *x*1*x*4 *x*3*y*2 *y*1 | | | | | | | |  |
|  |  |  |
|  |  |  |  |  |



Jadi titik (x, y) yang saling berpotongan adalah

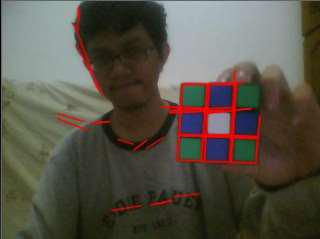
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *x*  *x*1 *u* | *a* | | (*x*2  *x*1) |  |
|  |  |  |
| *y*  *y*1 *u* | | *a* | ( *y*2  *y*1) |  |
|  |  |  |  |

(3.5)

Jika dilihat dari persamaan diatas, kita dapat menentukan sebelumnya garis mana yang saling berpotongan dengan melihat bahwa pembagi antara ua dan ub memiliki persamaan yang sama, sehingga untuk den (garis yang saling berpotongan) yang lebih kecil dari 0,1 (bernilai 0) maka garis tersebut tidak saling berpotongan.

*den* *y*4 *y*3*x*2 *x*1*x*4 *x*3*y*2 *y*1

*abs*(*den*)0,1



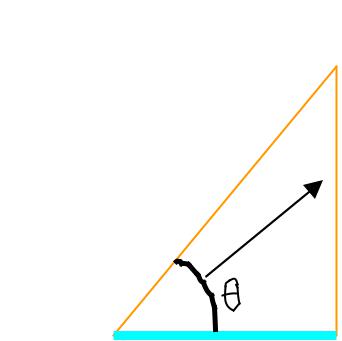
(3.6)

**Gambar 3.20** Hasil citra setelah dicari persamaan garis

36

Langkah selanjutnya adalah mencari sepasang garis yang mempunyai sudut yang lebih rendah 0,5 radians (30 derajat) dengan menggunakan rumus trigonometri.

TRIGONOMETRI



hypotenesa

hadap

|  |  |  |
| --- | --- | --- |
| arctan  | *hadap* |  |
| *dekat* |  |
|  |  |



dekat

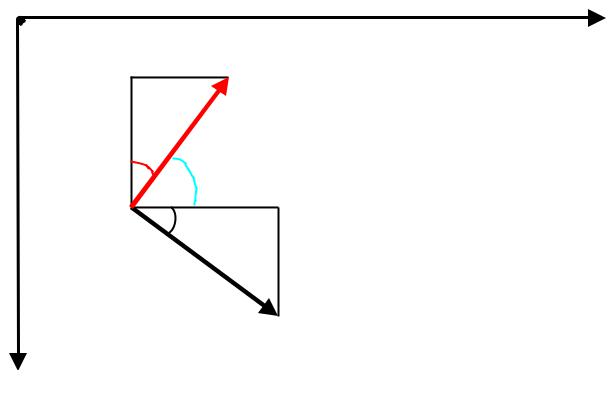
**Gambar 3.21** Trigonometri

*a*1arctan*p*1.*y*  *p*0.*y*, *p*1.*x*  *p*0.*x* *a*2arctan(*q*2.*y*  *q*0.*y*, *q*1.*x*  *q*0.*x*)

Jika, a1< 0 maka a1 = a1+ **

a2< 0 maka a2 = a2+ **

(0, 0) pixel



Px

garis a2

30o

60o

30o

garis a1

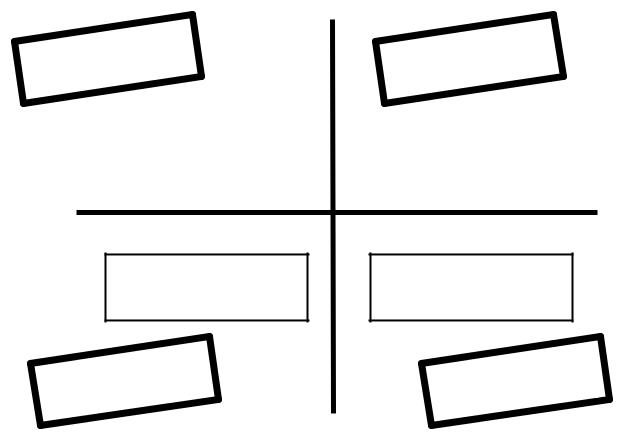
(3.7)

Py

**Grafik 3.2** Dua garis yang membentuk sudut dalam pixel

37

Setelah kita mendapatkan dua sudut antara garis yang saling berpotongan seperti yang terlihat pada grafik 3.12 maka kita bisa menemukan sudut yang terbentuk antara dua garis tersebut.



|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Tan (+) | |  |  |  | Cos (+) |  |
|  |  |  |  |  |  |  |
|  | Kuadran III | |  | Kuadran IV | | |
|  | 180 +** | |  |  | 360-** | |

Kuadran II

180 - **

Kuadran I

**

Sin (+) Semua (+)

**Gambar 3.22** Kuadran dalam grafik pixel

Dengan melihat kuadran pada masing-masing garis dimana garis a1 terletak pada kuadran I dan garis a2 terletak pada kuadran II.

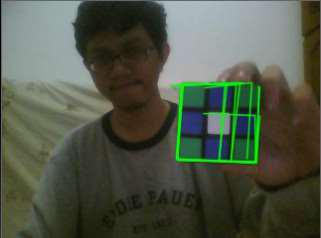
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Maka, *sudut*  *abs**abs**a*2 | |  *a*1 | ** |  | (3.8) |  |
|  |  |  | 2 |  |  |  |
| *sudut*  *abs**abs*303090 | | | | |  |  |
| *sudut* 30 | *o* |  |  |  |  |  |
|  |  |  |  |  |  |



**Gambar 3.23** Hasil citra setelah dicari sudut yang tegak lurus

38

Banyak parameter diatas sudah berada di posisi yang benar tetapi yang kita butuhkan hanya sudut yang saling tegak lurus, berpotongan dan mempunyai panjang yang sama.



**Gambar 3.24** Hasil seleksi garis setelah dicari garis dan sudut

Pada langkah berikutnya kita mencari setiap sudut dan menyelesaikannya dengan affine transformation matrix 3x3 yaitu mengambil titik dari citra yang terdeteksi dalam sistem kordinat untuk didefiniskan menjadi sudut dan dua baris yang sesuai.

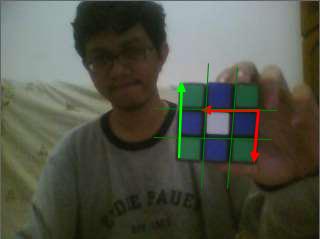
|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| *x*' | | |  | *x*1 | | *x*2 | *h**x* | | |  |  |
|  |  |  |  |  | *y*1 | *y*2 | *k* |  | *y* |  |  |
|  | *y*' | |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  | 1 |  |  |  | 0 | 0 | 11 | | |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |

(3.9)

Kemudian kami menginterasi semua segmen garis yang ditemukan menggunakan hough transform dan menempatkan titik-titik akhir tersebut ke dalam koordinat lokal. Terakhir kita menghitung jumlah segmen setiap garis yang kedua titik akhir terletak pada tepat pada grid. Artinya, dalam sistem koordinat lokal, baik yang koordinat pertama atau kedua tepat 1/3 atau 2/3 dalam margin 0,05 dan koordinat tidak berada di luar jarak[-0.1,1.1]. Jika kedua titik terletak di grid dan sudut dari segmen garis deket dengan sudut sumbu maka segmen ini dihitung sebagai milik grid. Perhatikan kita tidak mencari pembuktian di sepanjang garis kubus (yang cocok dengan salah satu koordinat 0 atau

39

1). Hal ini dikarenakan alogaritma dapat tertipu dalam menemukan ukuran rubik 2x2 atau 3x3. karena ada cukup bukti pada garis luar. Tetapi jika kita hanya mencari bukti didalam garis, maka grid yang lebih kecil akan memilki bukti yang lebih sedikit karena 1/3 atau 2/3 garis yang lolos hampir secara langsung mendekati stiker rubik.



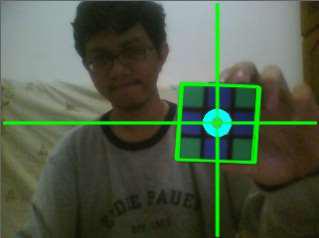
(1,0)

(0,0) 

(0,1)

**Gambar 3.25** Visualiasi dari sistem koordinatdalam grid

Visualisai dari sistem koordinat yang akan kita selesaikan. Kita mencari bukti dari segmen garis yang lain sepanjang 1/3 dan 2/3 di setiap koordinat (diperlihatkan pada garis hijau). Sebuah contoh dari kesalahan 2x2 grid yang kadang-kadang bisa terdeteksi jika kita juga mencari bukti di luar garis seperti yang digambarkan pada garis merah.



UL UR

BL BR

**Gambar 3.26** Posisi rubik terhadap sistem koordinat

40

Selanjutnya mengatur daftar sistem koordinat sesuai dengan nomor dari segmen garis yang sejajar dengan melihat kuadran pada sudut masing-masing titik yang kita peroleh. Terakhir kita iterasi calon-calon titik yang membentuk kotak yang diatas 5% threshold dari jumlah total tepi dan memilih salah satu yang terbaik dari deteksi iterasi yang sebelumnya. Dengan menggunakan sistem koordinat seperti terlihat pada gambar 3.15 kita dapat menemukan secara tepat posisi sembilan stiker dan menampilkan grid ke citra.



**Gambar 3.27** Rubik yang terdeteksi di citra

Setelah menemukan koordinat rubik di citra, proses selanjutnya kita tracking koordinat tersebut sehingga kita tidak perlu selalu mendeteksi posisi rubik terus menerus. Hal ini memudah kita untuk mengambil data secara stabil dan membuat proses kerja lainnya lebih cepat. Metode tracking yang kita gunakan Lucas-Kanade Optical Flow, kita menginisialisasi tracker dengan empat titik dimana kita letakkan di tengah dari 4 garis silang yang kita temukan didalam gird (yang garis merah pada gambar 3.15) . Tracker ini sudah diimplementasikan dalam OpenCV dan tidak hanya sangat cepat tetapi juga sangat akurat. Setelah optical flow tracking sudah diinisialisasi, terjadi perubahan pada grid dideteksi akan mengubah grid menjadi sangat halus yang mengikuti sisi rubik tesebut. Kita juga memeriksa setiap iterasi yang kita tracking tidak

41

mengalami kesalahan dengan terus menerus memeriksa empat titik tersebut harus memenuhi syarat yaitu:

1. Jumlah yang kita tracking harus mempunyai empat titik
2. Jarak antara titik-titik yang kita tracking harus konstan.

Jika dua kondisi ini tidak terpenuhi maka kita akan kembali lagi ke proses deteksi. Karena kita membutuhkan data yang akurat untuk mengambil warna pada rubik tersebut.

**3.5 Pembacaan Warna**

Rubik mempunyai 6 warna yang berbeda-beda mengikuti sisi-sisinya. Warna pada stiker dibaca dengan membagi rata-rata nilai warna yang kita dapat dari daerah kecil di sekitar tengah pada setiap stiker, dengan tujuan untuk mengurangi noise yang sangat menggangu seperti noda-noda kecil menempel di stiker dan intensitas cahaya yang berubah-ubah.

Metode yang kita gunakan tidak susah seperti metode sebelumnya, disini kita hanya mengunakan perbandingan warna dari tiap nilai warna yang kita peroleh. Pertama, Kita menggunakan warna RGB untuk mendeteksi warna utama yaitu merah, hijau, biru dan putih, selanjutnya kita juga menggunakan warna HSV untuk membedakan intensitas cahaya yang terjadi apabila kita mengambil pada saat kondisi gelap atau terang dan saya juga menambahkan untuk menggukan warna Lab (L = *lightness* dan a, b adalah warna yang berlawanan dimensi) sama halnyadengan menggunakan *negative color* pada citra. Karena untuk intensitas cahaya yang cukup tinggi atau kurang dengan menggunakan HSV saja tidaklah cukup untuk membedakan warna yang mempunyai kemiripan yang sama. Contohnya warna kuning bila dalam keadaan terang atau memperoleh intensitas cahaya yang cukup tinggi maka citra yang ditangkap kamera dapat terlihat seperti warna putih, sebaliknya warna orange bila dalam keadaaan gelap maka citra dapat terlihat seperti warna merah. Sama halnya seperti yang terjadi pada warna hijau dan biru.

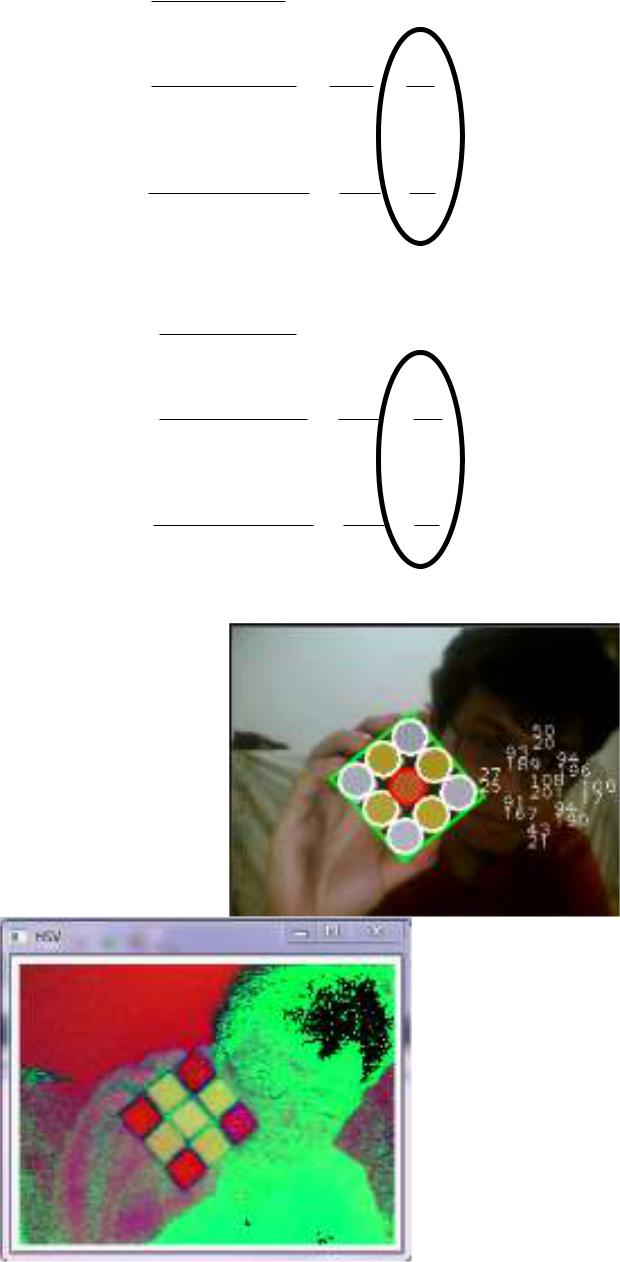
Dua nilai warna RGB dengan intensitas cahaya yang berbeda,

|  |  |
| --- | --- |
| R1 = 50 | R2 = 100 |
| G1 = 70 dan | G2 = 140 |
| B1 = 60 | B2 = 120 |

Untuk warna merah (r1),

42

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| *r* |  |  |  | *r*1 |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
| 1 | |  |  | *r*1 *g*1 *b*1 |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |
| *r* |  |  |  | 50 |  | | 50 |  | | 5 |  |
|  |  |  |  |  |  |
| 1 | |  |  | 50  70  60 |  | 180 | |  | 18 | |  |
| Dan r2, |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |
| *r* |  |  |  | 100 |  |  | 100 |  |  | 5 |  |
|  |  |  |  |  |  |  |  |
| 2 | |  | 100 140 120 | | |  | 360 |  |  | 18 |  |
|  |  |  |  |  |  |  |
| Warna hijau (g1), | | | | |  |  |  |  |  |  |  |
| *g* | |  |  | *g*1 |  |  |  |  |  |  |  |
| 1 | *r*1 *g*1 *b*1 |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |
| *g* | |  |  | 70 |  |  | 70 |  |  | 7 |  |
| 1 | 50  70  60 | | 180 |  | 18 |  |
|  |  |  |  |  |  |  |
| Dan g2, |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |
| *g* |  |  |  | 140 |  |  | 140 |  |  | 7 |  |
| 2 | | 100 140 120 | | 360 | | 18 |  |
|  |  |  |  |  |
|  |  |  |  |  |  |  |

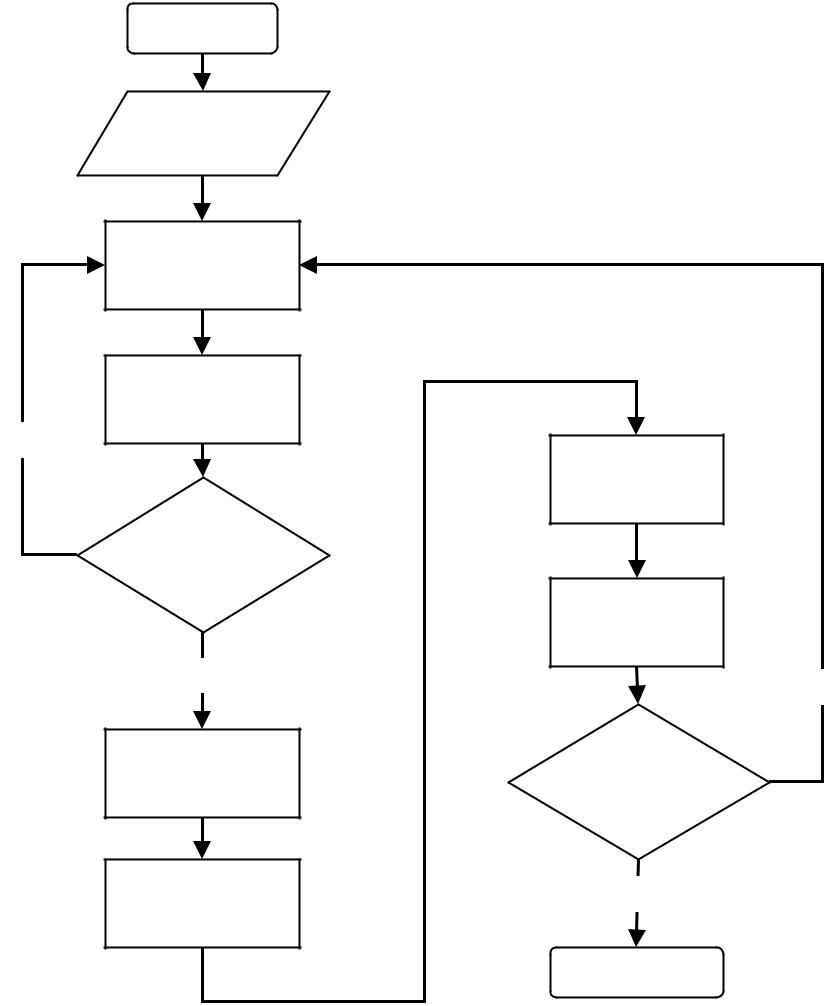


(3.10)



**Gambar 3.28** Bekerja dengan kedalaman warna yang berbeda

43



Start

Query Frame

Preprocessing

Detection

N

Matching

detection ?

Y

Tracking

Learn

Scaning

Send to Data

Serial

N

Push the exit

button ?

Y

Finish

**Gambar 3.29** Diagram sistem kerja program alat

44

**BAB IV**

**PENGUJIAN ALAT**

Pada tugas akhir ini, pengujian dilakukan untuk mengetahui kinerja sistem secara keseluruhan. Pengujian sistem ini terdiri atas pengujian perangkat keras, pengujian perangkat lunak serta pengujian keseluruhan sistem.

**4.1 Pengujian Gerak Robot**

Pada pengujian gerak robot, dilakukan dengan cara 10 kali keberhasilan robot untuk menggerakkan rubik sampai berhasil mendapatkan enam sisi-sisi rubik terdeteksi dan mengubah langkah pergerakan rubik.

**Tabel 4. 1** Hasil pengujian pergerakan robot

|  |  |  |
| --- | --- | --- |
| Percobaan | Deteksi Sisi Rubik | Penyelesaian Rubik |
|  |  |  |
| 1 | Sukses | Sukses |
| 2 | Sukses | Sukses |
| 3 | Sukses | Sukses |
| 4 | Sukses | Gagal |
| 5 | Sukses | Sukses |
| 6 | Sukses | Sukses |
| 7 | Sukses | Gagal |
| 8 | Gagal | Sukses |
| 9 | Sukses | Sukses |
| 10 | Sukses | Sukses |

**4.2 Pengujian Perangkat Lunak**

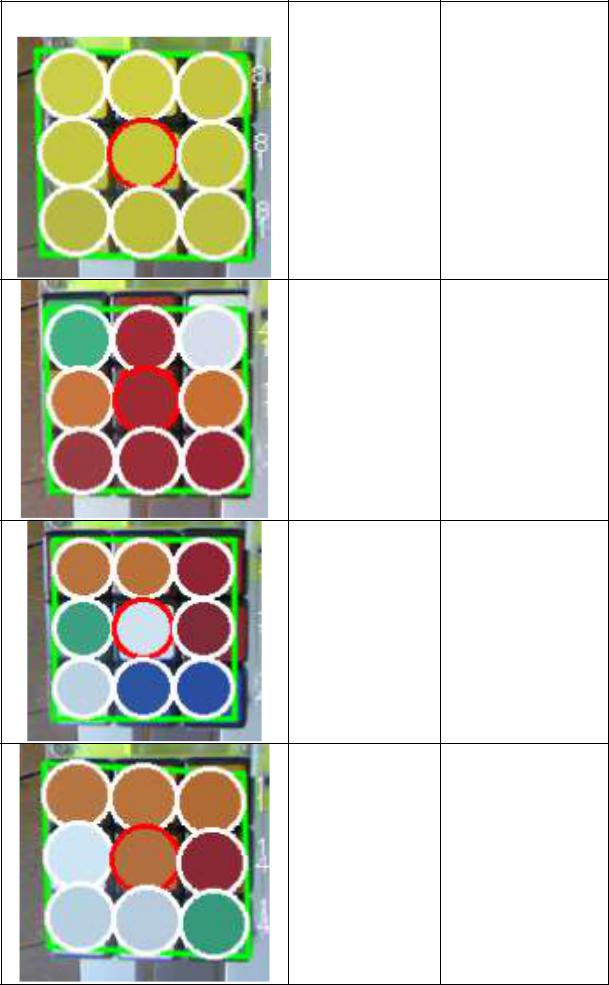
Pengujian perangkat lunak merupakan pengujian terhadap program pendeteksian rubik dan warna pada sisi-sisi rubik. Tujuan dari pengujian ini adalah untuk mendapatkan kondisi rubik dari user agar sistem bekerja pada tingkat keberhasilan maksimal.

45

**4.2.1** **Pengujian Deteksi Rubik**

Pengujian deteksi rubik dilakukan untuk mengetahui faktor-faktor yang mempengaruhi proses deteksi rubik dan pengujian ini dilakukan pada siang hari dan malam hari dengan tujuan untuk mencoba intensitas cahaya pada rubik yang berbeda. Pada saat pengujian terdapat prosedur yang harus dijalankan yaitu tidak ada garis hitam yang membentuk pola persegi selain rubik yang ingin kita deteksi.

**Tabel 4. 2** Hasil pengujian deteksi rubik pada siang hari



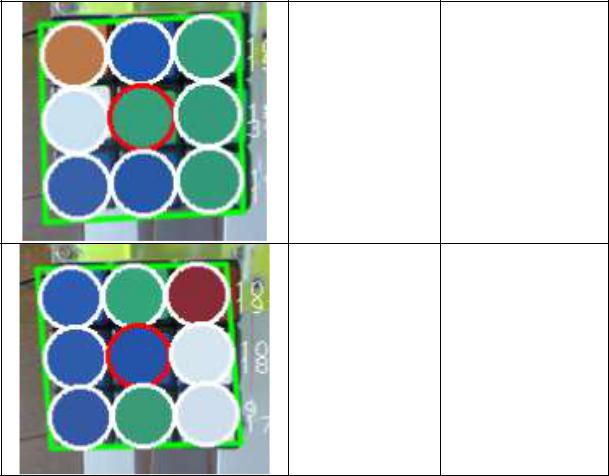
|  |  |  |
| --- | --- | --- |
| Gambar | Hasil | Threshold |
|  | Sukses | 76 |

Sukses 74

Sukses 70

Sukses 68

46

Sukses 69

Sukses 72

**Tabel 4. 3** Hasil pengujian deteksi rubik pada malam hari



|  |  |  |
| --- | --- | --- |
| Gambar | Hasil | Threshold |
|  | Sukses | 64 |

Sukses 88

Sukses 64

47

Sukses 75

Sukses 86

Sukses 75

**4.2.2** **Pengujian Identifikasi Warna Rubik**

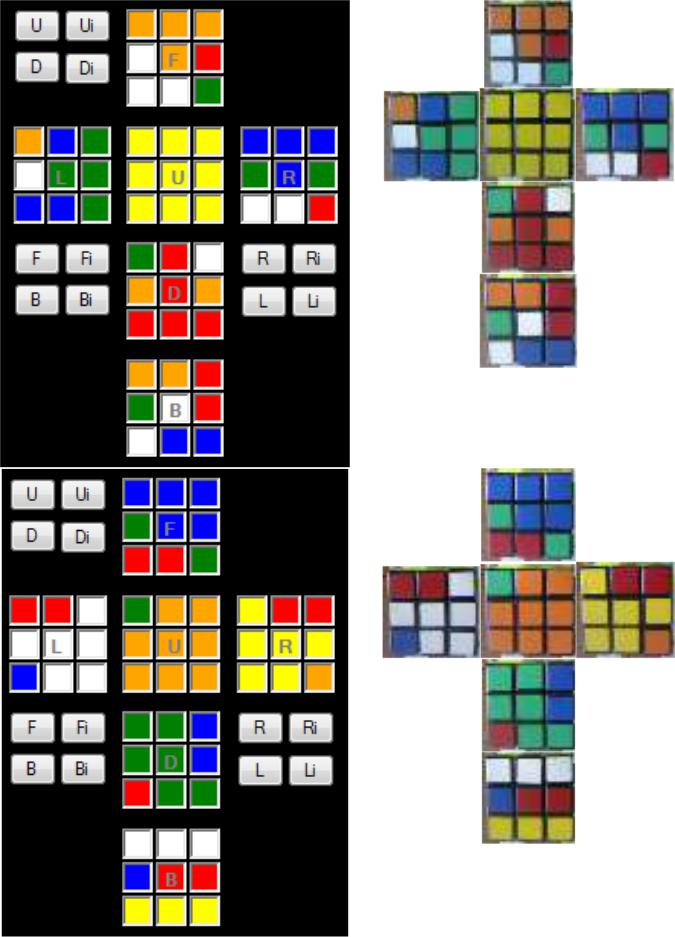
Pengujian pengidentifikasian warna rubik merupakan salah satu data yang sangat penting sebelum proses penyelesaian rubik. Data yang kita peroleh harus sesuai dengan rubik aslinya 100% karena data-data tersebut akan diproses lagi ke dalam *solving algorithms* untuk dicari lagi penyelesaiannya, bila adasalah satu data yang salah dari 54 data (warna pada tiap sisi rubik) maka penyelesaian rubik juga tidak bisa diselesaikan.

Pada pengujian ini dilakukan pada siang hari dan malam hari untuk mencoba intensitas cahaya yang berbeda dimana dapat mempengaruhi identifikasi warna yang diterima oleh kamera terhadap *user*.

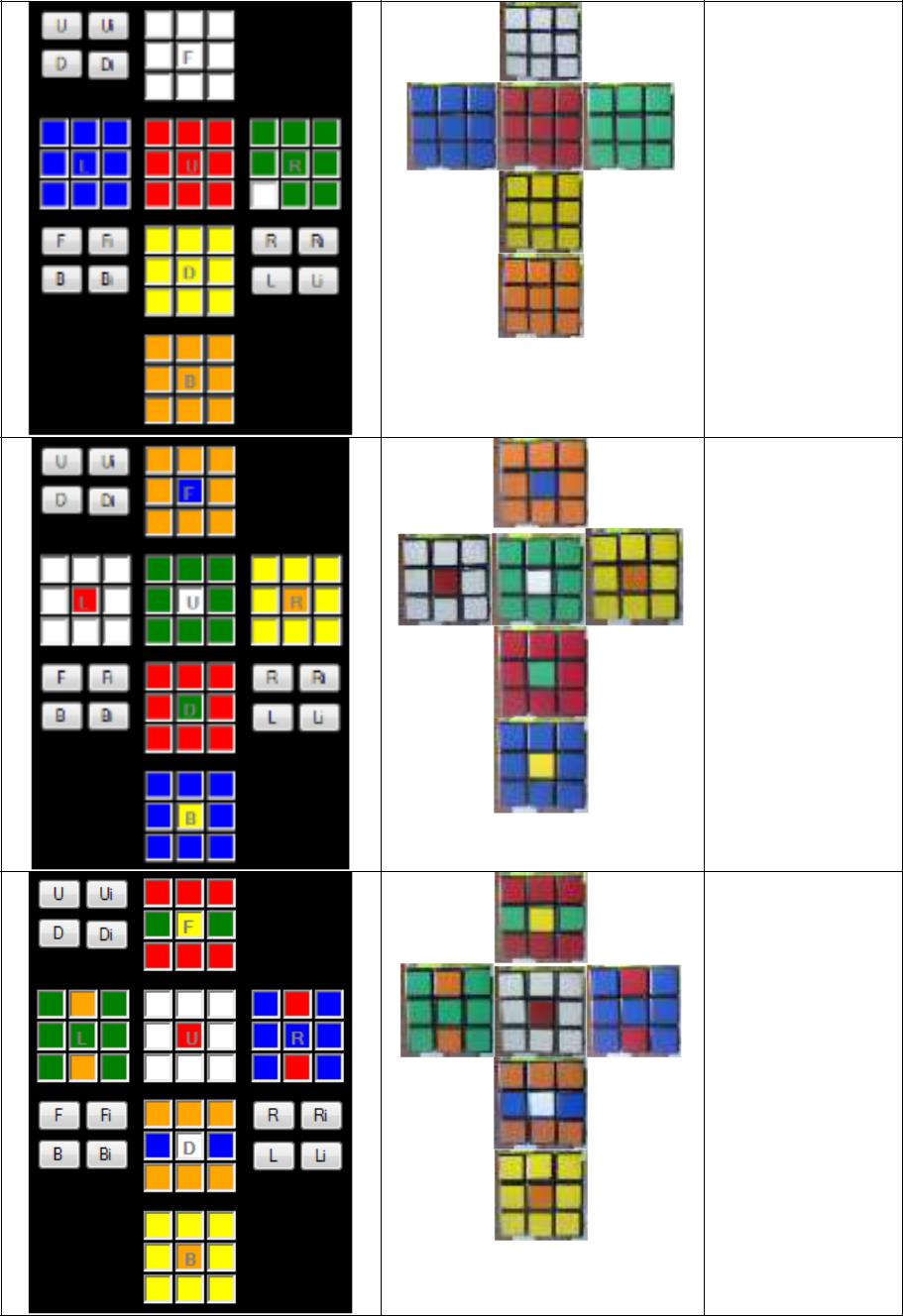
48

**Tabel 4. 4** Hasil pengujian identifikasi warna rubik pada siang hari

|  |  |  |
| --- | --- | --- |
| Gambar Data Warna Rubik | Gambar Rubik yang | Persentase |
|  | Asli | (%) |
|  |  | 100% |
|  |  |  |
|  |  | 100% |
|  |  |  |



49

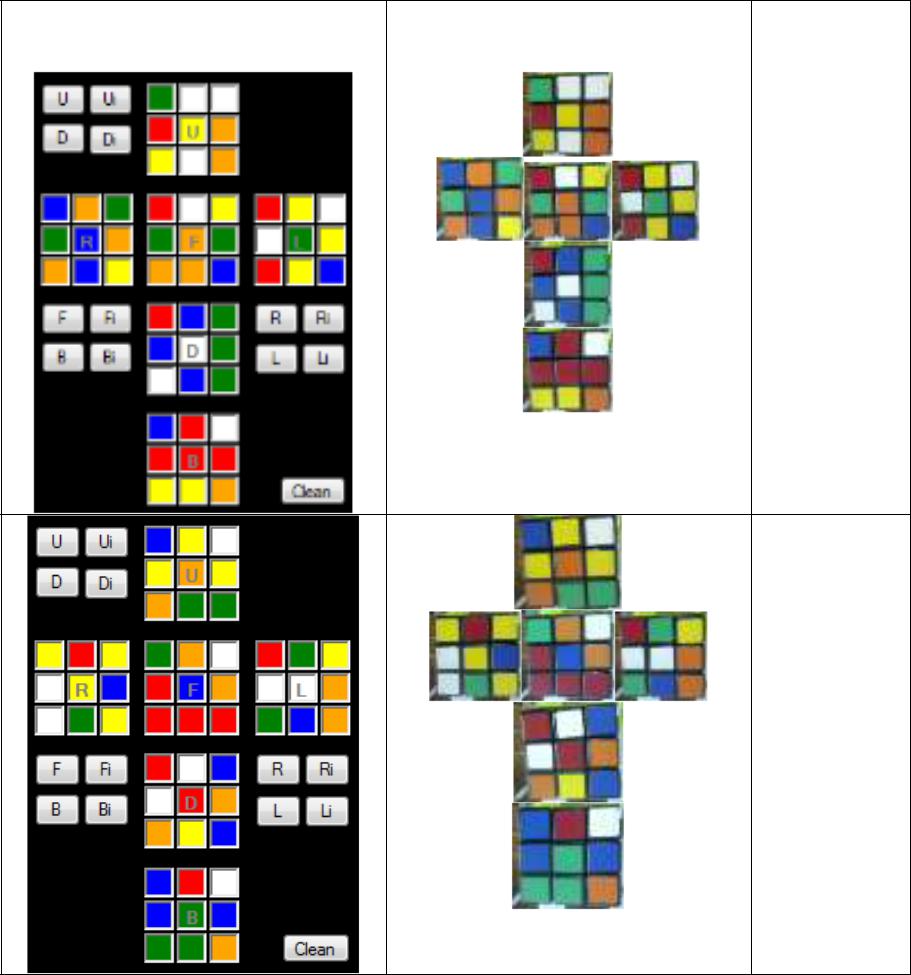
99%

100%

100%

50

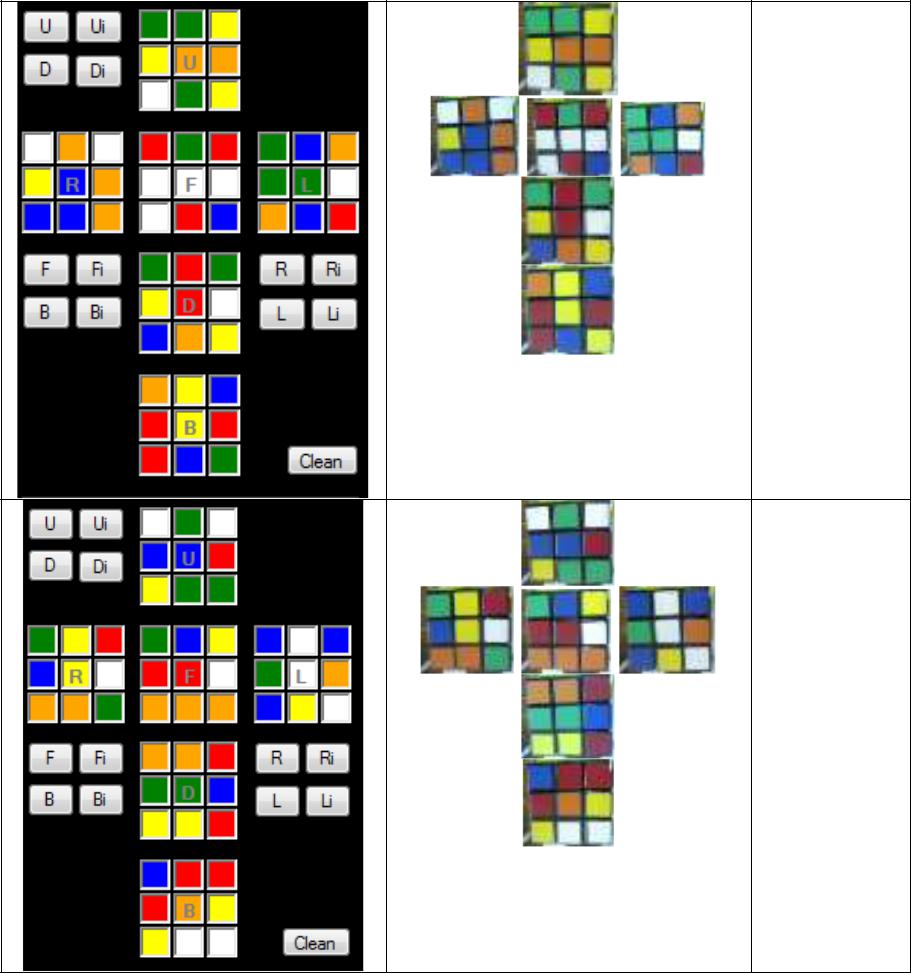
**Tabel 4. 5** Hasil pengujian identifikasi warna rubik pada malam hari



|  |  |  |
| --- | --- | --- |
| Gambar Data Warna Rubik | Gambar Rubik yang Asli | Persentase |
|  |  | (%) |
|  |  | 100 % |

100%

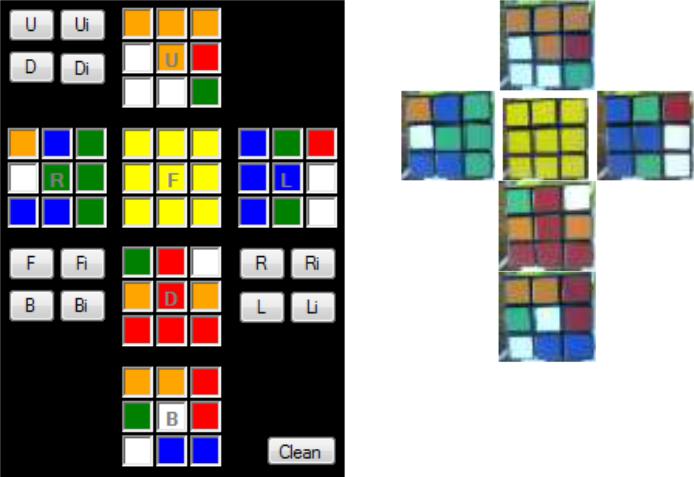
51

100%

100%

52

100%



**Tabel 4. 6** Hasil pengujian identifikasi warna dan penyelesaian rubik

|  |  |
| --- | --- |
| Percobaan | Identifikasi |
|  | Warna dan penyelesaian |
| 1 | Gagal |
| 2 | Sukses |
| 3 | Sukses |
| 4 | Sukses |
| 5 | Sukses |
| 6 | Gagal |
| 7 | Sukses |
| 8 | Sukses |
| 9 | Gagal |
| 10 | Sukses |

**4.3 Pembahasan Permasalahan**

Setelah dilakukan pengujian, didapatkan permasalahan yang perlu diatasi dalam tugas akhir ini. Ini ditujukan untuk mendapatkan hasil yang lebih baik dalam pendeteksian dan pembacaan warna rubik yang ada. Pendeteksian rubik ini menggunakan metode *Hough Transform* jadi sangat bergantung sekali dengan jumlah garis yang dapat dideteksi dan membentuk pola persegi. Untuk *solving algorithm* masih dilakukan

53

secara manual dimana input untuk menyelesaikan rubik sudah diketahui oleh *user* sebelumnya.

Pada proses pengujian gerak robot, rubik terkadang jatuh pada saat diputar secara vertikal. Ini disebabkan karena posisi rubik yang kurang presisi saat digenggam.



**Gambar 4. 1** Contoh rubik yang jatuh

Kesalahan pembacaan warna pada rubik terjadi bukan karena metode pembacaan yang kurang akurat, tetapi terjadi akibat posisi *LK* *tracking* rubik yang bergeser pada saat melakukan pergerakan langkahrubik dan tidak mengakibatkan terjadinya proses pendeteksian kembali.



**Gambar 4. 2** Contoh posisi*LK tracking*yang bergeser

Pada proses pendeteksian rubik, sinar matahari bukan merupakan salah satu faktor yang mempengaruhi pendeteksian. Dalam tugas akhir ini, nilai *thresholding* sudah dilakukan secara automatis. Bila garis

54

*Hough Transform* yang terdeteksi kurang dari 50 garis maka nilai *thresholding* kita tinggikan dan jika garis lebih dari 50 garis maka nilai *thresholding* kita turunkan.

55

*- Halaman ini sengaja dikosongkan -*

56

**BAB V**

**PENUTUP**

Setelah dilakukan rangkaian kegiatan perancangan sistem dan pengujian alat penulis memperoleh kesimpulan dan memberikan beberapa saran sebagai berikut

**5.1 Kesimpulan**

Kesimpulan yang diperoleh dalam Tugas Akhir ini adalah:

1. Prosedur yang harus dipenuhi untuk menggunakan robot rubik ini adalah tepi atau garis yang membentuk pola persegi diharapkan tidak ada karena akan mengganggu proses pendeteksian rubik.
2. Hasil pengujian pergerakan robot dalam 20 kali percobaan didapatkan hasil dengan proses keberhasilan 85%.
3. Hasil pengujian deteksi rubik dan identifikasi warna dalam 12 kali percobaan didapatkan hasil dengan proses keberhasilan 99%.
4. Pengujian dari penggabungan sistem pergerakan robot dan sistem pendeteksian serta identifikasi warna yang dilakukan sebanyak 10 kali percobaan didapatkan hasil dengan proses keberhasilan 70%.

**5.2 Saran**

Beberapa saran yang dapat penulis berikan untuk pengembangan

Tugas Akhir adalah sebagai berikut :

a. Mekanik yang dibuat masih sangat sederhana jadi untuk penyelesain rubik yang panjang membutuhkan waktu yang lama.

1. Metode *solving algorithm* masih perlu dikembangkan sehingga robot dapat menyelesaikan rubik itu sendiri tanpa dibantu oleh *user*.

57

*- Halaman ini sengaja dikosongkan -*

58

**DAFTAR PUSTAKA**

1. Ahmad, U. (2005). *Pengolahan Citra Digital dan Teknik* *Pemrogramannya.* Yogyakarta: Graha Ilmu.
2. Bourke, Paul, “Intersection of two Lines” <URL: [http://paulbourke.net/geometry/lineline2d/>,](http://paulbourke.net/geometry/lineline2d/) Agustus, 2011.
3. Castleman, Kenneth R. 1996. Digital Image Processing, Prentice Hall International Inc.
4. Gardner, Sue, “Hough Transform” <URL: [http://en.wikipedia.org/wiki/Hough\_transform>,](http://en.wikipedia.org/wiki/Hough_transform) September, 2011.
5. Kaehler, G. B. (2008). *Learning OpenCV.* Sebastopol: O'Reilly Media, Inc.
6. Karpathy, Andrej., “Extractng sticker colors on Rubik’s Cube

CPSC525 Project Report” <URL: [http://www.comp.leeds.ac.uk/fyproj/reports/0607/Green.pdf>,](http://www.comp.leeds.ac.uk/fyproj/reports/) Febuari, 2011.

1. Munir, R. (2004). *Pengolahan Citra Digital dengan Pendekatan* *Algoritmik.* Bandung: Informatika.
2. OpenCV Education Center, “CV Reference Manual“ <URL: [http://www.seas.upenn.edu/~bensapp/opencvdocs/ref/opencvref\_cv](http://www.seas.upenn.edu/~bensapp/opencvdocs/ref/opencvref_cv.htm)

[.htm>](http://www.seas.upenn.edu/~bensapp/opencvdocs/ref/opencvref_cv.htm) Maret, 2011.

1. Ramadi, S. (2009). *7 Jam Belajar Intreaktif Visual C++ .Net 2005* *untuk Orang Awam.* Palembang: Maxikom.
2. Seven Towns “Solution Guide”, London, 2008.
3. Vijn, J, “The Affine Transformation Matrix” <URL: [http://www.coranac.com/tonc/text/affine.htm>,](http://www.coranac.com/tonc/text/affine.htm) Maret, 2011.
4. Wahyu,Yohanes,“RahasiaFilterDigital”<URL:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | [http://www.fotografer.net/isi/artikel/lihat.php?id=381>,](http://www.fotografer.net/isi/artikel/lihat.php?id=381) | | | | September |
|  | , 2011. |  |  |  |  |
| [13] | Wikipedia, |  | “Aljabar | Linear” | <URL: |
|  | [http://id.wikipedia.org/wiki/Aljabar\_linear#Matriks\_Balikan\_.28In](http://id.wikipedia.org/wiki/Aljabar_linear#Matriks_Balikan_.28Invers.29) | | | | |
|  | [vers.29>,](http://id.wikipedia.org/wiki/Aljabar_linear#Matriks_Balikan_.28Invers.29) Agustus, 2011. | | |  |  |
| [14] | Yulias, | Zervani, | “Tutorial | Arduino” | <URL: |

http://blog.famosastudio.com/2011/07/tutorial/tutorial-arduino-servo/128>, November, 2011.

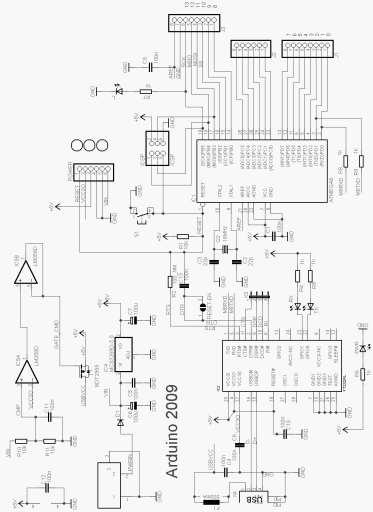
59

*- Halaman ini sengaja dikosongkan -*

60

**LAMPIRAN**

**SCHEMATIC ARDUINO DUEMILANOVE**



*- Halaman ini sengaja dikosongkan -*

**BIODATA PENULIS**

Maulana Rachman dilahirkan di Balikpapan 29 Mei 1988. Anak ketiga dari tiga bersaudara dari pasangan Ir. Abdullah Achmad



dan Milati Kamila Ulfa. Penulis menyelesaikan pendidikan dasar di SDN 015 Balikpapan kemudian penulis menyelesaikan pendidikan menengah di SMP Patra Dharma II Balikpapan dan SMA Patra Dharma Balikpapan. Pada tahun 2006, penulis memulai pendidikan di jurusan Teknik Elektro, Fakultas

Teknologi Industri, Institut Teknologi Sepuluh Nopember (ITS) Surabaya. Selama kuliah penulis aktif membantu penyelenggaran kegiatan dan aktif sebagai asisten laboratorium Elektronika Dasar pada semester 2009-2011.

Email : [tigerdota@yahoo.com](mailto:tigerdota@yahoo.com)

*- Halaman ini sengaja dikosongkan -*