TUGAS AKHIR – TE 141599

PENENTUAN POSISI ROBOT SEPAK BOLA BERODA MENGGUNAKAN *ROTARY ENCODER* DAN KAMERA

Adnan Rachmawan
NRP 2215105009

Dosen Pembimbing
Dr. Ir. Hendra Kusuma, M.Eng.Sc.
Ronny Mardiyanto, ST., MT., Ph.D.

DEPARTEMEN TEKNIK ELEKTRO
Fakultas Teknologi Elektro
Institut Teknologi Sepuluh Nopember
Surabaya 2017

TUGAS AKHIR - TE 141599

PENENTUAN POSISI ROBOT SEPAK BOLA BERODA MENGGUNAKAN *ROTARY ENCODER* DAN KAMERA

Adnan Rachmawan
NRP 2215105009

Dosen Pembimbing :
Dr. Ir. Hendra Kusuma, M.Eng.Sc.
Ronny Mardiyanto, ST., MT., Ph.D.

DEPARTEMEN TEKNIK ELEKTRO
Fakultas Teknologi Elektro
Institut Teknologi Sepuluh Nopember
Surabaya 2017

Institut
Teknologi
Sepuluh Nopember

FINAL PROJECT - TE 141599

WHEELED SOCCER ROBOT LOCALIZATION USING ROTARY ENCODER AND CAMERA

Adnan Rachmawan
NRP 2215105009

Supervisors :
Dr. Ir. Hendra Kusuma, M.Eng.Sc.
Ronny Mardiyanto, ST., MT., Ph.D.

ELECTRICAL ENGINEERING DEPARTEMENT
Faculty Of Electrical Technology
Institut Teknologi Sepuluh Nopember
Surabaya 2017

PERNYATAAN KEASLIAN TUGAS AKHIR

Dengan ini saya menyatakan bahwa isi sebagian maupun keseluruhan Tugas Akhir saya dengan judul "Penentuan Posisi Robot Sepak Bola Beroda Menggunakan *Rotary Encoder* Dan Kamera" adalah benar benar hasil karya intelektual mandiri, diselesaikan tanpa menggunakan bahan-bahan yang tidak diizinkan 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, Juli 2017

Adnan Rachmawan
NRP. 2215105009

PENENTUAN POSISI ROBOT SEPAK BOLA BERODA MENGGUNAKAN *ROTARY ENCODER* DAN KAMERA

TUGAS AKHIR

Diajukan Guna Memenuhi Sebagian Persyaratan
Untuk Memperoleh Gelar Sarjana Teknik
Pada
Bidang Studi Elektronika
Departemen Teknik Elektro
Fakultas Teknologi Elektro
Institut Teknologi Sepuluh Nopember

Menyetujui:

Dosen Pembimbing I	Dosen Pembimbing II
Dr. Ir. Hendra Kusuma, M.Eng.Sc.	**Ronny Mardiyanto, S.T., M.T., Ph.D.**
NIP. 196409021989031003	**NIP. 198101182003121003**

SURABAYA
JULI, 2017

PENENTUAN POISISI ROBOT SEPAK BOLA BERODA MENGGUNAKAN *ROTARY ENCODER* DAN KAMERA

Nama : Adnan Rachmawan
Pembimbing I : Dr. Ir. Hendra Kusuma, M.Eng.Sc.
Pembimbing II : Ronny Mardiyanto, S.T., M.T., Ph.D.

ABSTRAK

Penentuan posisi robot adalah salah masalah utama pada pembuatan robot otomatis. Apabila robot tidak mengetahui di mana posisinya, tindakan robot selanjutnya akan sulit untuk ditentukan. Salah satu cara untuk mengetahui posisi robot adalah dengan menggunakan *rotary encoder* untuk mengetahui seberapa jauh robot sudah berpindah dari posisi awalnya dengan dilengkapi sensor kompas digital. Namun cara ini mudah terpengaruh slip sehingga menghasilkan error yang nilainya selalu terintegral sehingga semakin lama semakin besar. Untuk mengurangi error ini diperlukan sensor lain yang tidak terpengaruh pembacaan sebelumnya yaitu kamera yang terpasang pada robot. Pada tugas akhir ini, dibahas mengenai penggunaan *rotary encoder* yang nilainya diperbaharui menggunakan hasil pengolahan citra kamera dan jaringan syaraf tiruan. Hasil menunjukkan bahwa penentuan posisi robot sepak bola beroda menggunakan *rotary encoder* dan *gyroscope* memiliki *error* RMS sebesar 44.60 cm pada sumbu x dan 17.56 cm pada sumbu y, sedangkan penentuan posisi robot sepak bola beroda menggunakan kamera memiliki *error* RMS sebesar 16,24 cm pada sumbu x dan 10,50 cm pada sumbu y. Oleh karena itu penentuan posisi robot sepak bola beroda menggunakan *rotary encoder* dan *gyroscope* dapat diminimalisir menggunakan kamera.

Kata Kunci: Penentuan Posisi Robot, *rotary encoder*, *gyroscope*, kamera, jaringan syaraf tiruan.

Halaman ini sengaja dikosongkan

WHEELED SOCCER ROBOT LOCALIZATION USING ROTARY ENCODER AND CAMERA

Name **: Adnan Rachmawan**
Supervisor **: Dr. Ir. Hendra Kusuma, M.Eng.Sc.**
Co-Supervisor **: Ronny Mardiyanto, S.T., M.T., Ph.D.**

ABSTRACT

Robot localization is one of main problem in automatic robot. It is difficult to plan the motion of robot if robot position unknown. One way to localize robot position is using rotary encoder to know how far robot has moved from its starting position and is equipped with digital compass sensor, but this method is affected by slip. This problem results integrated error. To overcome this problem, it needs another sensor that don't affected by previous reading, that is the camera mounted on the robot. In this final project, it is explained about the use of rotary encoder which its value is updated using image processing and artificial neural network. The result shows that soccer robot localization using rotary encoder and gyroscope has RMS error value 44.60 cm on x axis and 17.56 cm on y axis, while soccer robot localization using camera has 16.24 cm on x axis and 10.50 cm on y axis. Therefore soccer robot localization using rotary encoder and gyroscope can be minimized using camera.

Keywords*: Robot Localization, Rotary Encoder, Camera, Artificial Neural Network.*

Halaman ini sengaja dikosongkan

KATA PENGANTAR

Alhamdulillahirobbil'alamin, puji syukur senaniasa terpanjatkan kepada Allah *Subhanahu wa Ta'ala* yang atas rahmat, petunjuk dan kasih sayang-Nya, sehingga penelitian dan penulisan tugas akhir dengan judul "**Penentuan Posisi Robot Sepak Bola Beroda Menggunakan *Rotary Encoder* dan Kamera"** ini dapat diselesaikan dengan baik.

Dalam proses menyelesaiakan tugas akhir ini penulis mendapat banyak bantuan dan dukungan dari berbagai pihak. Penulis mengucapkan banyak terima kasih pada:

1. Bapak, Ibu, Kakak, adik yang telah memberikan dukungan baik moril mauput materiil
2. Dr. Ir. Hendra Kusuma, M.Eng.Sc selaku dosen pembimbing 1 atas bimbingan dan arahan selama penulis mengerjakan tugas akhir ini.
3. Ronny Mardiyanto, S.T., M.T., Ph.D. selaku dosen pembimbing 2 atas bimbingan dan arahan selama penulis mengerjakan tugas akhir ini.
4. Ir. Tasripan, M.T. selaku Koordinator Bidang Studi Elektronika.
5. Dr.Eng. Ardyono Priyadi, S.T., M.Eng. selaku Ketua Jurusan Teknik Elektro ITS Surabaya.
6. Seluruh dosen bidang studi Elektronika Jurusan Teknik Elektro ITS Surabaya.
7. Tim Robot ITS KRSBI Beroda 2017 yang telah membantu penyelesaian tugas akhir ini.
8. Rusyda Dyah yang senantiasa memberikan semangat dan motivasi kepada penulis.

Penulis menyadari bahwa tugas akhir ini tidaklah sempurna, namun penulis berharap agar tugas akhir ini dapat menjadi bahan bagi rekan-rekan untuk menambah wawasan serta dapat membantu pengerjaan tugas akhir selanjutnya.

Surabaya, Juli 2017

Adnan Rachmawan

Halaman ini sengaja dikosongkan

DAFTAR ISI

Halaman

DAFTAR GAMBAR

Halaman

DAFTAR TABEL

Halaman

Halaman ini sengaja dikosongkan

BAB I

PENDAHULUAN

1.1. Latar Belakang

Penentuan posisi robot adalah masalah utama pada pembuatan robot otomatis. Apabila robot tidak mengetahui di mana posisinya, tindakan robot selanjutnya akan sulit untuk ditentukan. Untuk mengetahui posisinya, robot mengolah data dari sensor-sensor yang dimiliki. Pada robot beroda, penentuan posisi robot yang umum digunakan adalah menggunakan *rotary encoder* untuk mendapatkan kecepatan putaran setiap roda yang dilengkapi dengan sensor kompas digital berupa gyroscope untuk mengetahui arah muka robot. Nilai kecepatan dan arah inilah yang diolah sedemikian rupa sehingga diperoleh posisi robot.

Saat robot bergerak pada lingkungannya, posisi yang terukur berbeda dari yang diperintahkan pada robot karena adanya pengaruh dari luar. Penggunaan r*otary encoder* misalnya, rawan mengalami slip akibat lantai yang licin. Kesalahan akibat slip tersebut akan terakumulasi setiap waktu, sehingga semakin lama posisi yang terukur semakin jauh berbeda dengan posisi yang sesungguhnya. Oleh karena itu dibutuhkan suatu metode untuk mengurangi kesalahan pengukuran akibat adanya slip tersebut.

Robot sepak bola beroda dengan lingkungannya berupa lapangan sepak bola, sangat memerlukan posisi dari dirinya agar perencanaan gerakannya menjadi lebih efisien, sehingga kesalahan penentuan posisi robot perlu ditanggulangi. Penanggulangan yang dapat dilakukan salah satunya adalah dengan memanfaatkan sensor yang kesalahannya tidak terpengaruh oleh kesalahan sebelumnya, misalnya adalah penggunaan kamera yang terdapat pada robot. Pada [1], kamera dapat digunakan untuk mengetahui jarak robot lain, dengan penalaran yang serupa, kamera juga terhadap markah lapangan yang posisinya sudah diketahui. Dengan mengetahui jarak robot terhadap markah lapangan pada setiap sudut, posisi dari robot dapat diketahui dengan menggunakan jaringan syaraf tiruan yang telah dilatih secara *offline*. Dengan memanfaatkan *rotary encoder* dan kamera sebagai kalibrasinya, lokasi yang akurat dapat diperoleh.

Dalam penelitian ini, telah dirancang dan dibuat sistem penentuan posisi robot dengan memanfaatkan sistem *odometry* pada robot yang menggunakan *rotary encoder* dan *gyroscope* yang nilainya dapat diperbaharui dari hasil pengolahan citra dari kamera.

1.2. Rumusan Permasalahan

Permasalahan yang dibahas pada tugas akhir ini adalah:

a. Bagaimana memperoleh posisi robot berdasarkan keluaran *rotary encoder* dan *gyroscope*.
b. Bagaimana memperoleh jarak piksel robot terhadap marka lapangan.
c. Bagaimana memperoleh posisi robot berdasarkan data berupa jarak piksel robot terhadap marka lapangan menggunakan jaringan syaraf tiruan.
d. Bagaimana memperbaharui posisi robot berdasarkan hasil keluaran jaringan syaraf tiruan.

1.3. Batasan Masalah

Adapun batas ruang lingkup dari tugas akhir iniadalah:

a. Objek yang terdapat pada lapangan hanya satu buah robot.
b. Metode yang digunakan untuk mengenali lapangan berdasarkan pembacaan kamera adalah dengan jaringan syaraf tiruan.
c. Pembaharuan posisi robot hanya pada arah 0^o.
d. Luas area yang dikenali menggunakan jaringan syaraf tiruan digunakan adalah setengah lapangan sepak bola pada pertandingan robot sepak bola beroda, yaitu 4,5 m x 6 m.

1.4. Tujuan

Tujuan yang ingin dicapai dalam perancangan ini adalah:

a. Menentukan posisi robot berdasarkan pergerakan *rotary encoder* dan *gyroscope*.
b. Menentukan jarak piksel robot terhadap marka lapangan sebagai masukan dari jaringan syaraf tiruan.
c. Memperbaharui posisi robot berdasarkan hasil keluaran jaringan syaraf tiruan.

1.5. Metodologi

Langkah-langkah yang dikerjakan pada tugas akhir ini adalah sebagai berikut:

a. Studi Literatur

Pada tahap ini dilakukan pengumpulan dasar teori yang menunjang dalam penulisan Tugas Akhir. Dasar teori ini dapat diambil dari buku-buku, jurnal, dan artikel-artikel di internet serta forum diskusi internet.

b. Perancangan Perangkat Keras

Pada tahap ini dilakukan perancangan posisi *pemasangan rotary encoder* dan kamera pada robot.

c. Perancangan Perangkat Lunak

Perancangan perangkat lunak terdiri dari perancangan pada sistem mikrokontroler dan PC. Mikrokontroler berfungsi sebagai *slave* yang melakukan pembacaan *rotary encoder* dan kompas yang hasilnya dikirimkan kepada PC. Pengolahan data kamera dilakukan pada PC di mana citra dari kamera difilter, disegmentasi hingga diperoleh fitur-fitur berupa jarak piksel robot terhadap marka lapangan. Fitur-fitur tersebut kemudian menjadi masukan jaringan syaraf tiruan dan akan dihasilkan posisi dari robot untuk dilakukan pembaharuan posisi.

d. Pengujian Sistem

Pengujian sistem dilakukan untuk menentukan keandalan dari sistem yang telah dibuat. Pengujian dilakukan untuk melihat apakah perangkat keras dan lunak telah bekerja dengan baik.

Pengujian dilakukan pada setiap sensor, yaitu dari *rotary encoder*, kompas dan kamera. Pengujian *rotary encoder* dilakukan dengan mengubah-ubah posisi robot dan merekam hasilnya. Pengujian sensor kompas dilakukan dengan mengubah-ubah orientasi dari robot dan merekam hasilnya. Pengujian kamera dilakukan pada kondisi pencahayaan yang berbeda.

e. Analisa

Data-data yang diperoleh pada pengujian dianalisa lebih lanjut untuk memperoleh kelebihan dan kekurangan dari sistem.

f. Penyusunan Laporan Tugas Akhir

Tahap penyusunan laporan tugas akhir adalah tahapan terakhir dari proses pengerjaan tugas akhir ini. Laporan tugas akhir berisi seluruh hal yang berkaitan dengan tugas akhir yang telah dikerjakan yaitu meliputi pendahuluan, tinjauan pustaka dan teori penunjang, perancangan sistem, pengujian dan penutup.

1.6. Sistematika

Dalam buku tugas akhir ini, pembahasan mengenai sistem yang dibuat dibagi menjadi lima bab dengan sistematika penulisan sebagai berikut:

BAB I : PENDAHULUAN

Bab ini meliputi penjelasan latar belakang, perumusan masalah, batasan masalah, tujuan, metodologi, sistematika penulisan, dan relevansi.

BAB II : TINJAUAN PUSTAKA DAN TEORI PENUNJANG

Bab ini menjelaskan teori penunjang dan literatur yang dibutuhkan dalam pengerjaan tugas akhir. Dasar teori yang menunjang meliputi citra digital, jaringan syaraf tiruan, *rotary encoder* dan gyroscope.

BAB III : PERANCANGAN SISTEM

Bab ini menjelaskan tentang perancangan sistem, baik perangkat keras maupun perangkat lunak untuk sistem penentuan posisi robot menggunakan *rotary encoder* dan kamera.

BAB IV : PENGUJIAN

Pada bab ini dijelaskan hasil uji coba sistem beserta analisanya.

BAB V : PENUTUP

Bagian ini merupakan bagian akhir yang berisikan kesimpulan yang diperoleh dari pembuatan Tugas Akhir ini, serta saran-saran pengembangan lebih lanjut.

1.7. Relevansi dan Manfaat

Hasil yang diharapkan dari tugas akhir ini diharapkan mampu meningkatkan performansi perencanaan gerak pada robot sepak bola beroda.

BAB II

TINJAUAN PUSTAKA

Pada bab ini akan dijelaskan mengenai teori-teori penunjang dalam penentuan posisi robot sepak bola beroda menggunakan *rotary encoder* dan kamera.

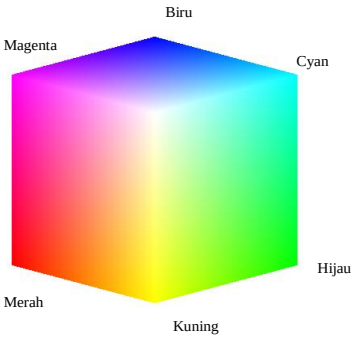
2.1. Pengolahan Citra Digital

Citra dapat didefinisikan sebagai fungsi dua dimensi, $f(x, y)$, di mana x dan y adalah koordinat bidang, dan amplitudo dari f pada setiap koordinat (x, y), disebut intensitas atau tingkat keabuan pada titik tersebut. Saat semua nilai x, y dan amplitudo dari f memiliki nilai yang terbatas, yang bernilai diskrit, citra tersebut disebut dengan citra digital. Pemrosesan citra digital dapat didefinisikan sebagai pemrosesan citra digital menggunakan computer digital. Citra digital tersusun atas elemen-elemen dengan jumlah tertentu yang terbatas, di mana masing-masing memiliki lokasi dan nilai terntentu. Elemen-elemen inilah yang disebut dengan elemen citra, atau biasa disebut dengan pixel [1].

Pada pengolahan citra digital, kualitas dari aspek radiometrik; yang terdiri dari peningkatan kontras, restorasi citra, transformasi warna; dan aspek geometrik; yang terdiri dari rotasi, skala, translasi dan transformasi geometrik. Pengolahan ini bertujuan untuk memperoleh informasi yang terkandung dari citra, untuk kemudian diolah dan dianalisa [2].

2.1.1. Citra Dengan Ruang Warna RGB

Berdasarkan penelitian, secara umum mata manusia dapat membedakan warna merah, hijau dan biru (*Red, Green, Blue*). Oleh karena itu, warna dapat dipandang sebagai kombinasi dari ketiga warna tersebut [1]. Ruang warna RGB ini biasa diterapkan pada kebanyakan sistem grafika computer. Ruang warna ini biasanya dimodelkan dalam bentuk kubus tiga dimensi, dengan warna merah, hijau dan biru berada pada sudut sumbu. Warna hitam berada pada titik asal dan warna putih berada di ujung kubus yang berseberangan. Gambar 2.1 menujukkan visualisasi ruang warna RGB menggunakan resolusi 24 bit.

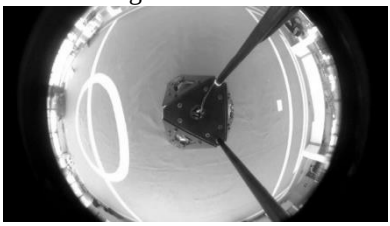


Gambar 2. 1 Model Ruang Warna RGB

Pada aplikasinya, ruang warna ini tidak ideal untuk digunakan, karena warna warna merah, hijau dan biru sesungguhnya terkolerasi erat sehingga sulit untuk beberapa algoritma pemrosesan citra. Dalam aplikasi yang memproses warna tertentu, lebih mudah apabila menggunakan ruang warna HSV [2].

2.1.2. Citra dengan Skala Keabuan

Citra jenis ini memiliki gradasi warna hitam dan putih, yang menghasilkan efek warna abu-abu. Intensitas warna citra skala keabuan berkisar antara 0 sampai dengan 255, dengan 0 menyatakan hitam dan 255 menyatakan putih. Gambar 2.2 menunjukkan contoh citra dengan skala keabuan.



Gambar 2. 2 Citra dengan Skala Keabuan

2.1.3. Citra Dengan Ruang Warna HSV

HSV merupakan ruang warna yang merepresentasikan warna seperti yang dilihat oleh mata manusia. H berasal dari kata "hue", yaitu sesuatu yang biasa disebut oleh sebagian orang sebagai warna, misalnya perbedaan antara warna merah dan merah muda. S berasal dari "saturation" yang menunjukkan jumlah warna yang ada. V berasal dari "value" yang menunjukkan jumlah cahaya yang ada, misalnya perbedaan antara merah gelap dengan merah cerah [3]. Model ruang warna HSV ditunjukkan oleh Gambar 2.3. Gambar 2.4 dan 2.5 menunjukkan contoh hasil pengubahan warna RGB menjadi HSV.

Nilai H, S dan V dapat diperoleh berdasarkan nilai R, G dan B dengan rumus sebagai berikut [2]:

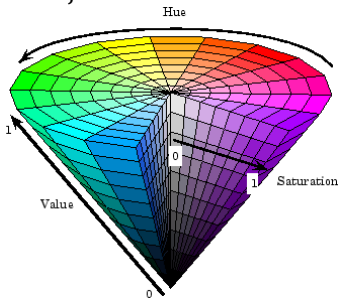
$$r = \frac{R}{(R+G+B)}, g = \frac{G}{(R+G+B)}, b = \frac{B}{(R+G+B)} \quad (2.1)$$

$$V = \max(r, g, b) \quad (2.2)$$

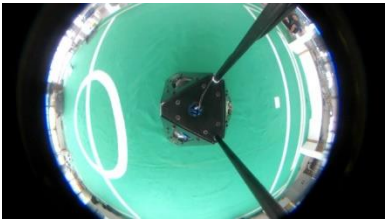
$$S = \begin{cases} 0, & \text{jika V} = 0 \\ 1 - \frac{\min(r,g,b)}{V}, & \text{V} > 0 \end{cases} \quad (2.3)$$

$$H = \begin{cases} 0, & jika\ S = 0 \\ \frac{60*(g-b)}{S*V}, & jika\ V = r \\ 60 * \left[2 + \frac{b-r}{S*V}\right], & jika\ V = g \\ 60 * \left[4 + \frac{r-g}{S*V}\right], & jika\ V = b \end{cases} \quad (2.4)$$

$$H = \text{H} + 360 \text{ jika H} < 0 \quad (2.5)$$



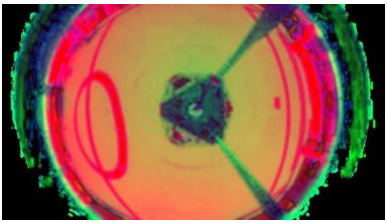
Gambar 2. 3 Model Ruang Warna HSV



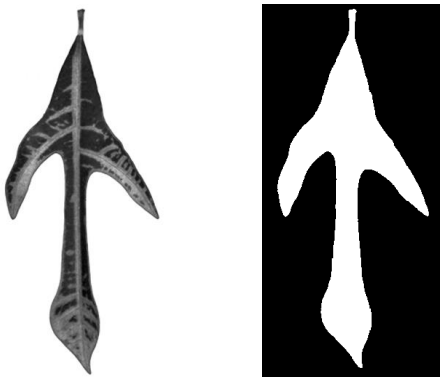
Gambar 2. 4 Citra dengan Ruang Warna RGB

2.1.4. Citra Biner

Citra biner adalah citra dengan nilai piksel yang hanya memiliki dua kemungkinan, yaitu nilai 0, yang merepresentasikan warna hitam, atau 1, yang merepresentasikan warna putih. Citra jenis ini biasa digunakan untuk kepentingan memperoleh bentuk suatu objek, seperti yang dicontohkan pada gambar 2.6.



Gambar 2. 5 Citra dengan Ruang Warna HSV



Gambar 2. 6 Citra Daun Berskala Keabuan dan Biner

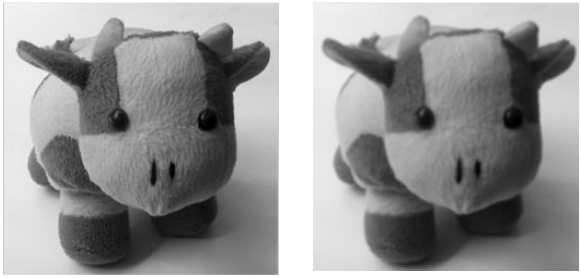
Untuk menkonversi suatu citra dengan ruang warna tertentu menjadi citra biner, digunakan suatu nilai yang disebut sebagaai nilai ambang. Nilai tersebut dipakai untuk menentukan suatu intensitas akan dikonversi menjadi 1 [2].

2.1.5. Filter Lolos-Bawah

Filter lolos-bawah (*low-pass filter*) adalah filter dengan karakteristik yang akan melewatkan sinyal dengan frekuensi yang rendah dan menghilangkan sinyal dengan frekuensi tinggi. Filter ini berguna untuk menghaluskan derau atau untuk kepentingan interpolasi tepi objek dalam citra. Salah satu filter lolos-bawah yang umum digunakan pada pengolahan citra adalah filter *Gaussian*. Model dua dimensi dari filter gaussian adalah sebagai berikut:

$$G(y.x) = e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{2.6}$$

Dengan σ adalah deviasi standar dengan piksel pusat (x,y) memiliki bobot terbesar dengan nilai 1 [2]. Filter *Gaussian* memiliki ukuran kernel minimal sebesar 5x5. Gambar 2.6 adalah contoh hasil penerapan filter *Gaussian* ini.

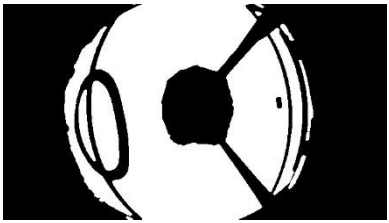


Gambar 2. 7 Citra Boneka Sebelum dan Setelah Dilakukan Filter Gaussian

2.1.6. Segmentasi Citra

Segmentasi citra merupakan proses yang ditujukan untuk mendapatkan objek-objek yang terkandung di dalam citra atau membagi citra ke dalam beberapa daerah dengan setiap objek atau daerah memiliki kemiripan atribut. Citra dengan satu objek saja dapat dibedakan dari latar belakangnya, sedangkan citra yang mengandung sejumlah objek, segmentasi menjadi lebih kompleks.

Segmentasi dilakukan sebagai langkah awal untuk klasifikasi objek. Dari hasil segmentasi ini kemudian dapat diperoleh fitur dari objek yang diambil. Salah teknik segmentasi yang biasa digunakan adalah teknik peng-ambangan warna. Teknik ini dapat dilakukan dengan cara yang sederhana, yaitu dengan menggunakan ambang intensitas. Nilai yang lebih kecil daripada nilai ambang dikelompokkan sebagai area pertama dan yang lebih besar atau sama dengan dikelompokkan sebagai area yang ke dua [2]. Gambar 2.8 menunjukkan contoh peng-ambangan warna hijau.



Gambar 2. 8 Gambar Hasil Peng-ambangan Warna

2.1.7. Open Source Computer Vision

Open Source Computer Vision atau yang lebih dikenal dengan OpenCV adalah pustaka fungsi-fungsi pemrograman untuk computer vision. Pustaka ini menggunakan lisensi BSD dan oleh karena itu, pustaka ini bebas digunakan untuk bidang akademik maupun komersial. Pustaka ini ada dalam bahasa C++, Python dan Java (Android) dan memiliki dukungan untuk sistem operasi Windows, Linux, Android, iOS dan Mac OS [4].

Sejak versi alfa dirilis pada bulan Januari 1999, OpenCV sudah digunakan pada berbagai aplikasi, produk dan riset. Aplikasi-aplikasi ini antara lain pemrosesan citra satelit, pengurangan derau pada citra medis, analisa obyek, sistem monitoring dan keamanan, dan lain-lain [4]. Gambar 2.9 menunjukkan logo dari OpenCV.

2.2. Jaringan Syaraf Tiruan

Jaringan syaraf tiruan adalah sebuah model dari otak manusia yang terdiri dari neuron dan interkoneksi yang dapat belajar dan memiliki kemampuan dalam hal persepsi dan pengenalan suatu bentuk atau pola.

Elemen dasar dari jaringan syaraf tiruan adalah neuron, seperti halnya elemen dasar dari otak manusia. Neuron adalah sebuah model dari neuron natural, seperti yang ditunjukkan oleh Gambar 2.10. Neuron memiliki x_1, x_2, …, x_m yang masuk pada neuron. Masukan-masukan ini adalah tingkatan stimulus pada neuron natural. Setiap masukan x_i dikalikan dengan masing masing bobotnya (w_i), lalu hasil perkalian dari $x_i w_i$ masuk pada neuron [5].

Gambar 2. 9 Logo OpenCV

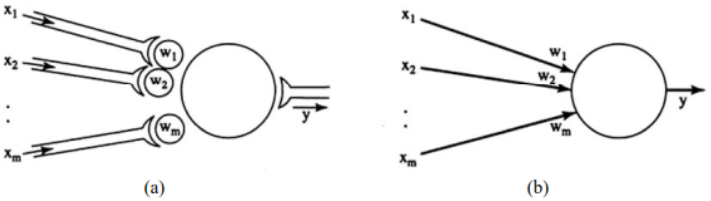
Pada jaringan syaraf tiruan yang paling sederhana, pembelajaran dilakukan dengan memberikan urutan vektor atau pola pelatihan di mana masing-masing memiliki vektor target yang ingin dicapai. Bobot-bobotnya kemudian diperbaharui nilainya menggunakan algoritma pelatihan. Proses ini disebut dengan *supervised training*.

Jaringan syaraf tiruan yang paling sederhana digunakan untuk mengklasifikasikan pola, misalnya untuk mengklasifikasikan apakah suatu vektor termasuk pada kategori tertentu atau tidak. Pada jenis ini, keluaran yang dihasilkan adalah 1 atau -1. Untuk masalah klasifikasi yang lebih sulit, digunakan jaringan dengan *multilayer* yang dilatih dengan algoritma *backpropagation* [6].

2.2.1. Back Propagation

Back propagation adalah sebuah metode yang dirumuskan oleh Werbos dan dipopulerkan oleh Rumelhart dan McClelland, yang sistematik untuk pelatihan jaringan saraf tiruan. Algoritma ini didasarkan pada interkoneksi yang sederhana yaitu jika keluaran memberikan hasil yang salah, maka bobot dikoreksi supaya error dapat diperkecil dan keluaran selanjutnya diharapkanakan lebih mendekati nilai yang benar.

a. Dimulai dengan lapisan masukan, hitung keluaran dari setiap elemen pemroses melalui lapisan luar.
b. Hitung kesalahan pada lapisan luar yang merupakan selisih antara data aktual dan target.
c. Transformasikan kesalahan tersebut pada kesalahan yang sesuai di sisi masukan elemen pemroses.
d. Rambatkan balik kesalahan-kesalahan ini pada keluaran setiap elemen pemroses ke kesalahan yang terdapat pada masukan. Ulangi proses ini sampai masukan tercapai.
e. Ubah seluruh bobot dengan menggunakan kesalahan pada sisi masukan elemen dan luaran elemen pemroses yang terhubung.



Gambar 2. 10 Model Neuron Natural

Fungsi aktivasi yang digunakan dari metode error backpropagation algorithm adalah fungsi sigmoid pada bagian hidden layer dan fungsi linier pada bagian output layer.

Algoritma yang digunakan pada pembelajaran ini terdiri dari tiga tahap utama, yaitu proses feedforward, backpropagation dan proses perbaharui bobot. Pada proses feedforward, setiap unit masukan dijumlahkan pada node-node layer 1 (hidden layer 1) yang kemudian diaktifasi menggunakan fungsi sigmoid dan menghasilkan Zj, dengan j adalah layer. Hasil-hasil tersebut kemudian djumlahkan ke layer selanjutnya hingga dihasilkan Yj pada layer output atau layer terakhir. Nilai tersebut kemudian dibandingkan dengan nilai keluaran yang diharapkan sehingga didapatkan error. Error tersebut kemudian dirambatkan balik pada setiap bobot untuk didapatkan bobot baru pada proses pembaharuan bobot [6]. Berikut ini merupakan algoritma pelatihan menggunakan *backpropagation*.

Prosedur inisialisasi

Langkah 0: Inisialisasi bobot dan masukan dengan menggunakan metode *random gaussian* dengan rata-rata 0 dan deviasi 0,1.

Langkah 1: Jika kondisi tidak tercapai, maka ulangi langkah 2 – 9.

Langkah 2: Untuk setiap pasangan pembelajaran, lakukan langkah 3 - 8.

Prosedur Feedforward

Langkah 3: Tiap unit masukan (x_i, i = 1,…, n) menerima sinyal x_i dan menghantarkan sinyal ini ke semua unit lapisan *hidden layer* 1,

Langkah 4: Setiap unit pada *hidden layer* 1 ($z1_j$, j = 1,…, p) jumlahkan bobot sinyal masukannya,

$$z\_in1_j = v1_{0j} + \sum_{i=1}^{n} x_i \, v1_{ij} \tag{2.7}$$

Aplikasikan fungsi aktivasinya untuk menghitung sinyal keluarannya, $z1_j = f(z\_in1_j)$, dan kirimkan sinyal ini keseluruh unit pada lapisan diatasnya (*hidden layer 2).*

Langkah 5: Setiap unit pada *hidden layer* 2 ($z2_k$, k = 1,…, q) jumlahkan bobot sinyal masukannya,

$$z\_in2_k = v2_{0k} + \sum_{i=1}^{p} z1_j \, v2_{jk} \tag{2.8}$$

Aplikasikan fungsi aktivasinya untuk menghitung sinyal keluarannya, $z2_k = f(z\_in2_k)$, dan kirimkan sinyal ini keseluruh unit pada lapisan diatasnya (unit keluaran).

Langkah 6: Tiap unit keluaran (y_l, l = 1,…, m) jumlahkan bobot sinyal masukannya,

$$y\_in_l = w_{0l} + \sum_{j=1}^{q} z2_k w_{kl} \tag{2.9}$$

w_{0k} = bias pada unit keluaran l dan aplikasikan fungsi aktivasinya untuk menghitung sinyal keluarannya, $y_k = f(y\_in_k)$.

Prosedur Back Propagation

Langkah 7: Tiap unit keluaran (y_l, l = 1,…, m) menerima pola target yang saling berhubungan pada masukan pola pelatihan, hitung kesalahan informasinya,

$$\delta_l = (t_l - y_l) f'(y_{in_l}) \quad (2.10)$$

Hitung koreksi bobot (digunakan untuk memperbaharui w_{jk} nantinya),

$$\Delta w_{kl} = \mu \delta_l z2_k + \alpha[w_{kl}(t) - w_{kl}(t-1)] \quad (2.11)$$

Hitung koreksi bias (digunakan untuk memperbaharui w_{0k} nantinya).

$$\Delta w_{0l} = \mu \delta_l z_k + \alpha[w_{0l}(t) - w_{0l}(t-1)] \quad (2.12)$$

δ_l lalu digunakan pada layer selanjutnya.

Langkah 8: Setiap unit pada *hidden layer 2* ($z2_j$, k = 1,…, q) jumlahkan hasil perubahan masukannya (dari unit-unit lapisan diatasnya),

$$\delta\_in2_k = \sum_{l=1}^{m} \delta_l w_{kl} \quad (2.13)$$

Kalikan dengan turuan fungsi aktivasinya untuk menghitung informasi kesalahannya,

$$\delta 2_k = \delta\_in2_k f'(z\_in2_k) \quad (2.14)$$

Hitung koreksi bobotnya (digunakan untuk memperbaharui v_{oj} nanti,

Langkah 9: Tiap unit keluaran ($z2_j$, j = 1,…, p) perbaharui bias dan bobotya (j =0,..., n) dengan rumus perbaharui bobot dengan momentum,

$$\Delta v2_{jk} = \mu \delta 2_k z2_j + \alpha\left[v2_{jk}(t) - v2_{jk}(t-1)\right] \quad (2.15)$$

Hitung koreksi bias (digunakan untuk memperbaharui w_{0k} nantinya).

$$\Delta v2_{0k} = \mu\delta2_k z2_j + \alpha[w_{0k}(t) - w_{0k}(t-1)] \quad (2.16)$$

Langkah 10: Setiap unit pada *hidden layer 1* ($z1_j$, k = 1,…, q) jumlahkan hasil perubahan masukannya (dari unit-unit lapisan diatasnya),

$$\delta\_in2_k = \sum_{l=1}^{m} \delta_l w_{kl} \quad (2.17)$$

Kalikan dengan turuan fungsi aktivasinya untuk menghitung informasi kesalahannya,

$$\delta2_j = \delta\_in2_j f'(z\_in2_j) \quad (2.18)$$

Hitung koreksi bobotnya (digunakan untuk memperbaharui v_{oj} nanti,

Langkah 11: Setiap unit pada *hidden layer 1*($z1_j$, j = 1,…, p) jumlahkan hasil perubahan masukannya (dari unit-unit lapisan diatasnya),

$$\delta\_in1_j = \sum_{j=1}^{q} \delta2_j w_{jk} \quad (2.19)$$

Kalikan dengan turuan fungsi aktivasinya untuk menghitung informasi kesalahannya,

$$\delta1_j = \delta\_in1_j f'(z\_in1_j) \quad (2.20)$$

Hitung koreksi bobotnya (digunakan untuk memperbaharui v_{oj} nanti,

Langkah 12: Tiap unit keluaran ($z1_j$, j = 1,…, p) perbaharui bias dan bobotya (i =0,..., n) dengan rumus perbaharui bobot dengan momentum,

$$\Delta v1_{ij} = \mu\delta1_j z1_i + \alpha\left[v1_{ij}(t) - v1_{ij}(t-1)\right] \quad (2.21)$$

Hitung koreksi bias (digunakan untuk memperbaharui w_{0k} nantinya).

$$\Delta v1_{0j} = \mu\delta2_j z2_i + \alpha\left[v1_{0j}(t) - v1_{0j}(t-1)\right] \quad (2.22)$$

Langkah 13: Setiap unit pada *hidden layer 1* ($z1_j$, k = 1,…, p) jumlahkan hasil perubahan masukannya (dari unit-unit lapisan diatasnya),

$$\delta\_in1_j = \sum_{k=1}^{q} \delta_k w_{jk} \quad (2.23)$$

Kalikan dengan turuan fungsi aktivasinya untuk menghitung informasi kesalahannya,

$$\delta1_j = \delta\_in1_j\left(z\_in1_j\right) \quad (2.24)$$

Hitung koreksi bobotnya (digunakan untuk memperbaharui v_{oj} nanti,

Langkah 14: Untuk setiap unit keluaran (Y_l, l = 1, … ,m) perbaharui bias dan bobotnya (k=0, … , p);

$$w_{kl}(baru) = w_{kl}(lama) + \Delta w_{kl} \quad (2.25)$$

Untuk setiap unit pada *hidden layer 2* ($Z2_k$, k= 1, … ,q) perbaharui bias dan bobotnya (j=0, … ,n):

$$v2_{jk}(baru) = v2_{jk}(lama) + \Delta v2_{jk} \quad (2.26)$$

Untuk setiap unit pada *hidden layer 1* ($Z1_j$, j= 1, … ,q) perbaharui bias dan bobotnya (i=0, … ,n):

$$v2_{ij}(baru) = v2_{ij}(lama) + \Delta v2_{ij} \qquad (2.27)$$

Langkah 15: Uji kondisi berhenti.

2.3. Odometry

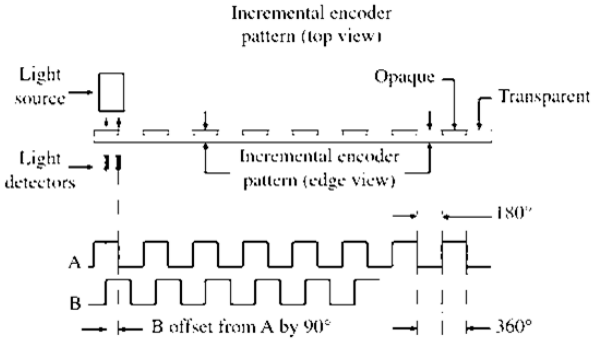
Odometry dapat dikatakan sebagai suatu sistem pengukuran yang menggunakan data-data dari aktuator untuk memperkirakan pergerakan dari robot [10]. Pada robot beroda, *odometry* gerakan translasi robot dapat dilakukan menggunakan putaran dari roda yang posisinya sudah diketahui secara pasti. Sedangkan untuk *odometry* rotasi robot dapat dilakukan menggunakan sensor *gyroscope* yang posisi awalnya sudah diketahui.

2.3.1. Rotary Encoder

Perpindahan linier dapat diukur dan dikomunikasikan oleh alat yang bernama encoder tanpa menggunakan pengubahan dari sinyal analog menjadi digital karena sinyal keluaran yang dihasilkan sudah dalam bentuk digital. Encoder dibagi menjadi encoder *incremental* dan absolut yang mencakup beberapa teknik pendeteksian, salah satunya adalah jenis pendeteksian menggunakan optik.

Encoder optik menggunakan komponen optoelektronik *transmitter/receiver*. Bagian *transmitter* berupa sumber cahaya, biasanya berupa satu atau lebih LED. Bagian *receiver* biasanya terdiri dari dua atau lebih fototransistor [7]. Gambar 2.11 menunjukkan bagian-bagian dari *encoder* optik.

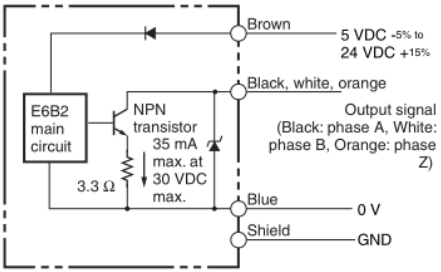
Rotary encoder adalah sensor yang mendeteksi posisi dan kecepatan dengan mengubah perpindahan mekanik menjadi sinyal elektrik dan memroses sinyal tersebut.



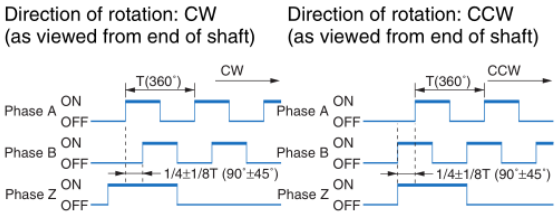
Gambar 2. 11 *Incremental Optical Encoder* dengan LED dan Dua Fototransisitor

Rotary encoder OMRON dengan seri E6B2-CWZ6C ditunjukkan oleh Gambar 2.12. Sensor ini memiliki diameter 40 mm dan menghasilkan 200 pulsa setiap putarannya, dengan keluaran *open collector* sehingga membutuhkan rangkaian *pull-up*. Model rangkaian dari *rotary encoder* ini ditunjukkan oleh gambar 2.13 di mana terdapat 3 keluaran *open collector* yang memiliki fase yang berbeda-beda yang ditunjukkan oleh gambar 2.14.

Gambar 2. 12 *Rotary Encoder* E6B2-CWZ6C



Gambar 2. 13 Model Rangkaian E6B2-CWZ6C

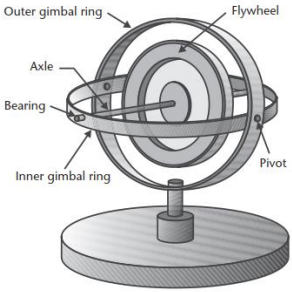


Gambar 2. 14 Fase Keluaran E6B2-CWZ6C

2.3.2. Gyroscope

Gyroscope adalah sensor yang mendeteksi kecepatan sudut. *Gyroscop*e terdiri dari *flywheel* yang terletak pada cincin gimbal. Momentum sudut dari *flywheel* meniadakan torsi eksternal yang diterima dan mempertahankan orientasi dari sumbu putar agar tidak berubah. *Gyroscope* memiliki kepresisian yang berasal dari momentum sudut yang besar, yang proporsional dengan massa berat dari *flywheel*, ukurannya yang besar, dan putaran yang cepat yang diilustrasikan oleh Gambar 2.15.

Salah satu *gyroscope* yang banyak di pasaran adalah MPU-6050 yang tertanam pada modul. Modul sensor InvenSense MPU-6050, yang ditunjukkan oleh Gambar 2.16, memiliki MEMS accelerometer dan MEMS gyroscope pada satu chip. modul ini memiliki keakurasian yang tinggi dengan 16-bit Analog to Digital Converter (ADC) pada setiap kanal. Modul ini menggunkanan I^2C sebagai antarmuka dengan mikrokontroler.



Gambar 2. 15 Ilustrasi Gyroscope

2.3.3. Quaternion

Quaternion dicetuskan pertama kali oleh William Rowan Hamilton pada pertengahan abad ke 19, agar dapat mengaplikasikan bilangan kompleks dalam bentuk tiga dimensi (3D). Quaternion adalah empat buah vektor q = (q_0, q_1, q_2, q_3), yang memiliki sifat operasi perhitungan bilangan seperti bilangan kompleks.

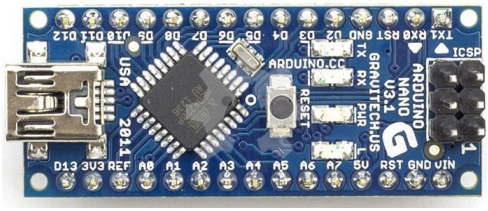
Gambar 2. 16 Modul Gyroscope MPU-6050

2.4. Arduino Nano

Arduino Nano adalah modul mikrokontroler ATmega328 (untuk Arduino Nano versi 3.x) atau ATmega 168 (untuk Arduino versi 2.x) yang berukuran kecil, lengkap dan mendukung penggunaan breadboard. Arduino Nano kurang lebih memiliki fungsi yang sama dengan Arduino Duemilanove, tetapi dalam paket yang berbeda. Arduino Nano tidak memiliki port DC berjenis Barrel Jack, dan terhubung dengan komputer menggunakan port USB Mini-B. Gambar 2.17 menunjukkan gambar Arduino Nano.

2.5. Mikrokontroler STM32F4

STM32F4 merupakan mikrokontroler berbasis ARM 32 bit. Mikrokontroler jenis STM32F4 menggunakan ARM core-M4F yang dapat melakukan pengolahan sinyal digital. STM32F4 memiliki fitur penambahan kecepatan clock yang lebih tinggi dari STM32F2, 64K CCM static RAM, full duplex I^2S, dan memiliki kecepatan konversi ADC. Gambar 2.18 merupakan gambar dari *Board STM32F4-Discovery* yang sudah dilengkapi dengan *downloader* tipe ST-Link untuk memasukkan program dari komputer ke mikrokontroler STM32F4.



Gambar 2. 17 Arduino Nano

Gambar 2. 18 Board STM32F4-Discovery

2.6. Komputer Mini

Komputer *mini* atau *single board computer* adalah kelas komputer *multi-user* yang memiliki bentuk yang lebih kecil daripada komputer personal pada umumnya. Komputer mini memiliki level komputasi yang berada di posisi menengah di bawah kelas komputer mainframe dan sistem komputer *single-user* seperti komputer pribadi. Salah satu bentuk fisik dari computer mini dapat dilihat pada Gambar 2.19.

Spesifikasi atau kemampuan yang dimiliki oleh komputer mini dapat beberapa kali lebih besar jika dibanding dengan personal komputer pada umumnya. Hal ini disebabkan karena *micropocessor* yang digunakan dalam pemrosesan data memiliki kemampuan jauh lebih unggul jika dibanding dengan *micropocessor* yang digunakan pada komputer personal biasa. Intel NUC 6i7-KYK adalah salah satu computer mini keluaran Intel seperti pada gambar 3.2 dengan spesifikasi prosesor Intel® Core™ i7-6770HQ dan RAM 32 GB. Komputer mini ini memiliki konektivitas Thunderbolt™ 3 (40 Gbps), USB 3.1 Gen2 (10 Gbps) melalui konektor USB Tipe C dan Ethernet menggunakan Intel® I219-LM 10/100/1000 Mbps.

Gambar 2. 19 Komputer Mini

2.7. Pustaka OpenFrameworks

OpenFrameworks adalah open source C ++ toolkit yang dirancang untuk membantu proses kreatif dengan menyediakan kerangka intuisi yang sederhana dan intuitif. Logo OpenFrameworks dapat dilihat seperti pada Gambar 2.20.

OpenFrameworks dirancang untuk mempermudah pengguna dalam mengembangkan perangkat lunak. Openframework terdiri dari beberapa *library* yang umum digunakan, termasuk:

- OpenGL, GLEW, GLUT, libtess2 dan cairo untuk grafis
- RtAudio, PortAudio, OpenAL dan Kiss FFT atau FMOD untuk input, output dan analisis audio
- FreeType untuk font
- FreeImage untuk penghematan dan pemuatan gambar
- Quicktime, GStreamer dan videoInput untuk pemutaran video dan grabbing
- Poco untuk berbagai utilitas
- OpenCV untuk visi komputer
- Assimp untuk pemuatan model 3D

Kode ditulis secara masal *cross-compatible*. Saat ini OpenFrameworks mendukung lima sistem operasi (Windows, OSX, Linux, iOS, Android) dan empat IDE (XCode, Code :: Blocks, Visual Studio dan Eclipse). API dirancang untuk menjadi minimal dan mudah dipahami. OpenFrameworks didistribusikan di bawah Lisensi MIT.

Gambar 2. 20 Logo OpenFrameworks

2.8. Microsoft Visual Studio 2015

Microsoft Visual Studio merupakan *integrated development environtment* (IDE) dari Microsoft. IDE ini dapat digunakan untuk mengembangkan aplikasi antarmuka pengguna konsol dan grafis beserta aplikasi Windows Forms, situs web, aplikasi web, dan layanan web di kedua kode asli beserta kode terkelola untuk semua *platform* yang didukung oleh Microsoft Windows, Windows Phone, Windows CE,. NET Framework, .NET Compact Framework dan Microsoft Silverlight. Logo visual studio dapat dilihat seperti pada Gambar 2.21.

Gambar 2. 21 Logo Visual Studio 2015

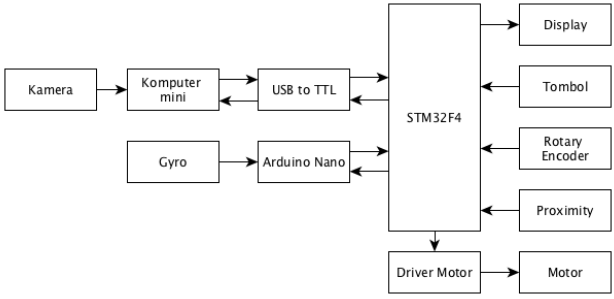
Halaman ini sengaja dikosongkan

BAB III

PERANCANGAN SISTEM

Pada bab ini akan dibahas mengenai perancangan sistem, mulai dari perancangan perangkat keras yang meliputi perancangan pembuatan sistem *odometry* menggunakan *rotary encoder*, kompas digital dan kamera, hingga perancangan perangkat lunak yang meliputi perancangan pengambilan serta pengolahan data sensor *rotary encoder*, kompas digital pada mikrokontroler dan kamera pada komputer mini. Dalam bab ini juga dijelaskan mengenai platform robot sepak bola beroda yang digunakan.

Gambar 3.1 menunjukkan sistem perangkat keras Robot Sepak Bola Beroda yang digunakan. Masukan dari mikrokontroler, dalam hal ini STM32F4, yang berupa data keluaran sensor dibaca oleh mikrokontroler dan nilainya dikirim pada komputer mini menggunakan komunikasi serial. komputer mini juga memperoleh masukan berupa citra dari kamera. Dari data-data yang diterima oleh komputer mini ini, dilakukan suatu proses untuk mendapatkan parameter-parameter kontrol yang nilainya kemudian dikirimkan pada mikrokontroler untuk dieksekusi melalui driver motor. Pada pembacaan sensor kompas digital, digunakan Arduino Nano sebagai *slave* karena keterbatasan pin input output dari STM32F4.

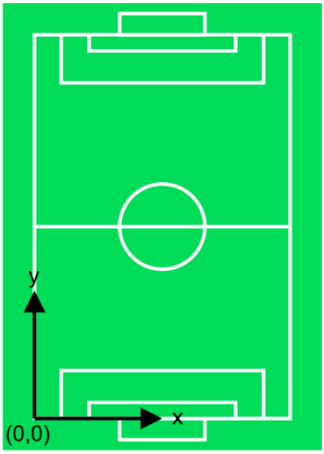


Gambar 3. 1 Diagram Blok Perangkat Keras Robot Sepak Bola Beroda

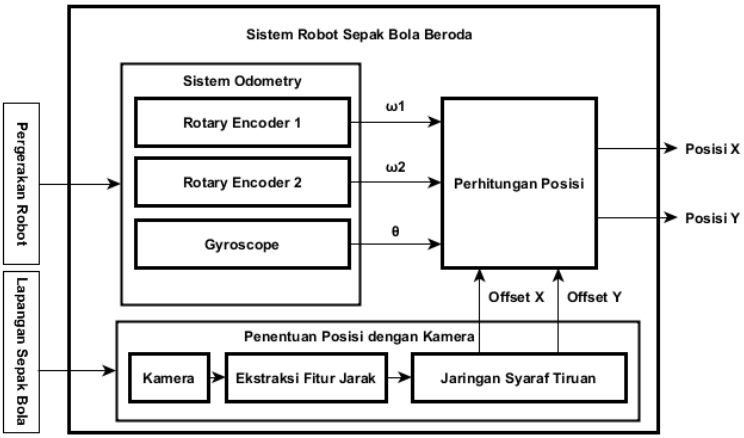
Pada tugas akhir ini, posisi robot pada robot sepak bola beroda didefinisikan sebagai koordinat x dan y dari robot terhadap titik origin (0,0) seperti yang diilustrasikan pada Gambar 3.2. Penentuan posisi robot menggunakan dua buah *rotary encoder* dan *gyroscope* di mana pembacaan dari kedua jenis sensor tersebut dilakukan pada mikrokontroler yang nilainya kemudian dikirimkan pada komputer mini untuk dilakukan operasi vektor dari ketiga nilai tersebut.

3.1. Perancangan Sistem Penentuan Posisi

Diagram blok dari sistem penentuan posisi robot sepak bola beroda menggunakan *rotary encoder* dan kamera ditunjukkan oleh Gambar 3.3. Penentuan posisi robot dilakukan menggunakan *rotary encoder* yang dikombinasikan dengan *gyroscope*. Pada saat terjadi goal, dilakukan pembaharuan nilai dari posisi robot agar dapat mengurangi kesalahan akibat integral *error* dari *rotary encoder* dan *gyroscope*. Pembaharuan dilakukan berdasarkan hasil penentuan posisi robot menggunakan kamera yang menggunakan jaringan syaraf tiruan.



Gambar 3. 2 Ilustrasi Posisi Robot Sepak Bola Beroda



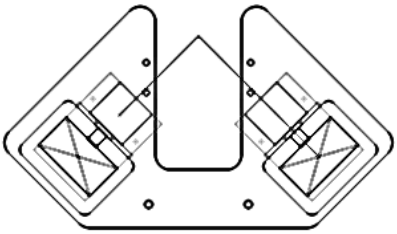
Gambar 3. 3 Diagram Blok Sistem Robot Sepak Bola Beroda

3.2. Konfigurasi Sistem Odometry

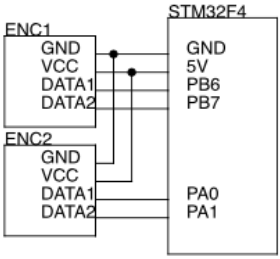
Sistem odometri yang dirancang menggunakan dua buah *rotary encoder* keluaran OMRON dengan seri E6B2-CWZ6C, yang menghasilkan 200 pulsa setiap satu putaran, dan modul sensor *gyroscope*l keluaran INVENSENSE dengan seri MPU6050.

3.2.1. *Konfigurasi* Rotary Encoder

Rotary E6B2-CWZ6C dipasang pada roda omni tersendiri yang terpisah dari roda penggerak dengan posisi seperti pada Gambar 3.6.



Gambar 3. 4 Perancangan Posisi Roda Omni *Rotary Encoder*



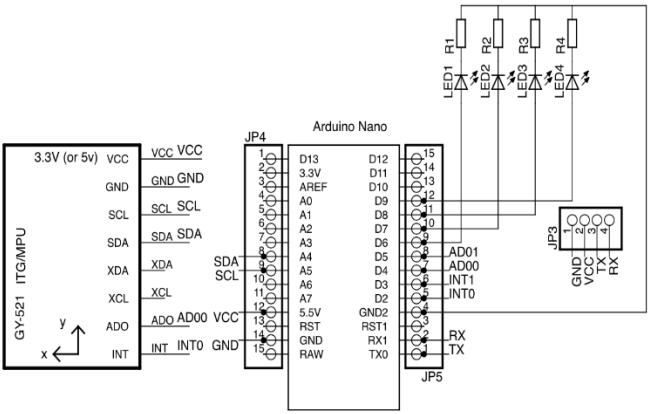
Gambar 3. 5 Interkoneksi *Rotary Encoder* dengan STM32F4

Dengan pemasangan *rotary encoder* seperti pada gambar 3.4, di mana jarak *rotary encoder* dengan pusat robot adalah sama, dengan selisih sudut 90^o, di mana yang satu berada pada posisi 135^o dan yang ke dua berada posisi 225^o.

Rotary encoder E6B2-CWZ6C memiliki 3 keluaran *open collector* dan menghasilkan 200 pulsa setiap putaran. Untuk mengurangi besarnya kesalahan pembacaan pulsa, digunakan dua buah keluaran dari *rotary encoder*. Keluaran yang digunakan adalah fase A dan fase B dengan konfigurasi yang ditunjukkan oleh gambar 3.5, di mana Pa dan Pb adalah port A dan port B STM32F4.

3.2.2. Interkoneksi Gyroscope

Penggunaan dua buah *rotary encoder* pada penentuan posisi robot akan mengalami kesalahan apabila arah muka robot tidak berada pada posisi 0^0 atau mengarah ke depan, oleh karena itu, digunakan sensor gyroscope sebagai kompas digital agar dihasilkan penentuan posisi yang tepat pada segala arah. Pada tugas akhir ini, digunakan modul sensor gyroscope MPU-6050 yang diakses oleh Arduino Nano yang kemudian dikirm secara serial kepada STM32F4. Gambar 3.8 menunjukkan konfigurasi modul sensor gyroscope MPU-6050 dengan Arduino Nano di mana Pd9 dan Pd8 adalah port serial dari STM32F4.



Gambar 3. 6 Interkoneksi Modul MPU-6050 dengan Arduino Nano

3.3. Konfigurasi Kamera

Dalam tugas akhir ini, kamera berfungsi sebagai pendeteksi fitur-fitur citra berupa jarak piksel robot terhadap marka lapangan. Kamera yang digunakan adalah kamera Logitech C922. Pada bagian depan kamera ini, dipasang lensa *fish eye* 235^0, yang diposisikan pada bagian kepala robot dengan arah menghadap ke bawah dengan pelindung berupa tabung yang terbuat dari bahan akrilik. Pemasangan kamera ini dapat dilihat pada Gambar 3.7. Kamera ini diakses menggunakan komputer mini Intel NUC 6i7-KYK melalui USB.

Gambar 3. 7 Pemasangan Kamera pada Kepala Robot

3.4. Penentuan Posisi Robot Menggunakan *Rotary Encoder* dan *Gyroscope*

3.4.1. Pengolahan Data Gyroscope

Kompas digital yang digunakan memanfaatkan *gyroscope* sumbu *yaw* pada modul MPU-6050. Program yang dikembangkan menggunakan library *open source* yang memanfaatkan fitur yang sudah terdapat pada modul ini, yaitu *Digital Motion Processing* (DMP) yang memiliki fitur algoritma pemrosesan gerakan 3 dimensi dan pengenalan gerakan isyarat.

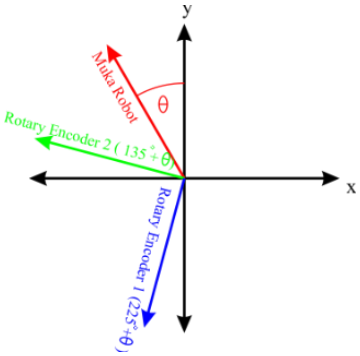
Output dari DMP yang digunakan adalah output berupa data quaternion dari posisi *gyroscope*. Data quaternion ini kemudian diubah menjadi data sudut *yaw* dari *gyroscope* dan diperoleh sudut θ dari arah robot menggunakan rumus sebagai berikut:

$$\theta = \tan^{-1}\left(\frac{2(q_1 q_2 - q_0 q_3)}{2(q_0^2 + q_1^2) - 1}\right) \quad (3.1)$$

Data yang diperoleh ini kemudian dikirimkan kepada STM32F4 melalui komunikasi serial untuk diolah dengan data dari *rotary encoder*.

3.4.2. *Pengolahan Data* Rotary Encoder *dan* Gyroscope

Rotary encoder yang digunakan sejumlah 2 buah yang dipasang dengan selisih sudut 90^o dengan posisi terhadap muka robot 135^o dan 225^o.Gambar 3.8 menunjukkan ilustrasi vektor dari putaran *rotary encoder* 1 dan 2 serta arah robot yang diperoleh dari kompas digital.



Gambar 3. 8 Vektor *Rotary Encoder* dan *Gyroscope*

Berdasarkan arah robot dan kecepatan putar dari kedua *rotary encoder*, penentuan kecepatan pada arah x dan y dari robot memiliki persamaan berikut:

$$\begin{bmatrix}\dot{x}\\ \dot{y}\end{bmatrix} = \begin{bmatrix}-\cos(225° + \theta) & -\cos(135° + \theta)\\ -\sin(225° + \theta) & -\sin(135° + \theta)\end{bmatrix}\begin{bmatrix}\omega_1 R_1\\ \omega_2 R_2\end{bmatrix} \quad (3.2)$$

Dengan θ adalah sudut yang diperoleh dari sensor *gyroscope*, ω_1 adalah kecepatan sudut *rotary encoder* 1 dan R_1 adalah jari-jari *rotary encoder* 1. Kecepatan sudut masing-masing *rotary encoder* diperoleh dari *counter* STM32F4 yang diperbaharui setiap 1 ms. Nilai yang diperoleh dari persamaan tersebut adalah kecepatan arah x dan y dari robot. Berikut adalah implementasi persamaan tersebut menggunakan bahasa pemrograman C++ sehingga diperoleh kecepatan arah x dan y dari robot:

```
buff_vx[0]= -(float)w_odo[0]*cosf(3.9269908169+theta)*R1;

buff_vx[1]= -(float)w_odo[1]*cosf(2.3561944901+theta)*R2;

buff_vy[0]= -(float)w_odo[0]*sinf(3.9269908169+theta)*R1;

buff_vy[1]= -(float)w_odo[1]*sinf(2.3561944901+theta)*R2;

vx = buff_vx[0] + buff_vx[1];
vy = buff_vy[0] + buff_vy[1];
```

Posisi x dan y dari robot diperoleh melalui proses integral kecepatan arah x dan y dengan persamaan sebagai berikut:

$$x = \sum x \qquad (3.4)$$

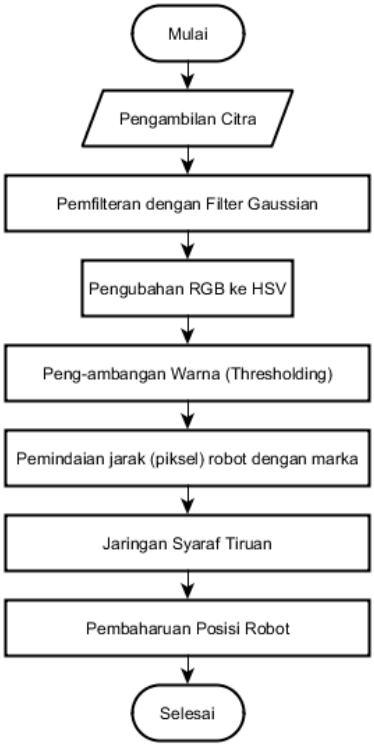
$$y = \sum y \qquad (3.5)$$

Di mana x adalah posisi arah x dan y adalah posisi arah y robot yang diperoleh dari hasil integral kecepatan arah x dan arah y, sehingga diimplementasikan pada program sebagai berikut:

```
robot_x += vx;
robot_y += vy;
```

3.6. Penentuan Posisi Robot Menggunakan Kamera

Penentuan posisi robot menggunakan kamera dilakukan menggunakan fitur jarak setiap sudut robot terhadap marka lapangan, yang dihasilkan dari pengolahan citra dari kamera, yang masuk pada jaringan syaraf tiruan. Pelatihan dari jaringan syaraf tiruan dilakukan secara *offline* dan menggunakan *supervisory training* sehingga dihasilkan bobot-bobot dan bias untuk dapat menentukan posisi robot. Gambar 3.9 menunjukkan diagram alir sistem penentuan posisi robot menggunakan kamera.



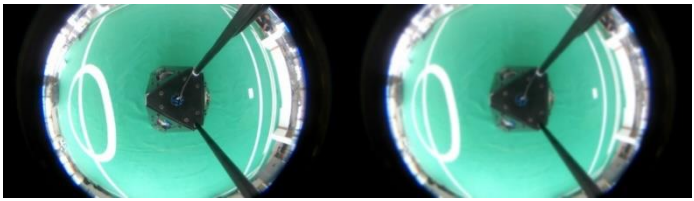
Gambar 3. 9 Diagram Alir Sistem Penentuan Posisi Robot Menggunakan Kamera

3.6.1. Filter

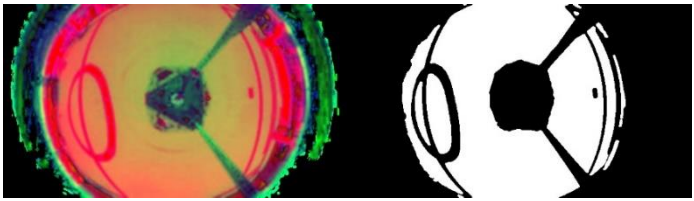
Pada pengambilan citra menggunakan kamera, selalu terdapat derau yang dapat mengganggu pemrosesan selanjutnya. Derau ini memiliki frekuensi yang tinggi, sehingga untuk mengurangi atau menghilangkannya, diperlukan filter lolos-bawah. Pada implementasinya, filter yang digunakan adalah filter *Gaussian*. Gambar 3.14 menunjukkan contoh hasil penggunaan filter gaussian pada sistem ini.

3.6.2. Peng-ambangan Warna

Lapangan sepak bola memiliki warna khusus, yaitu warna hijau dengan tanda-tanda lapangan yang berwarna putih. Berdasarkan keunikan tersebut, pengambangan warna yang digunakan memisahkan warna hijau dengan yang warna lain. Pada ruang warna hsv, warna hijau memiliki nilai hue 60, oleh karena itu, peng-ambangan warna dilakukan dengan batas hue mulai dari 60 sampai dengan 90. Jangkauan nilai *saturation* dan *value* perlu diubah-ubah tergantung dari kondisi pencahayaan pada lapangan. Gambar 3.11 menunjukkan contoh peng-ambangan warna pada sistem ini.



Gambar 3. 10 Citra Sebelum dan Sesudah Dilakukan Proses Filter



Gambar 3. 11 Citra HSV dan Citra Biner dengan Pengambangan

3.6.3. Ekstraksi Fitur Berupa Jarak Robot Terhadap Marka Lapangan

Pada [10], pendeteksian halangan dapat dilakukan dengan memindai jarak piksel berwarna hitam pada setiap sudut dari robot. dengan menggunakan penalaran yang serupa, pendeteksian jarak dengan marka lapangan dapat dilakukan dengan memindai piksel hitam pada setiap sudut di sekeliling robot dari citra yang sudah dilakukan peng-ambangan warna. Proses pemindaian dilakukan dengan implementasi yang menggunakan bahasa C++ sebagai berikut:

```
void find_obstacle (float* _object_distance, float*
_object_tetha){
for (int tetha = 0; tetha < 360; tetha += 10){
object = false; int temp = 0;
for (int i = radius_circle; i < radius_circle2 -
(thickness_circle2 / 2); i++){
int x = obstacle.cols / 2 - i * cos(tetha * CV_PI / 180);
int y = obstacle.rows / 2 + i * sin(tetha * CV_PI / 180);
if (x < obstacle.cols && x > 0 && y < obstacle.rows && y
> 0){
if (obstacle.at<unsigned char>(y, x) == 255){ object =
true;
cv::circle(frame, Point(x, y), 5, Scalar(255, 0, 0));
_object_distance[tetha / 10] = i;
_object_tetha[tetha / 10] = tetha; break;}
temp = i;
cv::line(frame, Point(x, y), Point(obstacle.cols / 2 -
radius_circle * cos(tetha * CV_PI / 180), obstacle.rows /
2 + radius_circle * sin(tetha * CV_PI / 180)),
Scalar(255, 0, 255));}}
```

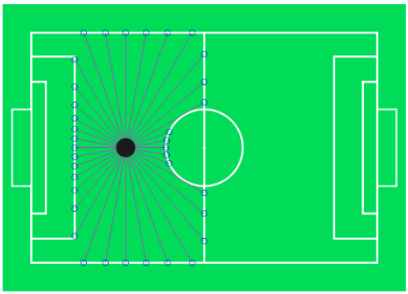
Gambar 3.12 menunjukkan ilustrasi pemindaian dengan interval sudut 10^o. Garis magenta merepresentasikan masukan untuk jaringan syaraf tiruan, dan lingkaran biru merepresentasikan titik di mana marka lapangan terdeteksi.

3.6.4. Jaringan Syaraf Tiruan

Jaringan syaraf tiruan yang digunakan memiliki 4 lapisan, input layer, 2 hidden layer dan output layer. Algoritma pelatihan yang digunakan menggunakan algoritma *backpropagation error*. Fungsi aktifasi yang digunakan adalah fungsi sigmoid pada hidden layer 1 serta hidden layer 2 dan linier pada output layer. Jumlah node pada hidden layer pertama sejumlah 100 dan hidden layer ke dua 20.

Prosedur pertama pada jaringan syaraf tiruan adalah memberikan nilai acak pada bobot-bobot setiap lapisan, dengan nilai rata-rata 0 dan deviasi 0,1. Pada Visual C++, terdapat fungsi yang menghasilkan nilai integer acak dengan nilai maksimal tertentu, yaitu fungsi rand(). Agar menghasilkan nilai rata-rata 0 dan memiliki deviasi 0,1, digunakan persamaan dalam bahasa pemrograman C++ sebagai berikut:

```
w[i][j]=(double)rand() / (10 * RAND_MAX) - 0.05;
```



Gambar 3. 12 Pemindaian Jarak Marka pada Setiap Sudut

Prosedur selanjutnya adalah prosedur *feed forward*, di mana setiap masukan dirambatkan maju menuju *hidden layer*, dan keluaran dari *hidden layer* juga dirambatkan maju hingga mencapai lapisan keluaran. Implementasi dari *feed forward* pada bahasa pemrograman C++ adalah sebagai berikut:

```
double backPro::aggregate(double* x, double* w, int
sizeofx){
double temp=0;
for(int i=1;i<=sizeofx;i++){ temp+=x[i]*w[i];}
temp+=w[0];
return temp;
}
void backPro::feedForward(){
    for(int i=1; i<= node1; i++){
        zin1[i]=agregate(xin,win[i],inLayer);
        z1[i]=sigmoid(zin1[i]);
    }
    for(int i=1; i<= node2; i++){
        zin2[i]=agregate(z1,whid[i],node1);
        z2[i]=sigmoid(zin2[i]);
    }
    for(int i=1; i<= outLayer; i++){
        yin[i]=agregate(z2,wout[i],node2);
       y[i] = yin[i];
    }
}
```

Pada *hidden layer*, hasil penjumlahan pada fungsi aggregate() diaktifasi menggunakan fungsi log sigmoid, sedangkan pada lapisan keluaran, digunakan fungsi aktifasi linier. Fungsi aktifasi log simoid diimplementasikan sebagai berikut:

```
double backPro::sigmoid(double v){
   return 1/(1+exp(-v));
}
```

Prosedur selanjutnya adalah prosedur *back propagation*, di mana dari hasil *back propagation* ini, nilai dari setiap bobot akan diperbaharui berdasarkan besarnya *error* yang diperoleh dari prosedur *feed forward*. Hal pertama yang dilakukan adalah dengan menghitung *error* yang dihasilkan, yaitu δ_k:

```
double backPro::dk(double des, double res, double
v){
return (des - res);
```

Berdasarkan δ_k yang diperoleh, kemudian dihitung koreksi bobot dan bias pada lapisan keluaran, yang digunakan untuk memperbaharui nilai bobot pada lapisan keluaran.

```
double backPro::dwk(double del, double x){
    return u*del*x;
}
```

```
for(int i=1;i<= outLayer;i++){
        d[i]=dk(t[i],y[i],yin[i]);
    }
for(int i=1; i<=node2;i++){
        for(int j=1; j<=outLayer;j++){
           dwout[i][j]=dwk(d[j],z2[i]);
           dwout[0][j]=dwk(d[j],1);
        }
}
```

Pada *hidden layer*, informasi *error* diperoleh dari hasil penjumlahan perubahan masukannya, yang kemudian dikalikan dengan turunan fungsi aktifasinya sebagai berikut:

```
double backPro::dhid(double* delk, double* wk, double
zin){
   double temp =0;
   for(int i=1;i<=outLayer;i++){
       temp+=d[i]*wk[i];
   }
   temp=temp*sigmoidDiff(zin);
   return temp;

}
```

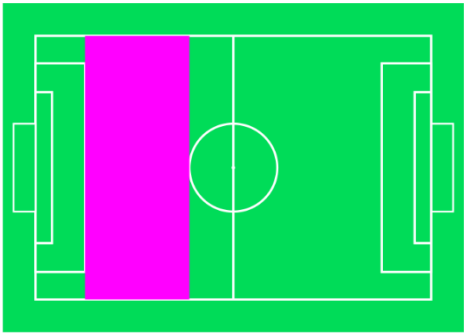
```
for(int i=1;i<=node1;i++){
    dh1[i]=dhid(dh2,temp2[i],zin1[i]);
}

for(int i=1;i<=inLayer;i++){
    for(int j=1; j<=node1;j++){
          dwin[i][j]=dwk(dh1[j],xin[i]);
          dwin[0][j]=dwk(dh1[j],1);
     }
}
```

Hasil dari prosedur *back propagation* yang berupa nilai koreksi bobot kemudian digunakan untuk memperbaharui nilai dari setiap bobot. Proses ini dilakukan dengan menjumlahkan nilai dari bobot yang lama dengan besarnya koreksi bobot yang diperoleh, yang diimplementasikan sebagai berikut:

```
void backPro::updateW(){
    for(int i=1;i<=node1;i++){
        for(int j=0;j<=inLayer;j++){
            win[i][j]+=dwin[j][i];
        }
    }
    for(int i=1;i<=node2;i++){
        for(int j=0;j<=node1;j++){
            whid[i][j]+=dwhid[j][i];
        }
    }
    for(int i=1;i<=outLayer;i++){
        for(int j=0;j<=node2;j++){
            wout[i][j]+=dwout[j][i];
        }
    }
}
```

Jaringan syaraf tiruan digunakan untuk memperoleh posisi robot terhadap lapangan di mana nilai pembacaannya tidak terpengaruh oleh pembacaan sebelumnya, sehingga dapat digunakan untuk memperbaharui posisi robot. Pada tugas akhir ini, pembaharuan dilakukan pada daerah pada Gambar 3.17 dengan warna magenta.



Gambar 3. 13 Daerah yang Dijangkau oleh Jaringan Syaraf Tiruan

3.6.5. Pembaharuan Posisi Robot

Pembaharuan posisi robot dilakukan dengan menambahkan nilai *offset* pada perhitungan posisi robot. Nilai *offset* baru diperoleh dari penjumlahan nilai offset lama dengan posisi yang diperoleh dari jaringan syaraf tiruan, dengan implementasi pada program adalah sebagai berikut:

```
offset_robot_x += robot_x – y[1];

offset_robot_y += robot_y – y[2];
```

BAB IV

PENGUJIAN DAN ANALISA

Pada bab ini, dibahas mengenai pengujian dari sistem yang telah dikembangkan. Bab ini bertujuan untuk mengetahui apakah permasalahan dalam perancangan sistem pada tugas akhir ini telah terselesaikan atau tidak. Pengujian pada bab ini terdiri dari pengujian *odometry*, berupa pengujian penentuan posisi menggunakan *rotary encoder* dan *gyroscope*, serta pengujian penentuan posisi menggunakan kamera dan hasil jaringan syaraf tiruan.

4.1. Pengujian Odometry

4.1.1. Pengujian Gyroscope

Pengujian sensor *gyroscope* dilakukan untuk mengetahui di mana robot sedang menghadap. Pengujian dilakukan dengan memutar robot pada sudut tertentu yang telah diketahui nilainya kemudian mengamati nilai keluaran dari sensor. Hasil pengujian *gyroscope* ini disajikan pada Tabel 4.1. Dari data yang diperoleh, dihasilkan nilai *Root Mean Square Error* (RMSE) pada pengujian *gyroscope* sebesar 1,34^{o}.

4.1.2. *Pengujian* Rotary Encoder *dan* Gyroscope Origin to Point

Pengujian dilakukan dengan mengubah posisi robot dari posisi origin (0,0) menuju ke titik tertentu hingga dihasilkan data pada Tabel 4.2. Dari data yang diperoleh, dihasilkan nilai RMSE sebesar 27,39 cm pada sumbu x dan 32,53 cm pada sumbu y.

4.1.3. *Pengujian* Rotary Encoder *dan* Gyroscope Point to Point

Pengujian dilakukan dengan mengubah posisi robot dari posisi origin (0,0) menuju ke titik tertentu, dan dilanjutkan ke titik-titk lain hingga dihasilkan data pada Tabel 4.3. Dari data yang diperoleh, dihasilkan nilai RMSE sebesar 5,68 cm pada sumbu x dan 16,51 cm pada sumbu y.

Tabel 4. 1 Data Pengujan Kompas Digital

No	Sudut (º)	Sudut Terukur (º)	Error (º)
1	0	0,8	0,8
2	10	11,7	1,7
3	20	20,5	0,5
4	30	30,7	0,7
5	40	42,1	2,1
6	50	52,3	2,3
7	60	61,5	1,5
8	70	72,6	2,6
9	80	80,7	0,7
10	90	90,4	0,4
11	100	100,4	0,4
12	110	110,2	0,2
13	120	120,3	0,3
14	130	131,6	1,6
15	140	142,2	2,2
16	150	151,1	1,1
17	160	160,5	0,5
18	170	171,1	1,1
19	180	180,2	0,2
RMSE (º)			1,34

4.2. Pengujian Pengolahan Citra

Pengujian dilakukan dengan mengatur batas-batas nilai ambang yang digunakan untuk memperoleh citra biner yang memiliki warna putih pada citra RGB warna hijau dan warna hitam pada citra RGB warna selain hijau. Nilai batas ambang ini perlu diatur pada setiap lingkungan yang

berbeda. Gambar 4.1 menunjukkan citra asli, Gambar 4.2 menunjukkan citra hasil filter Gaussian, Gambar 4.3 menunjukkan citra hasil pengubahan RGB ke HSV dan Gambar 4.4 menunjukkan citra hasil pengambangan warna.

Tabel 4. 2 Data Pengujian Origin to Point

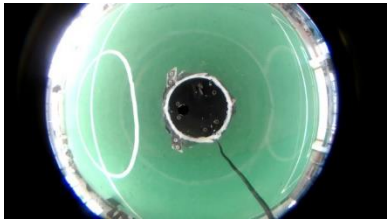
No	Posisi		Posisi Terukur		Error	
	x (cm)	y (cm)	x (cm)	y (cm)	x (cm)	y (cm)
1	900	0	896	-13	-4	-13
2	450	780	405	804	-45	24
3	780	450	790	508	10	58
4	900	0	910	55	10	55
5	780	-450	799	-471	-19	-21
6	450	-780	493	-746	-43	34
7	0	-900	26	-882	26	18
8	-450	-780	-467	-797	-17	-17
9	-780	-450	-800	-475	-20	-25
10	-900	0	-909	-48	-9	-48
11	-780	450	-810	440	-30	-10
12	-450	780	-497	762	-47	-18
RMSE (cm)					27,39	32,53

Tabel 4. 3 Data Pengujian Point to Point

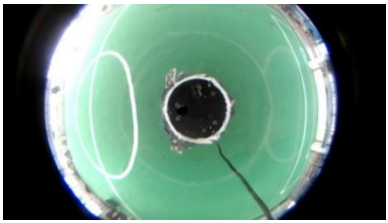
No	Posisi		Posisi Terukur		Error	
	x (cm)	y (cm)	x (cm)	y (cm)	x (cm)	y (cm)
1	100	150	99	154	-1	4
2	300	150	304	166	4	16
3	500	150	512	179	12	29
4	500	350	500	374	0	24
5	300	350	293	362	-7	12
6	100	350	100	357	0	7
7	100	150	96	155	-4	5
RMSE (cm)					5,68	16,51

4.3. Pengujian Penentuan Posisi Robot dengan Kamera

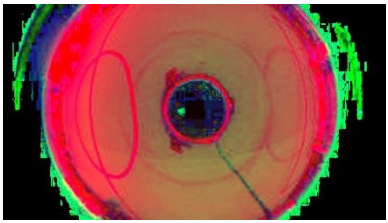
Pengujian dilakukan dengan menempatkan robot pada titik-titk di lapangan yang posisi sesungguhnya sudah diketahui, yang kemudian dilakukan proses perolehan fitur dan dihasilkan keluaran jaringan syaraf tiruan berupa posisi x dan y dari robot terhadap lapangan. Gambar 4.5 menunjukkan citra yang diperoleh dari proses perolehan fitur. Garis warna magenta merepresentasikan masukan untuk jaringan syaraf tiruan, yang mewakili setiap sudut, dan lingkaran biru merepresentasikan titik di mana marka lapangan terdeteksi.



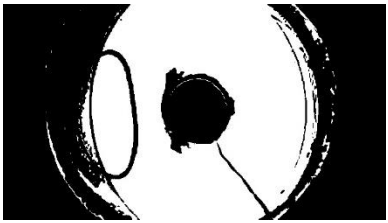
Gambar 4. 1 Citra Asli



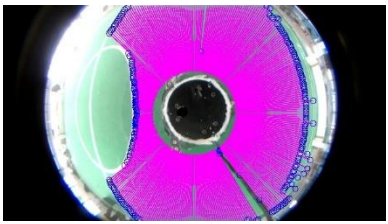
Gambar 4. 2 Citra Hasil Filter Gaussian



Gambar 4. 3 Citra Hasil Pengubahan RGB ke HSV



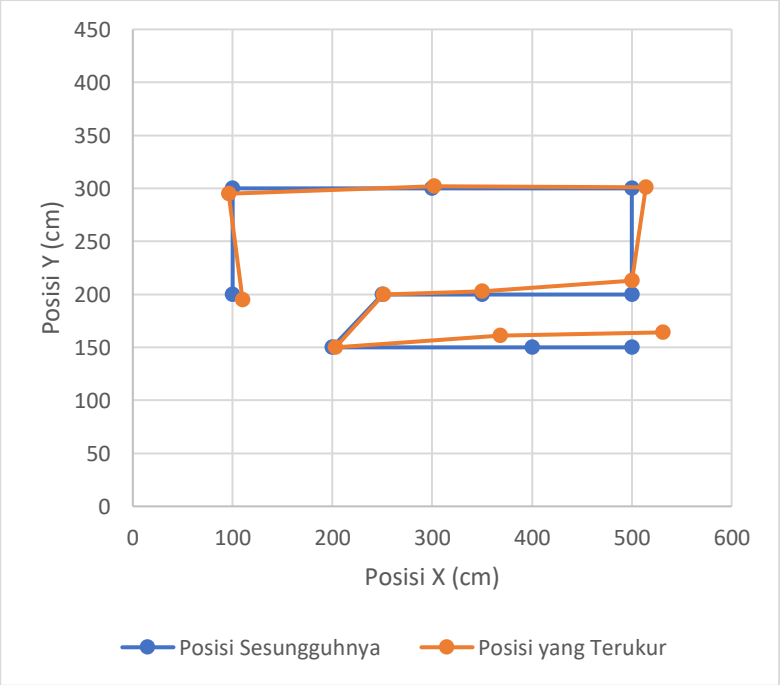
Gambar 4. 4 Citra Hasil Peng-ambangan Warna



Gambar 4. 5 Citra Hasil Proses Perolehan Fitur

Nilai-nilai fitur yang berjumlah 360 tersebut kemudian menjadi masukan jaringan syaraf tiruan. Gambar 4.6 menunjukkan hasil pengujian pada beberapa titik.

Pengujian dilakukan dengan menggeser posisi robot sejauh 50 cm dengan data terlampir dan diperoleh *Root Mean Square Error* (RMSE) sebesar 20,39 cm pada sumbu x dan 10,09 cm pada sumbu y. Nilai ini masih dapat ditoleransi apabila dibandingkan dengan ukuran robot, yaitu 50 x 50 cm^2. Berdasarkan pengamatan, penentuan posisi robot dengan kamera ini memiliki respon yang lambat.



Gambar 4. 6 Pengujian Penentuan Posisi Robot dengan Kamera pada Beberapa Titik

4.4. Perbandingan Pengujian

Perbandingan pengujian antara hasil penentuan posisi menggunakan *rotary encoder* dan *gyroscope* dengan kamera ditunjukkan pada Tabel 4.4 di mana nilai *error* dihitung berdasarkan nilai *Root Mean Square Error* (RMSE) dari *error* arah x dan y.

Berdasarkan data tersebut, dapat diketahui bahwa nilai RMSE dari kamera, yaitu sebesar 16,24 cm pada sumbu x dan 10,50 cm pada sumbu y, lebih kecil daripada nilai RMSE dari *rotary encoder* dan *Gyroscope*, yaitu sebesar 44,60 cm pada sumbu x dan 17,56 cm pada sumbu y. Oleh karena itu, penggunaan kamera untuk pembaharuan posisi robot dapat meminimalisir *error* dari hasil penentuan posisi menggunakan *rotary encoder* dan *gyroscope*. Pembaruan posisi dilakukan dengan mengubah nilai *offset* posisi x dan y pada robot berdasarkan keluaran jaringan syaraf tiruan dalam periode tertentu. Pembaharuan ini menghasilkan posisi x dan y robot sama dengan keluaran jaringan syaraf tiruan.

Tabel 4. 4 Perbandingan Penentuan Posisi menggunakan *Rotary Encoder* dengan Kamera

No	Posisi		Posisi Terukur				Error			
			RE dan Gyro		Kamera		RE dan Gyro		Kamera	
	x (cm)	y (cm)	x (cm)	y (cm)	x (cm)	y (cm)	x (cm)	y (cm)	x (cm)	y (cm)
1	100	150	99	157	105	151	-1	7	5	1
2	150	150	150	159	149	149	0	9	-1	-1
3	200	150	201	161	203	150	1	11	3	0
4	250	150	253	163	250	156	3	13	0	6
5	300	150	305	166	273	155	5	16	-27	5
6	350	150	358	168	355	168	8	18	5	18
7	400	150	412	170	368	161	12	20	-32	11
8	450	150	466	173	410	168	16	23	-40	18
9	500	150	521	176	531	164	21	26	31	14
10	550	150	577	179	541	185	27	29	-9	35
11	550	200	577	227	541	219	27	27	-9	19
12	500	200	528	225	500	213	28	25	0	13
13	450	200	479	223	467	204	29	23	17	4
14	400	200	431	220	440	206	31	20	40	6
15	350	200	383	218	350	203	33	18	0	3
16	300	200	336	216	310	201	36	16	10	1
17	250	200	290	214	251	200	40	14	1	0
18	200	200	244	211	189	199	44	11	-11	-1
19	150	200	199	209	150	199	49	9	0	-1
20	100	200	155	206	110	199	55	6	10	-1
21	100	250	155	254	98	241	55	4	-2	-9
22	150	250	206	256	150	243	56	6	0	-7
23	200	250	257	258	190	245	57	8	-10	-5
24	250	250	309	260	243	240	59	10	-7	-10
25	300	250	361	263	298	243	61	13	-2	-7
26	350	250	414	265	376	246	64	15	26	-4
27	400	250	468	267	409	245	68	17	9	-5
28	450	250	522	270	454	249	72	20	4	-1
29	500	250	577	273	485	246	77	23	-15	-4
30	550	250	633	276	547	260	83	26	-3	10
RMSE (cm)							44.60	17.56	16.24	10.50

BAB V

KESIMPULAN DAN SARAN

5.1. Kesimpulan

Berdasarkan data hasil pengujian dalam pembuatan tugas akhir ini, dapat ditarik kesimpulan sebagai berikut:

1. Penentuan arah robot menggunakan g*yroscope* memiliki nilai *error* RMS sebesar 1,34^{o}.
2. Pada pengujian *point to* origin dari penentuan posisi robot menggunakan *rotary encoder* dan *gyroscope*, dihasilkan *error* RMS sebesar 27,39 cm pada sumbu x dan 32,53 cm pada sumbu y.
3. Pada pengujian *point to* origin dari penentuan posisi robot menggunakan *rotary encoder* dan *gyroscope*, dihasilkan *error* RMS sebesar 5,68 cm pada sumbu x dan 16,51 cm pada sumbu y.
4. Penentuan posisi robot menggunakan *rotary encoder* dan *gyroscope* memiliki *error* RMS sebesar 44,60 cm pada sumbu x dan 17,56 cm pada sumbu y, sedangkan penentuan posisi robot menggunakan kamera memiliki *error* RMS sebesar 16,24 cm pada sumbu x dan 10,50 cm pada sumbu y, sehingga penentuan posisi robot menggunakan *rotary encoder* dan *gyroscope* dapat diminimalisir menggunakan kamera.

5.2. Saran

Saran yang dapat diberikan oleh penulis untuk pengembangan dari tugas akhir ini adalah sebagai berikut:

1. Pengembangan penentuan posisi robot menggunakan kamera yang dapat memperbaharui arah robot.

Halaman ini sengaja dikosongkan

DAFTAR PUSTAKA

[1] N. L. A. J. R. N. João Silva, "Cooperative detection and identification of obstacles in a robotic," dalam *Advances in Robotics: Modeling, Control and Applications*, Portugal, iConcept Press, 2014, pp. 3 - 5.

[2] R. E. W. Rafael C. Gonzalez, Digital Image Processing Second Edition, New Jersey : Prentice-Hall, Inc., 2002.

[3] M. Y. A., Penerjemahan Bahasa Isyarat Indonesia Menggunakan Kamera pada Telepon Genggam Android, Surabaya: ITS, 2016.

[4] A. S. Abdul Kadir, Teori dan Aplikasi Pengolahan Citra, Yogyakarta: Penerbit Andi, 2013.

[5] J. C. Russ, The Image Processing Handbook Sixth Edition, Boca Raton: CRC Press, 2011.

[6] G. Bradski, "OpenCV Wiki," OpenCV, 28 February 2017. [Online]. Available: https://github.com/opencv/opencv/wiki. [Diakses 15 May 2017].

[7] A. K. Gary Bradski, Learning OpenCV, Sebastopol: O'Reilly Media, 2008.

[8] T. Munakata, Fundamentals of the New Artificial Intelligence - Neural, Evolutionary, Fuzzy and More Second Edition, London: Springer, 2008.

[9] L. V. Fausett, Fundamentals of Neural Network - Architectures, Algorithms, and Applications, New Jersey: Prentice-Hall, 1994.

[10] M. J. Gregory Dudek, "Inertial Sensors, GPS, and Odometry," dalam *Springer Handbook of Robotics*, Berlin - Heidelberg, Springer Berlin Heidelberg, 2008, pp. 477-490.

[11] D. S. Nyce, Linear Position Sensors - Theory and Application, New Jersey: John Wiley & Sons, Inc., 2004.

[12] K. W. Nadim Maluf, An Introduction to Microelectromechanical Systems Engineering Second Edition, Norwood: Artech House, Inc., 2004.

[13] H. P. Halvorsen, *Introduction to Visual Studio and C#,* Notodden: University College of Southeast Norway, 2016.

LAMPIRAN

A. Pengujian Rotary Encoder Arah Depan

No	Posisi		Posisi Terukur		Error	
	X (cm)	Y (cm)	X (cm)	Y (cm)	X (cm)	Y (cm)
1	0	50	0	52	0	2
2	0	100	0	101	0	1
3	0	150	0	151	0	1
4	0	200	0	200	0	0
5	0	250	0	251	0	1
6	0	300	-1	300	-1	0
7	0	350	-2	349	-2	-1
8	0	400	-3	398	-3	-2
9	0	450	-3	448	-3	-2
10	0	500	-4	498	-4	-2
11	0	550	-5	547	-5	-3
12	0	600	-6	597	-6	-3
13	0	650	-7	647	-7	-3
14	0	700	-8	697	-8	-3
15	0	750	-10	747	-10	-3
16	0	800	-11	796	-11	-4
17	0	850	-12	845	-12	-5
18	0	900	-13	896	-13	-4
RMSE (cm)					6.44	2.60

B. Pengujian Rotary Encoder Arah Kanan

No	Posisi		Posisi Terukur		Error	
	X (cm)	Y (cm)	X (cm)	Y (cm)	X (cm)	Y (cm)
1	50	0	51	2	1	2
2	100	0	101	4	1	4
3	150	0	152	6	2	6
4	200	0	202	9	2	9
5	250	0	253	11	3	11
6	300	0	304	13	4	13
7	350	0	354	16	4	16
8	400	0	405	19	5	19
9	450	0	456	22	6	22
10	500	0	506	24	6	24
11	550	0	555	28	5	28
12	600	0	606	31	6	31
13	650	0	656	35	6	35
14	700	0	706	38	6	38
15	750	0	759	43	9	43
16	800	0	809	47	9	47
17	850	0	860	50	10	50
18	900	0	910	55	10	55
RMSE (cm)					5.98	29.91

C. Pengujian Rotary Encoder Arah Kiri

No	Posisi		Posisi Terukur		Error	
	X (cm)	Y (cm)	X (cm)	Y (cm)	X (cm)	Y (cm)
1	-50	0	-51	-2	-1	-2
2	-100	0	-101	-4	-1	-4
3	-150	0	-152	-7	-2	-7
4	-200	0	-202	-9	-2	-9
5	-250	0	-253	-11	-3	-11
6	-300	0	-304	-13	-4	-13
7	-350	0	-354	-16	-4	-16
8	-400	0	-405	-18	-5	-18
9	-450	0	-456	-21	-6	-21
10	-500	0	-506	-23	-6	-23
11	-550	0	-556	-26	-6	-26
12	-600	0	-607	-29	-7	-29
13	-650	0	-656	-32	-6	-32
14	-700	0	-707	-35	-7	-35
15	-750	0	-758	-38	-8	-38
16	-800	0	-809	-41	-9	-41
17	-850	0	-859	-44	-9	-44
18	-900	0	-909	-48	-9	-48
RMSE (cm)					5.89	27.00

D. Pengujian Penentuan Posisi Robot dengan Kamera

No	Posisi		Posisi Terukur		Error	
	X (cm)	Y (cm)	X (cm)	Y (cm)	X (cm)	Y (cm)
1	50	150	70	150	-20	0
2	100	150	105	151	-5	-1
3	125	150	128	151	-3	-1
4	150	150	149	149	1	1
5	175	150	178	150	-3	0
6	200	150	203	150	-3	0
7	225	150	207	152	18	-2
8	250	150	250	156	0	-6
9	275	150	243	152	32	-2
10	300	150	273	155	27	-5
11	325	150	295	157	30	-7
12	350	150	355	168	-5	-18
13	375	150	303	156	72	-6
14	400	150	368	161	32	-11
15	425	150	382	172	43	-22
16	450	150	410	168	40	-18
17	475	150	487	172	-12	-22
18	500	150	531	164	-31	-14
19	525	150	538	158	-13	-8
20	550	150	541	185	9	-35
21	50	175	48	165	2	10
22	75	175	68	177	7	-2

No	Posisi		Posisi Terukur		Error	
	X (cm)	Y (cm)	X (cm)	Y (cm)	X (cm)	Y (cm)
23	100	175	100	168	0	7
24	50	200	70	165	-20	35
25	100	200	110	195	-10	5
26	150	200	150	199	0	1
27	200	200	189	199	11	1
28	250	200	251	200	-1	0
29	300	200	310	201	-10	-1
30	350	200	350	203	0	-3
31	400	200	440	206	-40	-6
32	450	200	467	204	-17	-4
33	500	200	500	213	0	-13
34	550	200	541	219	9	-19
35	50	250	55	235	-5	15
36	100	250	98	241	2	9
37	150	250	150	243	0	7
38	200	250	190	245	10	5
39	250	250	243	240	7	10
40	300	250	298	243	2	7
41	350	250	376	246	-26	4
42	400	250	409	245	-9	5
43	450	250	454	249	-4	1
44	500	250	485	246	15	4
45	550	250	547	260	3	-10

No	Posisi		Posisi Terukur		Error	
	X (cm)	Y (cm)	X (cm)	Y (cm)	X (cm)	Y (cm)
46	50	300	102	302	-52	-2
47	100	300	96	295	4	5
48	150	300	152	296	-2	4
49	200	300	214	296	-14	4
50	250	300	260	298	-10	2
51	300	300	302	302	-2	-2
52	350	300	357	300	-7	0
53	400	300	410	301	-10	-1
54	450	300	465	296	-15	4
55	500	300	514	301	-14	-1
56	550	300	520	307	30	-7
57	50	350	119	349	-69	1
58	100	350	105	347	-5	3
59	150	350	148	347	2	3
60	200	350	191	349	9	1
61	250	350	276	350	-26	0
62	300	350	294	339	6	11
63	350	350	363	354	-13	-4
64	400	350	374	351	26	-1
65	450	350	457	350	-7	0
66	500	350	498	343	2	7
67	550	350	558	359	-8	-9
68	50	175	40	161	10	14

No	Posisi		Posisi Terukur		Error	
	X (cm)	Y (cm)	X (cm)	Y (cm)	X (cm)	Y (cm)
69	100	175	87	183	13	-8
70	150	175	127	185	23	-10
71	200	175	160	177	40	-2
72	250	175	233	181	17	-6
73	300	175	272	185	28	-10
74	350	175	338	185	12	-10
75	400	175	424	193	-24	-18
76	450	175	463	189	-13	-14
77	500	175	520	184	-20	-9
78	550	175	537	193	13	-18
79	50	225	63	209	-13	16
80	100	225	94	215	6	10
81	150	225	132	218	18	7
82	200	225	173	216	27	9
83	250	225	229	219	21	6
84	300	225	275	222	25	3
85	350	225	356	231	-6	-6
86	400	225	405	230	-5	-5
87	450	225	458	230	-8	-5
88	500	225	493	233	7	-8
89	550	225	539	244	11	-19
RMSE (cm)					**20.39**	**10.09**

E. Program Utama Pelatihan Data

```
#include <stdio.h>
#include <string>
#include <windows.h>
#include <time.h>
#include "neuralNetwork.h"
using namespace std;
clock_t t1, t2;
double dataOutputArray[1000][3], doANorm[1000][3];
double dataInputArray[1000][361], diANorm[1000][361];
double er[1000];

int main(){
  backPro nn(360, 2, 100, 20, 1000);
  nn.eps = 0.000001;
  t1 = clock();
  nn.weightInit();
  string line;
  ifstream input;
  string token;
  input.open("180617dataset.csv");
  int hitung = 0, i = 0;
  cout << "Reading input value " << endl;
  if (input.is_open()){
        int i = 0, j = 1;
        while (i<nn.dataSet){
              getline(input, token);
              istringstream ssin(token);
              while (j <= nn.inLayer){
                    getline(ssin, token, ',');
                    dataInputArray[i][j] = stod(token);
        diANorm[i][j] = 2 * dataInputArray[i][j] / 250 - 1;

                    dataInputArray[i][0] = 1;
                    diANorm[i][0] = 1;
```

```
                    j++;
                }
                j = 1;
                i++;
                int percent = i * 100 / nn.dataSet;
                cout << "\r" << percent << "% completed";
                //cout << std::string(percent/10, '|');
                cout.flush();
        }
        cout << endl;
}
else {  cout << "Failed to open the file" << endl;
        system("pause");
        return 0;
}
input.close();
cout << "Reading input done!" << endl;
input.open("180617target.csv");
cout << "Reading desired output " << endl;
if (input.is_open()){
        int i = 1, j = 1;
        while (i<nn.dataSet){
                getline(input, token);
                istringstream ssin(token);
                while (j <= nn.outLayer){
                        getline(ssin, token, ',');
                        dataOutputArray[i][j] = stod(token);
                doANorm[i][j] = dataOutputArray[i][j]/100;
                        dataOutputArray[i][0] = 1;
                        doANorm[i][0] = 1;
                        j++;
                }
                j = 1;
                i++;
                int percent = i * 100 / nn.dataSet;
                cout << "\r" << percent << "% completed";
```

```
                cout.flush();
            }
            cout << endl;
    }
    else {  cout << "Failed to open the file" << endl;
            system("pause");
            return 0;
    }
    input.close();
    cout << "Reading target done!" << endl;

    int iterasi = 0;
    while (true){
            for (int k = 0;k < nn.dataSet;k++){
                iterasi++;
                nn.xin = diANorm[k];
                nn.t = doANorm[k];
                nn.feedForward();
                er[k] = nn.getError(nn.t, nn.y);
                nn.backProp();
                nn.updateW();
            }

            double sse = nn.getSSE(er);
            int epoch = iterasi / nn.dataSet;
    cout << "Epoch : " << epoch << " , SSE = " << sse << endl;
            if (sse < nn.eps ){
                t2 = clock();
                cout << "Done!" << endl;
                nn.save_data(epoch,sse);
            }
            else if (epoch % 100 == 0) nn.save_data(epoch,sse);
    }
    system("pause");
    return 0;
}
```

BIODATA PENULIS

Adnan Rachmawan lahir di Sleman, 30 Agustus 1993, yang merupakan anak dari pasangan Tri Gunarto dan Istri Maryani. Penulis menyelesaikan pendidikan dasar di SDN Mejing 2 Sleman (1999 s.d. 2005) lalu dilanjutkan dengan pendidikan menengah di SMPN 6 Yogyakarta (2005 s.d. 2008) dan SMAN 7 Yogyakarta (2008 s.d. 2011). Penulis mengenyam pendidikan tahap D3 di jurusan Teknik Elektro, Sekolah Vokasi, Universitas Gadjah Mada yang kemudian melanjutkan ke jenjang S-1 di Jurusan Teknik Elektro, Fakultas Teknologi Elektro, Institut Teknologi Sepuluh Nopember Surabaya.

Email:

adnanrachmawan@gmail.com

Halaman ini sengaja dikosongkan