ITS
Institut
Teknologi
Sepuluh Nopember

TUGAS AKHIR – TE 141599

PENENTUAN POSISI ROBOT SEPAK BOLA BERODA BERDASARKAN PENGINDRAAN VISUAL

Pandu Surya Tantra
NRP 07111440000006

Dosen Pembimbing
Dr. Ir. Djoko Purwanto, M.Eng.
Dr. Ir. Hendra Kusuma, M.Eng.Sc.

DEPARTEMEN TEKNIK ELEKTRO
Fakultas Teknologi Elektro
Institut Teknologi Sepuluh Nopember
Surabaya 2018

ITS
Institut
Teknologi
Sepuluh Nopember

TUGAS AKHIR – TE 141599

PENENTUAN POSISI ROBOT SEPAK BOLA BERODA BERDASARKAN PENGINDRAAN VISUAL

Pandu Surya Tantra
NRP 07111440000006

Dosen Pembimbing
Dr. Ir. Djoko Purwanto, M.Eng.
Dr. Ir. Hendra Kusuma, M.Eng.Sc.

DEPARTEMEN TEKNIK ELEKTRO
Fakultas Teknologi Elektro
Institut Teknologi Sepuluh Nopember
Surabaya 2018

Institut
Teknologi
Sepuluh Nopember

FINAL PROJECT – **TE 141599**

WHEELED SOCCER ROBOT LOCALIZATION BASED ON VISUAL SENSING

Pandu Surya Tantra
NRP 07111440000006

Supervisors
Dr. Ir. Djoko Purwanto, M.Eng.
Dr. Ir. Hendra Kusuma, M.Eng.Sc.

DEPARTMENT OF ELECTRICAL ENGINEERING
Faculty of Electrical Technology
Institut Teknologi Sepuluh Nopember
Surabaya 2018

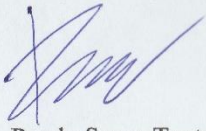
PERNYATAAN KEASLIAN TUGAS AKHIR

Dengan ini saya menyatakan bahwa isi sebagian maupun keseluruhan Tugas Akhir saya dengan judul "Penentuan Posisi Robot Sepak Bola Beroda berdasarkan Pengindraan Visual" 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, 20 April 2018



Pandu Surya Tantra
NRP 07111440000006

PENENTUAN POSISI ROBOT SEPAK BOLA BERODA BERDASARKAN PENGINDRAAN VISUAL

TUGAS AKHIR

Diajukan untuk 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. Djoko Purwanto, M.Eng.	Dr. Ir. Hendra Kusuma, M.Eng.Sc.
NIP. 196512111990021002	NIP. 196409021989031003

SURABAYA
JULI, 2018

Penentuan Posisi Robot Sepak Bola Beroda berdasarkan Pengindraan Visual

Nama : Pandu Surya Tantra
Pembimbing I : Dr. Ir. Djoko Purwanto, M.Eng.
Pembimbing II : Dr. Ir. Hendra Kusuma, M.Eng.Sc.

ABSTRAK

Penentuan posisi adalah salah satu masalah utama dalam pembuatan robot sepak bola beroda. Apabila robot tidak mengetahui posisinya pada lapangan, maka koordinasi antar robot dan tindakan selanjutnya akan sulit ditentukan. Salah satu cara untuk mengetahui posisi robot adalah dengan menggunakan metode *odometry*. Metode ini mudah terpengaruh slip sehingga menghasilkan nilai *error* terintegral yang membuat nilai *error* posisi semakin besar seiring dengan semakin jauhnya robot berpindah. Posisi robot dan nilai pembacaan kompas harus sering dikalibrasi pada saat pertandingan untuk menghilangkan *error* posisi. Oleh karena itu, dibuatlah sebuah sistem penentuan posisi robot dengan menggunakan sensor yang tidak terpengaruh nilai keluaran sebelumnya, yaitu sebuah kamera *omnidirectional* yang terpasang pada robot menggunakan algoritma *Convolutional Neural Network (CNN)*. Sistem dirancang dalam dua bagian, yaitu penentuan posisi global dan penentuan posisi lokal. Posisi global robot dikalibrasi menggunakan kamera yang dipasang di atas lapangan dengan algoritma jaringan syaraf tiruan. Sedangkan posisi lokal robot dikalibrasi menggunakan kamera *omnidirectional* yang terpasang pada robot dengan memanfaatkan data posisi global robot. Dari hasil pengujian, sistem penentuan posisi global robot dapat mengeluarkan data posisi robot dengan *error* rata-rata sebesar 6,25cm. Sedangkan sistem penentuan posisi lokal dapat mengeluarkan data posisi robot dengan ketelitian 100cm dan akurasi 100% saat robot diam serta akurasi 99,9% saat robot berjalan dengan kecepatan 15cm/s.

Kata kunci: *Convolutional Neural Network*, Robot Sepak Bola, Kamera *Omnidirectional*.

.....Halaman ini sengaja dikosongkan.....

Wheeled Soccer Robot Localization based on Visual Sensing

Name : Pandu Surya Tantra
Supervisor I : Dr. Ir. Djoko Purwanto, M.Eng.
Supervisor II: Dr. Ir. Hendra Kusuma, M.Eng.Sc.

ABSTRACT

Localization is one of the main problems in wheeled soccer robot game. If the robot does not know its position on the field, then coordination between robots and subsequent actions will be difficult to determine. One way to know the position of the robot is to use odometry method. This method is easily affected by the slip so as to produce an integrated error value that makes the position error value are increasing as the robot moves further away. The position of the robot and the compass reading value should often be calibrated at the time of the game to eliminate the position error. Therefore, a robot positioning system is created using sensors that are not affected by the previous output value, an omnidirectional camera mounted on the robot using the Convolutional Neural Network (CNN) algorithm. The system is designed in two parts, namely global positioning and local positioning. The global position of the robot is calibrated using a camera mounted on top of the field with artificial neural network algorithms. While the local position of the robot is calibrated using an omnidirectional camera mounted on the robot by utilizing the global position data of the robot. From the test results, robot global positioning system can issue robot position data with an average error of 6.25cm. While the local positioning, system can issue robot position data with 100cm precision, 100% accuracy when the robot is not moving and 99.9% accuracy when the robot moves at 15cm/s.

Keywords: Convolutional Neural Network, Soccer Robot, Omnidirectional Camera.

.....Halaman ini sengaja dikosongkan.....

KATA PENGANTAR

Puji syukur penulis panjatkan kepada Allah SWT yang telah melimpahkan berkah, rahmat, serta hidayah-Nya sehingga penulis dapat menyelesaikan tugas akhir dengan judul Penentuan Posisi Robot Sepak Bola Beroda berdasarkan Pengindraan Visual dengan lancar dan tepat waktu.

Tugas akhir ini diusulkan sebagai lanjutan dari penelitian robot sepak bola beroda dan menjadi prasyarat penulis untuk mendapatkan gelar Sarjana Teknik dari Departemen Teknik Elektro ITS, Bidang Studi Elektronika. Penulis ingin menyampaikan terima kasih kepada:

1. Bapak, ibu, dan kakak yang senantiasa memberikan dukungan, baik material maupun non material.
2. Bapak Ardyono Priyadi, S.T., M.Eng., Dr.Eng . selaku ketua Departemen Teknik Elektro, Fakultas Teknologi Elektro, Institut Teknologi Sepuluh Nopember.
3. Bapak Dr. Ir. Djoko Purwanto, M.Eng. dan Bapak Dr. Ir. Hendra Kusuma, M.Eng.Sc. selaku dosen pembimbing yang selalu memberikan saran serta bantuan dalam pengerjaan tugas akhir ini.
4. Bapak dan ibu dosen Departemen Teknik Elektro, khususnya Bidang Studi Elektronika.
5. Seluruh rekan serta dosen pembimbing Tim Robotika ITS yang telah memberikan tempat untuk mengembangkan diri selama menjalani masa perkuliahan.
6. Seluruh anggota Tim IRIS ITS yang telah membantu dalam pengerjaan robot sepak bola beroda.

Kesempurnaan hanya milik Allah SWT, penyusunan tugas akhir ini tentu masih banyak kekurangan. Untuk itu penulis sangat mengharapkan kritik dan saran yang membangun. Semoga penelitian ini dapat memberikan manfaat terutama untuk perkembangan robotika Indonesia agar dapat bersaing pada tingkat internasional.

Surabaya, 1 Juni 2018

Pandu Surya Tantra

.....Halaman ini sengaja dikosongkan.....

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

.....Halaman ini sengaja dikosongkan.....

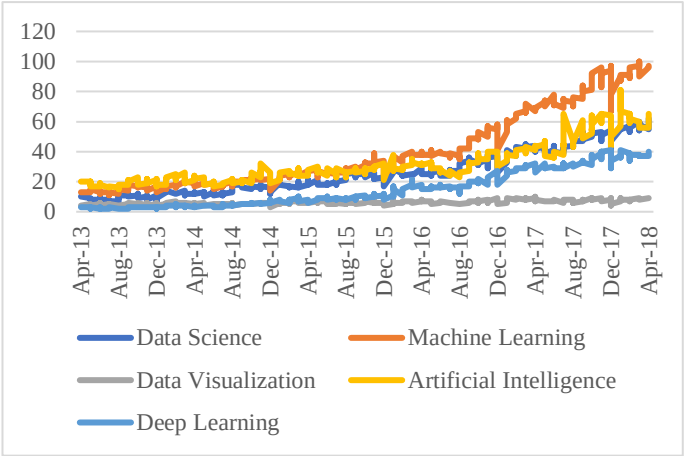
BAB 1
PENDAHULUAN

1.1 Latar Belakang

Penentuan posisi robot adalah salah satu masalah utama dalam pembuatan robot sepak bola. Apabila robot tidak mengetahui posisinya pada lapangan, maka koordinasi antar robot dan tindakan selanjutnya akan sulit ditentukan. Salah satu cara untuk mengetahui posisi robot adalah dengan menggunakan metode *odometry*. Komponen utama *odometry* adalah *rotary encoder* dan kompas digital. *Rotary encoder* digunakan untuk mengetahui seberapa jauh robot telah berpindah dari posisi awal, sedangkan kompas digunakan sebagai penunjuk ke arah manakah robot berpindah. Namun metode ini mudah terpengaruh slip sehingga menghasilkan nilai *error* terintegral yang membuat nilai *error* posisi semakin besar seiring dengan semakin jauhnya robot berpindah. Posisi robot dan nilai pembacaan kompas digital harus sering dikalibrasi pada saat pertandingan untuk menghilangkan *error* posisi. Hal tersebut dirasa kurang efektif karena memakan waktu yang tidak sebentar. Untuk menghilangkan nilai *error* yang terintegral diperlukan sensor yang tidak terpengaruh nilai keluaran sebelumnya, yaitu sebuah kamera *omnidirectional* yang terpasang pada robot.

Teknologi jaringan syaraf tiruan adalah salah satu cabang dari kecerdasan buatan yang saat ini semakin banyak dikembangkan. Tren pencarian kata kunci tentang kecerdasan buatan selama lima tahun terakhir [1] dapat dilihat pada **Gambar I.1**. Implementasi jaringan syaraf tiruan seperti pada [2], menggunakan kamera untuk menentukan orientasi *quadcopter* terhadap lingkungan. Dengan penalaran serupa, posisi lokal robot sepak bola terhadap lapangan dapat ditentukan menggunakan citra lapangan dan jaringan syaraf tiruan yang dilatih secara *offline*. Sebuah kamera dipasang di atas lapangan untuk menentukan posisi global robot. Posisi global tersebut kemudian digunakan sebagai kalibrasi posisi lokal robot sepak bola.

Dalam tugas akhir ini, telah dirancang dan dibuat sebuah sistem penentuan posisi robot sepak bola beroda menggunakan kamera *omnidirectional* dan jaringan syaraf tiruan. Diharapkan hasil dari tugas akhir ini dapat diimplementasikan pada pertandingan yang sebenarnya.



Gambar 1.1 Tren pencarian kata kunci tentang kecerdasan buatan.

1.2 Rumusan Masalah

Permasalahan yang dibahas pada tugas akhir ini antara lain:

a. Bagaimana mengalibrasi posisi global robot menggunakan kamera yang dipasang di atas lapangan.
b. Bagaimana mengalibrasi posisi lokal robot menggunakan posisi global robot dan citra lapangan dari kamera omnidirectional pada robot.

1.3 Tujuan

Tujuan yang ingin dicapai pada tugas akhir ini antara lain:

a. Memperoleh posisi global robot yang terkalibrasi menggunakan kamera yang dipasang di atas lapangan.
b. Memperoleh posisi lokal robot yang terkalibrasi menggunakan posisi global robot dan citra lapangan dari kamera *omnidirecitonal* pada robot.

1.4 Batasan Masalah

Batasan ruang lingkup dari tugas akhir ini antara lain:

a. Objek yang berada di lapangan adalah satu buah robot
b. Pengambilan dan pengujian data dilakukan pada waktu malam hari atau pada ruangan tertutup.

1.5 Metodologi Penelitian

Metodologi dalam menyelesaikan tugas akhir ini adalah sebagai berikut:

a. Studi Literatur

Melakukan pengumpulan dasar teori yang menunjang penulisan tugas akhir. Dasar teori yang menunjang tugas akhir ini antara lain:

- Model kinematika *omnidirectional holonomic drive*.
- Pengolahan citra digital.
- Arsitektur jaringan syaraf tiruan.
- Arsitektur jaringan syaraf konvolusi.

Dasar teori di atas diambil dari buku, jurnal, proceeding, dan artikel-artikel di internet.

b. Pembuatan mekanik robot

Sebagian besar bagian mekanik robot dikerjakan dengan mesin laser CNC supaya hasil potongan pelat yang dihasilkan bersih dan rapi. Bahan yang digunakan untuk mekanik robot adalah aluminium. Bagian struktur robot yang membutuhkan kekuatan lebih seperti *spacer* dan pelindung robot menggunakan bahan *stainless steel*/baja. Sedangkan bagian pendukung yang tidak termasuk struktur utama robot menggunakan bahan akrilik atau plastik.

c. Pembuatan elektronik robot

Sistem elektronik robot yang dibuat meliputi sensor, *motor driver*, *microcontroller*, dan komponen-komponen penunjang lainnya, seperti tombol dan *LCD display*. Pada tahap ini elektronik robot dan *PC* sudah dapat berkomunikasi secara dua arah dan dapat melakukan gerakan-gerakan dasar bergerak maju-mundur, kiri-kanan, dan berputar.

d. Pengambilan data kalibrasi

Data kalibrasi yang diambil dibagi menjadi dua, yaitu kalibrasi posisi global dan kalibrasi posisi lokal. Data kalibrasi global didapatkan dengan cara mencatat secara manual posisi robot menggunakan pencitraan kamera dari atas lapangan dan posisi robot di lapangan. Data kalibrasi lokal didapatkan dengan cara menjalankan robot pada jalur yang sudah ditentukan sambil merekam citra lapangan menggunakan kamera *omnidirectional* pada robot.

e. *Pre-process* data kalibrasi

Data kalibrasi global yang didapatkan berupa nilai koordinat pixel dan koordinat centimeter posisi robot dinormalisasi dengan cara membagi semua nilai dengan seribu supaya range data kalibrasi

berada pada $-1{,}0 \sim 1{,}0$ untuk mempercepat proses pelatihan jaringan syaraf tiruan [3]. Data kalibrasi lokal yang didapatkan berupa video kemudian dicuplik setiap *frame*-nya. *Frame-frame* tersebut dikelompokkan dan diolah terlebih dahulu untuk menghilangkan citra objek yang tidak diinginkan sebelum digunakan pada proses pelatihan jaringan syaraf tiruan.

f. Pelatihan dan verifikasi jaringan syaraf tiruan

Data kalibrasi yang sudah dinormalisasi dan di-*pre-process* kemudian digunakan untuk proses pelatihan jaringan syaraf tiruan. Seluruh proses pelatihan jaringan syaraf tiruan memanfaatkan API sumber terbuka *Keras* dan *TensorFlow*. Pelatihan dijalankan menggunakan *Graphic Processing Unit (GPU)* untuk mempersingkat waktu [4]. GPU yang digunakan adalah *NVIDIA GeForce GTX 1070* dan *NVIDIA GeForce 840M*.

g. Pengujian dan analisa

Pengujian dan analisa dilakukan untuk menentukan keandalan dari sistem yang telah dibuat. Pengujian dilakukan dengan cara menguji terlebih dahulu penentuan posisi global robot dan melakukan analisa. Kemudian jika sudah didapatkan hasil yang optimal pengujian dan analisa dilakukan pada penentuan posisi lokal robot.

h. 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 tentang pengerjaan tugas akhir yang telah dikerjakan, meliputi pendahuluan, tinjauan pustaka dan teori penunjang, perancangan sistem, pengujian, dan penutup.

1.6 Sistematika

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

- Bab I : Pendahuluan

  Bab ini berisi penjelasan latar belakang, perumusan masalah, batasan masalah, tujuan, metodologi penelitian, sistematika penulisan, serta relevansi.
- Bab II : Tinjauan Pustaka dan Teori Penunjang

  Bab ini menjelaskan teori penunjang dan literatur yang dibutuhkan dalam pengerjaan tugas akhir. Teori yang menunjang penelitian tugas akhir ini antara lain kinematika *omnidirectional holonomic drive*, pengolahan citra digital, dan jaringan syaraf tiruan.

- Bab III : Perancangan Sistem
  Bab ini menjelaskan tentang perancangan sistem, baik mekanik maupun elektronik dan pemrograman yang digunakan dalam menentukan posisi robot menggunakan pengindraan visual.
- Bab IV : Pengujian
  Bab ini berisi hasil pengujian sistem penentuan posisi robot dan analisa data atas hasil pengujian yang dilakukan.
- Bab V : Penutup
  Bab ini merupakan bagian akhir dari buku laporan tugas akhir yang berisi kesimpulan dari tugas akhir yang dikerjakan. Bagian ini juga berisi saran pengembangan penelitian yang lebih lanjut.

1.7 Relevansi dan Manfaat

Hasil yang diharapkan dari tugas akhir ini adalah sistem yang dapat menentukan posisi robot sepak bola beroda menggunakan pengindraan visual sehingga kesalahan posisi yang didapatkan seminimal mungkin.

.....Halaman ini sengaja dikosongkan.....

BAB 2
TINJAUAN PUSTAKA

2.1 Pengolahan Citra Digital

Citra dapat didefinisikan sebagai fungsi $f(x,y)$, di mana x dan y adalah koordinat bidang Cartesius dan f adalah amplitudo pada setiap titik koordinat (x,y) atau dapat diartikan sebagai intensitas atau tingkat keabuan pada titik koordinat tersebut. Apabila sebuah citra memiliki nilai f pada keseluruhan bidang (x,y) dengan range yang terbatas dan bernilai diskrit, maka citra tersebut merupakan citra digital. Pemrosesan citra digital dapat didefinisikan sebagai pemrosesan citra dengan menggunakan komputer digital. Citra digital tersusun atas elemen dengan jumlah terbatas dan masing-masing elemen tersebut menempati koordinat tertentu dan memiliki amplitudo/nilai. Elemen-elemen tersebut kemudian disebut dengan elemen citra atau pixel.

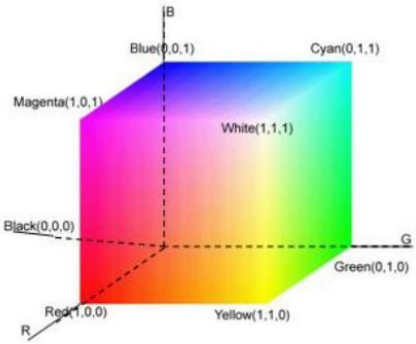
2.1.1 Citra dengan Skala Keabuan

Citra dengan skala keabuan hanya memiliki satu komponen amplitudo pixel, yaitu gradasi warna hitam-putih yang menghasilkan efek warna abu-abu. Ketelitian intensitas keabuan tergantung pada resolusi nilai gradasi hitam-putih yang digunakan, misalkan pada resolusi 8-bit nilai 0 merujuk pada warna hitam sempurna dan nilai 255 merujuk pada warna putih sempurna.

2.1.2 Citra dengan Ruang Warna RGB

Secara umum mata manusia dapat membedakan warna merah, hijau, dan biru (*Red, Green, and Blue*) sehingga warna tertentu dapat dipandang sebagai kombinasi dari ketiga wana tersebut. Ruang warna RGB biasa digunakan pada sistem grafik komputer dan kamera digital. Ruang warna ini dimodelkan dalam bentuk kubus tiga dimensi dengan pusat warna merah, hijau, dan biru berada pada ujung sumbu R, G, dan B seperti pada **Gambar 2.1**. Warna putih dan warna hitam berada di titik asal dan warna putih berada pada titik sudut yang berseberangan dengan titik asal.

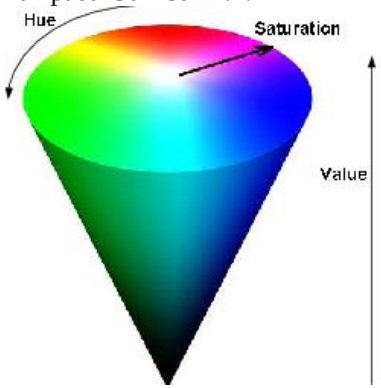
Ruang warna RGB tidak ideal diimplementasikan pada pengolahan citra digital. Hal tersebut dikarenakan warna merah, hijau, dan biru sesungguhnya saling mempengaruhi untuk membuat satu warna tertentu. Untuk mengolah warna tertentu, ruang warna HSV lebih mudah diolah daripada ruang warna RGB.



Gambar 2.1 Model ruang warna RGB

2.1.3 Citra dengan Ruang Warna HSV

HSV merupakan ruang warna yang merepresentasikan warna seperti yang dilihat oleh mata manusia. HSV kependekan dari *Hue, Saturation, and Value* merupakan besaran warna yang biasa digunakan dalam pengolahan citra digital. Ruang warna HSV dimodelkan dengan kerucut terbalik, pusat warna hitam berada pada dasar kerucut dan pusat warna putih berada pada pusat tutup kerucut dikelilingi oleh campuran komposisi warna merah, hijau, dan biru seperti ditunjukkan pada **Gambar 2.2**.



Gambar 2.2 Model ruang warna HSV

Nilai *hue* menunjukkan warna suatu pixel yang diturunkan dari komponen warna merah, hijau, dan biru. Nilai *saturation* menunjukkan intensitas warna suatu pixel. Nilai *value* menunjukkan kecerahan suatu pixel, misal merah gelap dan merah cerah. Nilai H, S, dan V dapat diperoleh berdasarkan nilai R, G, dan B dengan rumus sebagai berikut [5]:

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

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

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

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

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

2.1.4 Citra Biner

Citra biner adalah citra yang masing-masing pixel-nya hanya bisa mempunyai salah satu dari dua nilai, yaitu nilai 0 dan 1 (nilai biner). Nilai 0 merepresentasikan warna hitam sedangkan nilai 1 merepresentasikan warna putih. Citra biner digunakan untuk keperluan memperoleh bentuk suatu objek atau untuk keperluan *masking* pada pengolahan citra digital.

2.1.5 Segmentasi Citra

Secara umum segmentasi citra bertujuan untuk memisahkan atau memilih objek-objek yang terdapat pada citra. Segmentasi pada citra dengan satu objek sederhana bisa dilakukan dengan memisahkan citra objek tersebut dari latar belakangnya. Segmentasi citra dengan

objek yang rumit dan jumlah yang banyak akan membutuhkan algoritma yang lebih kompleks.

Segmentasi citra pada umumnya dilakukan sebagai langkah awal mengklasifikasikan objek di dalam citra sehingga pengolahan citra yang selanjutnya mudah dilakukan. Dari hasil segmentasi tersebut didapatkan fitur objek yang dipilih. Teknik segmentasi yang mudah dilakukan di antaranya dengan segmentasi warna pada citra. Cara kerja teknik ini adalah dengan memberikan nilai ambang atas dan nilai ambang bawah pada citra HSV. Pixel yang mempunyai nilai di luar batas ambang akan dihitamkan (diberi nilai 0) sedangkan pixel yang lainnya dijadikan putih (diberi nilai 1).

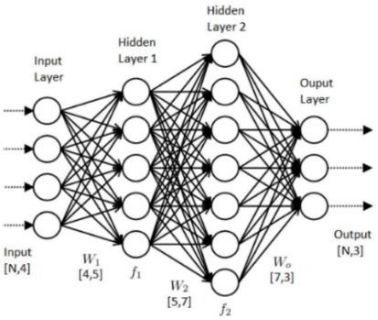
2.1.6 *OpenCV*

Open Source Computer Vision atau yang lebih dikenal dengan *OpenCV* adalah API sumber terbuka yang berisi fungsi-fungsi pemrograman terkait tentang pengolahan citra digital. API ini berlisensi BSD sehingga bebas digunakan untuk keperluan akademik, penelitian, maupun komersial. *OpenCV* ditulis dalam bahasa C++, Pyhton, dan Java. *OpenCV* mendukung sistem operasi Windows, Linux, Android, iOS, dan MacOS [6].

Sejak peluncuran *OpenCV* versi alfa pada Januari 1999, *OpenCV* sudah digunakan pada berbagai aplikasi, produk, dan riset untuk memecahkan masalah terkait pengolahan citra digital. Aplikasi-aplikasi tersebut antara lain pengolahan citra satelit, pengurangan derau pada citra medis, dan sistem pemantau keamanan [6].

2.2 Jaringan Syaraf Tiruan

Machine learning merupakan istilah yang dipakai dalam bidang keteknikan komputer di mana komputer dapat melakukan suatu tugas tanpa benar-benar diprogram. Dengan menggunakan sampel data dan algoritma tertentu, komputer akan dibuat seolah-olah belajar dari sampel data yang diberikan. Kualitas dan jumlah sampel data yang diberikan sangat mempengaruhi tingkat akurasi *machine learning*. Semakin benar dan semakin banyak jumlah sampel data yang diberikan maka tingkat akurasi dari *machine learning* akan semakin tinggi, dan juga berlaku sebaliknya.



Gambar 2.3 Struktur umum jaringan syaraf tiruan

Salah satu metode *machine learning* yang populer dan paling banyak digunakan saat ini adalah jaringan syaraf tiruan *(artificial neural network).* Jaringan syaraf tiruan terdiri dari sejumlah *node* yang tersusun berlapis-lapis *(layers)* dan saling terhubung satu sama lain. Masing-masing *node* memiliki koefisien pembobotan yang disebut *weight*. Dan masing-masing *layer* memiliki nilai konstanta yang disebut *bias.* Struktur umum jaringan syaraf tiruan ditunjukkan pada **Gambar 2.3**.

Nilai *weight* dan *bias* digunakan jaringan syaraf tiruan untuk menghitung keluaran dari masukan yang diberikan. Keluaran *layer* pertama digunakan sebagai masukan *layer* kedua, begitu juga keluaran *layer* kedua digunakan sebagai masukan *layer* ketiga, dan seterusnya. Sebelumnya, sebuah fungsi aktivasi diaplikasikan pada masing-masing *layer* untuk mengukur seberapa aktif *node* pada *layer* tersebut.

Jaringan syaraf tiruan perlu dilatih supaya nilai *weight* dan *bias* nya beradaptasi sehingga dapat memberikan nilai keluaran yang sesuai untuk masukan yang diberikan. Terdapat dua metode pelatihan jaringan syaraf tiruan, yaitu *supervised learning* dan *unsupervised learning* [7]. Metode pelatihan yang umum dilakukan adalah *supervised learning.*

2.2.1 *Supervised Learning*

Supervised learning merupakan metode yang umum digunakan untuk melatih jaringan syaraf tiruan. Pelatihan dilakukan dengan cara memberikan sampel data kalibrasi berupa pasangan antara masukan dan keluaran dari sebuah sistem yang ingin diselesaikan [7]. Pada awal proses pelatihan jaringan syaraf tiruan akan memberikan nilai yang salah untuk sebuah masukan. Kemudian nilai keluaran tersebut dibandingkan dengan nilai keluaran yang seharusnya. Selisih dari kedua nilai tersebut digunakan untuk memperbarui *weight* dan *bias*

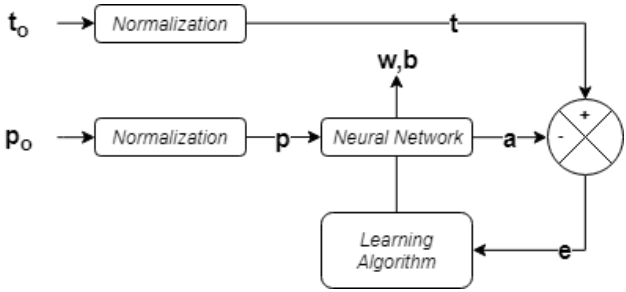
jaringan syaraf tiruan. Contoh implementasi *supervised learning* misalnya pada [2] menggunakan data kalibrasi berupa foto jalan setapak perhutanan dari tiga sudut pandang.

2.2.2 *Backpropagation*

Backpropagation merupakan salah satu algoritma yang digunakan untuk melatih jaringan syaraf tiruan. Algoritma ini termasuk dalam metode *supervised* learning dikarenakan pelatihan tidak dapat dilakukan tanpa memberikan sampel data kalibrasi pada sistem. Meskipun berbagai model jaringan syaraf tiruan sudah dipelajari, namun model dengan algoritma *backpropagation* adalah yang paling sering digunakan. Statistik tentang penggunaan algoritma *backpropagation* belum pernah dilakukan. Namun diperkirakan 90% dari total aplikasi jaringan syaraf tiruan pada komersial dan industri menggunakan algoritma *backpropagation* dan turunannya [7].

Algoritma *backpropagation* secara umum dapat dijelaskan sebagai berikut [7]:

- ***Outer loop***. Mengulangi *inner loop* sampai jaringan syaraf tiruan dianggap mampu memberikan nilai keluaran dari nilai masukan dengan benar.
- ***Inner loop***. Untuk setiap data kalibrasi yang dimasukkan ke dalam jaringan syaraf tiruan, mengulangi tiga langkah di bawah ini sampai keluaran yang dihasilkan sesuai dengan yang diharapkan seperti diilustrasikan pada **Gambar 2.4**.
  - Langkah 1. Memberikan data masukan kepada jaringan syaraf tiruan.
  - Langkah 2. *Feedforward.* Mengolah data masukan dengan nilai *weight* dan *bias* yang ada pada setiap *layer* kemudian mengamati nilai keluaran pada *layer* keluaran.
  - Langkah 3. *Backward propagation of error corrections.* Membandingkan keluaran jaringan syaraf tiruan dengan keluaran data kalibrasi. Jika keluaran jaringan syaraf tiruan sesuai dengan keluaran data kalibrasi, maka kembali ke proses awal *outer loop*. Jika tidak sesuai, maka merambat-balikkan nilai *error* keluaran melalui jaringan syaraf tiruan dan memperbarui nilai *weight* dan *bias.*



Gambar 2.4 Prosedur *backpropagation* pada jaringan syaraf tiruan

Implementasi algoritma *backpropagation* dalam memperbarui nilai *weight* dan *bias* pada jaringan syaraf tiruan M *layer* adalah dengan persamaan sebagai berikut:

Feedforward

Langkah pertama yang dilakukan adalah merambatkan maju nilai masukan jaringan syaraf tiruan. Nilai masukan dinormalisasi terlebih dahulu untuk mempercepat proses pelatihan [3].

$$\boldsymbol{a}^0 = \boldsymbol{p} \tag{2.6}$$

$$\boldsymbol{a}^{m+1} = \boldsymbol{f}^{m+1}(\boldsymbol{w}^{m+1}\boldsymbol{a}^m + \boldsymbol{b}^{m+1}), \quad m = 0,1,\cdots,M-1 \tag{2.7}$$

$$\boldsymbol{a} = \boldsymbol{a}^M \tag{2.8}$$

Di mana:

$\boldsymbol{a}^0$: masukan jaringan syaraf tiruan
$\boldsymbol{a}^M$: keluaran jaringan syaraf tiruan
$\boldsymbol{a}^i$: keluaran *node layer* ke-i
$\boldsymbol{p}$: masukan yang sudah dinormalisasi
$\boldsymbol{f}^i$: fungsi aktivasi *layer* ke-i
$\boldsymbol{w}^i$: *weight layer* ke-i
$\boldsymbol{b}^i$: *bias layer* ke-i

Backpropagation

Selanjutnya nilai keluaran jaringan syaraf tiruan dibandingkan dengan keluaran data kalibrasi. Selisih dari kedua nilai tersebut kemudian dirambatkan balik untuk memperbarui nilai *weight* dan *bias.*

$$\boldsymbol{s}^{M} = -2\dot{\boldsymbol{F}}^{M}(\boldsymbol{n}^{M})(\boldsymbol{t} - \boldsymbol{a}) \tag{2.9}$$

$$\boldsymbol{s}^{m} = \dot{\boldsymbol{F}}^{m}(\boldsymbol{n}^{m})(\boldsymbol{w}^{m+1})^{T}\boldsymbol{s}^{m+1}, \quad m = M-1, \cdots, 2, 1 \tag{2.10}$$

$$\dot{\boldsymbol{F}}^{m}(\boldsymbol{n}^{m}) = \begin{bmatrix} \dot{f}^{m}(n_1^m) & 0 & \cdots & 0 \\ 0 & \dot{f}^{m}(n_2^m) & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \dot{f}^{m}(n_{s^m}^m) \end{bmatrix} \tag{2.11}$$

$$\dot{f}^{m}(n_j^m) = \frac{\partial f^{m}(n_j^m)}{\partial n_j^m} \tag{2.12}$$

Di mana:

$\boldsymbol{a}$: keluaran jaringan syaraf tiruan
$\boldsymbol{t}$: keluaran data kalibrasi yang sudah dinormalisasi
$\boldsymbol{s}^{M}$: sensitifitas *layer* keluaran
$\boldsymbol{s}^{i}$: sensitifitas *layer* ke-i
$\boldsymbol{w}^{i}$: *weight layer* ke-i
$\boldsymbol{b}^{i}$: *bias layer* ke-i
$\dot{\boldsymbol{F}}^{i}$: turunan pertama fungsi aktivasi *layer* ke-i
$\boldsymbol{n}^{i}$: keluaran *node layer* ke-i

Perbaruan *weight* dan *bias*

Nilai *weight* dan *bias* diperbarui berdasarkan sensitivitas dari tiap - *layer* yang dirambati. Kecepatan jaringan syaraf tiruan dalam memperbarui nilai *weight* dan *bias* ditentukan oleh besarnya *learning rate* [8].

$$\boldsymbol{w}^m(k+1) = \boldsymbol{w}^m(k) - \alpha \boldsymbol{s}^m(\boldsymbol{a}^{m-1})^T \quad (2.13)$$

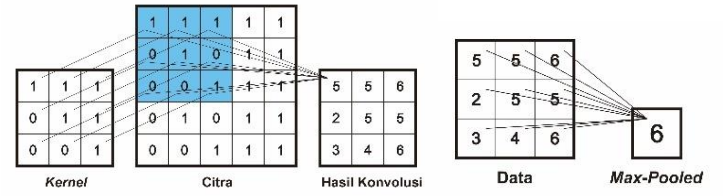
$$\boldsymbol{b}^m(k+1) = \boldsymbol{b}^m(k) - \alpha \boldsymbol{s}^m \quad (2.14)$$

Di mana:

$\boldsymbol{w}^i$: *weight layer* ke-i
$\boldsymbol{b}^i$: *bias layer* ke-i
$\boldsymbol{s}^i$: sensitifitas *layer* ke-i
k : iterasi
α : *learning rate*

2.2.3 *Convolutional Neural Network*

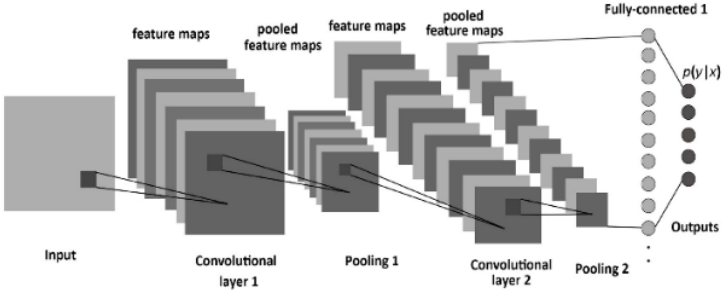
Jaringan syaraf konvolusi atau *Convolutional Neural Network (CNN)* adalah salah satu turunan dari jaringan syaraf tiruan yang didesain untuk mengolah data dua dimensi. Jaringan syaraf konvolusi termasuk dalam jenis *Deep Neural Network* karena kedalaman jaringan yang tinggi dan banyak diaplikasikan pada pengolahan citra digital. Pada kasus klasifikasi citra digital, jaringan syaraf tiruan biasa kurang sesuai untuk digunakan karena tidak menyimpan informasi spasial dari data citra digital dan menganggap setiap pixel adalah fitur yang independen sehingga menghasilkan hasil yang kurang baik. Secara umum, arsitektur jaringan syaraf konvolusi sama dengan jaringan syaraf tiruan. Yang membedakan adalah penggunaan *convolution layer* dan *pooling layer* pada jaringan syaraf konvolusi.



Gambar 2.5 Operasi konvolusi dan operasi *pooling*

- ***Convolution layer.*** *Convolution layer* melakukan operasi konvolusi pada keluaran *layer* sebelumnya. *Layer* tersebut adalah proses utama yang mendasari sebuah jaringan syaraf konvolusi. Konvolusi pada citra digital berarti mencuplik pixel dengan luas tertentu pada semua *offset* yang mungkin kemudian melakukan operasi perkalian dan penjumlahan dengan sebuah *kernel* yang luasnya sama dengan citra yang dicuplik. Jumlah dan ukuran *kernel* yang digunakan dalam jaringan syaraf konvolusi tidak dapat dipastikan tergantung pada implementasinya. Operasi konvolusi pada citra 5x5 pixel dengan *kernel* 3x3 pixel ditunjukkan pada **Gambar 2.5**.
- ***Pooling layer.*** *Pooling* adalah operasi yang bertujuan untuk mereduksi ukuran/resolusi data. Dalam pengolahan citra digital, *pooling* berguna untuk menurunkan variansi posisi fitur. Jenis *pooling layer* yang sering digunakan dalam jaringan syaraf konvolusi adalah *max pooling.* Cara kerja *max pooling* adalah dengan mengambil nilai terbesar dari data yang dimasukkan. Operasi *pooling* pada data ditunjukkan pada **Gambar 2.5**.

Pada jaringan syaraf konvolusi, penempatan *convolution layer* dan *pooling layer* berada di awal jaringan berdekatan dengan data masukan. Ukuran data akan mengecil setelah dirambatkan melalui *layer* tersebut yang kemudian dirambatkan melalui jaringan syaraf tiruan biasa. Arsitektur jaringan syaraf konvolusi secara utuh diilustrasikan pada **Gambar 2.6**.



Gambar 2.6 Arsitektur jaringan syaraf konvolusi

Gambar 2.7 Logo *TensorFlow*

2.2.4 *TensorFlow*

TensorFlow adalah API sumber terbuka yang berisi fungsi-fungsi pemrograman terkait tentang *machine learning* dan *data science*. *TensorFlow* dikembangkan oleh Google dan versi pertamanya dirilis pada April 2015. Keunggulan *TensorFlow* adalah fitur komputasinya yang dapat dijalankan secara paralel dengan memanfaatkan GPU [9]. *TensorFlow* kompatibel pada hampir semua sistem operasi termasuk Android dan iOS. Bahasa pemrograman yang digunakan adalah *Python*.

2.2.5 *Keras*

Keras adalah API sumber terbuka yang berisi fungsi-fungsi pemrograman terkait tentang jaringan syaraf tiruan dan turunannya. *Keras* dikembangkan oleh François Chollet dan versi pertamanya dirilis pada Maret 2015. *Keras* adalah API yang memudahkan dalam perancangan dan pemrograman jaringan syaraf tiruan. *Keras* tidak dapat berjalan sendiri dan secara *default* menggunakan *TensorFlow* sebagai *backend*-nya [10].

2.3 Odometry

Odometry adalah salah satu metode untuk menentukan posisi pada robot. Metode *odometry* memanfaatkan nilai keluaran sistem sensor tertentu untuk mengestimasi perpindahan robot. Pada robot sepak bola beroda, *odometry* yang digunakan untuk mengestimasi perpindahan robot memanfaatkan putaran *rotary encoder* dan orientasi kompas digital. Perpindahan translasi robot didapatkan dengan cara mengintegrasi kecepatan putaran *rotary encoder* dan orientasi kompas digital. Sedangkan perpindahan rotasi robot murni didapatkan dari orientasi kompas digital.

Gambar 2.8 Logo *Keras*

2.3.1 *Rotary Encoder*

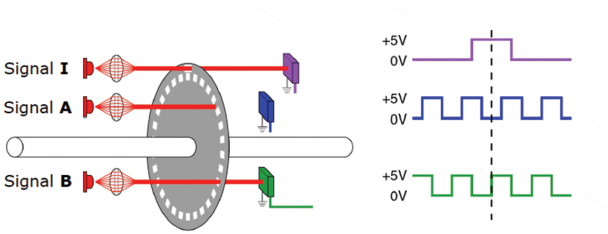
Rotary encoder adalah sensor yang berfungsi untuk mengukur perpindahan rotasi suatu benda yang berputar pada porosnya. Cara menggunakan *rotary encoder* adalah dengan memasang kopling pada poros *rotary encoder* dengan poros benda berputar yang ingin diukur. *Rotary encoder* menurut cara kerjanya dibagi menjadi dua, yaitu *absolute* dan *incremental.*

Absolute rotary encoder memiliki kemampuan untuk menghasilkan keluaran yang pasti ketika sensor dinyalakan. *Rotary encoder* jenis ini memiliki elemen pengukuran yang diberi kode tertentu menurut posisi porosnya. Bahkan perpindahan poros ketika sensor tidak dalam keadaan hidup dapat terbaca ketika sensor dinyalakan.

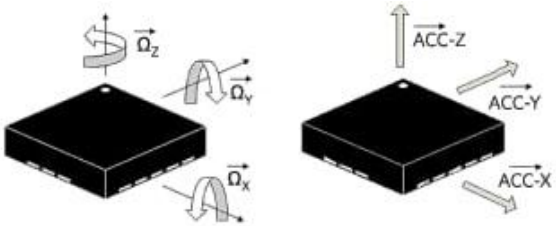
Incremental rotary encoder menghasilkan sinyal keluaran setiap kali porosnya berputar. Ketelitian dari sinyal keluaran terhadap putaran poros tergantung pada resolusi sensor. Setiap kali sensor dinyalakan, sensor akan mulai menghitung dari posisi nol atau pada posisi referensi yang sudah ditentukan sebelumnya. *Rotary encoder* jenis ini menggunakan sensor *hall-effect* atau sensor optik untuk mengukur perpindahan porosnya. *Rotary encoder* dengan sensor optik menggunakan komponen optoelektronik *transmitter/receiver* berupa LED dan fototransistor diilustrasikan pada **Gambar 2.9.**

2.3.2 Kompas digital

Rotary encoder dapat digunakan untuk mengetahui perpindahan translasi robot. Perpindahan tersebut dicuplik dengan periode tertentu sehingga kecepatan perpindahan translasi robot dapat diketahui. Kemudian pada setiap hasil cuplikan dihitung komponen kecepatan pada sumbu-x dan sumbu-y menggunakan data keluaran kompas digital.



Gambar 2.9 *Incremental rotary encoder*



Gambar 2.10 Giroskop 3 sumbu dan akselerometer 3 sumbu

Dalam *odometry*, kompas digital berperan sangat penting untuk menunjukkan orientasi robot terhadap lapangan. Karena setiap cuplikan kecepatan robot akan dihitung komponennya terhadap sumbu-x dan sumbu-y lapangan berdasarkan orientasi robot. Kesalahan penghitungan komponen kecepatan akan berakibat pada keluaran posisi bergeser (*drifting*) dari posisi yang sebenarnya.

Kompas digital terdiri dari sensor giroskop dan akselerometer mekanik yang berukuran sangat kecil. Sensor giroskop berfungsi menunjukkan kecepatan sudut benda pada porosnya sedangkan sensor giroskop berfungsi menunjukkan akselerasi benda pada sumbu tertentu. Besaran yang terukur menggunakan giroskop dan akselerometer diilustrasikan pada **Gambar 2.10**. Kedua sensor tersebut kemudian ditanam pada rangkaian terintegrasi yang disebut *MEMS (Microelectromechanical Systems)*. Ketelitian kompas digital ditentukan oleh resolusi keluaran dari *MEMS* yang dapat diatur melalui konfigurasi awal ketika sensor dinyalakan.

2.4 STM32F4

STM32F4 merupakan mikrokontroler berbasis *ARM* 32bit. Mikrokontroler ini memiliki *clock speed* maksimal sebesar 168Mhz dengan memanfaatkan fitur *Phase Locked Loop (PLL)* menggunakan kristal 8Mhz sampai 25Mhz. *STM32F4* memiliki 14 *timer* dan 6 *Universal Asynchronous Receiver Transmitter (UART)* yang dapat digunakan secara independen. 6 dari 14 *timer* dapat digunakan untuk membaca *rotary encoder*, yaitu TIM1, TIM2, TIM3, TIM4, TIM5, dan TIM8. Sedangkan TIM6 dan TIM7 hanya dapat digunakan sebagai *counter* atau *timer* internal dikarenakan tidak mempunyai pin keluaran fisik pada mikrokontroler. TIM9, TIM10, TIM11, TIM12, TIM13, dan TIM14 dapat digunakan sebagai penghasil sinyal *Pulse Width Modulation*

(PWM) untuk menggerakkan motor pada robot. Mikrokontroler STM32F4 juga dilengkapi dengan *Direct Memory Access (DMA)* yang dapat dimanfaatkan untuk mengelola *UART* tanpa membebani komputasi utama sehingga diharapkan semua komputasi dapat berjalan secara *realtime*.

2.5 Intel NUC

Intel NUC merupakan sebuah *mini-PC* yang memiliki spesifikasi perangkat keras cukup tinggi. *Intel NUC* menggunakan prosesor *Intel Core i7* generasi ke-6 dengan *clock speed* mencapai 3.5Ghz. *Intel NUC* menggunakan media penyimpanan berupa *Solid State Drive (SSD)* yang memiliki kecepatan transfer data jauh lebih tinggi daripada *harddisk* mekanik konvensional. *Mini-PC* digunakan sebagai pengolah citra digital dan pengolah jaringan syaraf tiruan untuk menentukan posisi robot. Seluruh kontrol robot juga diolah pada *mini-PC* yang kemudian keluarannya dikirim ke mikrokontroler. Pengiriman data dari *mini-PC* ke mikrokontroler dan sebaliknya menggunakan protokol serial *RS232*.

2.6 Kamera Omnidirectional

Kamera *omnidirectional* adalah kamera yang mempunyai sudut pandang 360 derajat pada bidang datar. Kamera *omnidirectional* terdiri dari tiga bagian utama, yaitu sensor, lensa, dan cermin. Yang membedakan kamera *omnidirectional* dengan kamera pada umunya adalah adanya cermin cembung yang berada di depan lensa. Cermin tersebut berfungsi untuk memantulkan cahaya dari lingkungan di sekitar kamera dari segala arah. Tingkat kecembungan cermin mempengaruhi seberapa luas sudut pandang kamera pada sumbu vertikal.

Gambar 2.11 *STM32F4 Discovery* dan *Intel NUC*

Gambar 2.12 Hasil citra kamera *omnidirectional*

Keuntungan menggunakan kamera *omnidirectional* pada robot sepak bola adalah robot dapat mengetahui citra dari segala arah seperti ditunjukkan pada **Gambar 2.12**.

2.7 Tinjauan Pustaka

Beberapa penelitian serupa atau yang sebagian menyerupai tugas akhir ini antara lain:

2.7.1 *A Machine Learning Approach to Visual Perception of Forest Trails for Mobile Robots* [2]

Pada penelitian tersebut, peneliti menggunakan algoritma jaringan syaraf konvolusi untuk mengendalikan *quadcopter* secara otomatis di dalam hutan dengan jalan setapak. Keluaran dari jaringan syaraf tiruan yang digunakan berupa tiga kelas besaran kendali: kiri, maju, dan kanan. Algoritma yang sama digunakan pada tugas akhir ini dengan modifikasi pada jumlah kelas keluaran, yaitu 63 kelas derajat kepercayaan posisi robot.

2.7.2 Desain Dan Implementasi Pengukuran Posisi Bola Menggunakan Kamera 360 Derajat [11]

Pada penelitian tersebut, peneliti menggunakan algoritma jaringan syaraf tiruan untuk mengestimasi jarak centimeter bola dari robot berdasarkan jarak pixel bola. Hasil yang didapatkan dari penelitian tersebut adalah jarak bola dapat diestimasi dengan tingkat

kesalahan yang kecil. Algoritma yang sama digunakan pada tugas akhir ini untuk mengestimasi koordinat centimeter robot pada lapangan berdasarkan koordinat pixel kamera.

2.7.3 Penentuan Posisi Robot Sepak Bola Beroda Menggunakan Rotary Encoder dan Kamera [12]

Pada penelitian tersebut, peneliti menentukan posisi lokal robot sepak bola beroda menggunakan jaringan syaraf tiruan dengan masukan berupa jarak pixel robot dengan marka terdekat pada semua arah. Pada penelitian tersebut, posisi robot hanya dapat diestimasi pada separuh bagian lapangan. Pada tugas akhir ini, seluruh nilai pixel citra lapangan digunakan untuk menentukan posisi lokal robot dengan pendekatan algoritma yang hampir sama.

2.7.4 *Optimization of GPU and CPU acceleration for neural networks layers implemented in python* [13]

Pada penelitian tersebut, peneliti membandingkan proses pelatihan jaringan syaraf tiruan menggunakan GPU dan CPU. Dari penelitian yang dilakukan, diketahui bahwa pelatihan jaringan syaraf tiruan menggunakan GPU lebih cepat hingga 250 kali tergantung spesifikasi GPU yang digunakan. Proses pelatihan jaringan syaraf tiruan pada tugas akhir ini menggunakan GPU *NVIDIA GeForce 840M* dan *NVIDIA GeForce GTX 1070*.

2.7.5 *CAMBADA Team Description Paper* [14]

Pada publikasi tersebut, Tim CAMBADA menggunakan algoritma Monte Carlo Localization untuk menentukan posisi robot. Masukan untuk Monte Carlo Localization berupa jarak robot dengan halangan di sekitarnya berdasarkan pembacaan LiDar. Selain hasil pembacaan LiDar, peta virtual di mana robot dijalankan juga dimasukkan ke dalam sistem.

BAB 3
PERANCANGAN SISTEM

Tugas akhir ini merupakan bagian dari pengembangan sistem penentuan posisi robot sepak bola Tim IRIS ITS. Seluruh perancangan dan pengerjaan robot dikoordinasikan dengan anggota tim dan diarahkan oleh dosen pembimbing tim. Robot yang digunakan dalam tugas akhir ini selanjutnya diikutkan dalam Kontes Robot Sepak Bola Beroda Indonesia sebagai salah satu dari tiga robot yang bertanding.

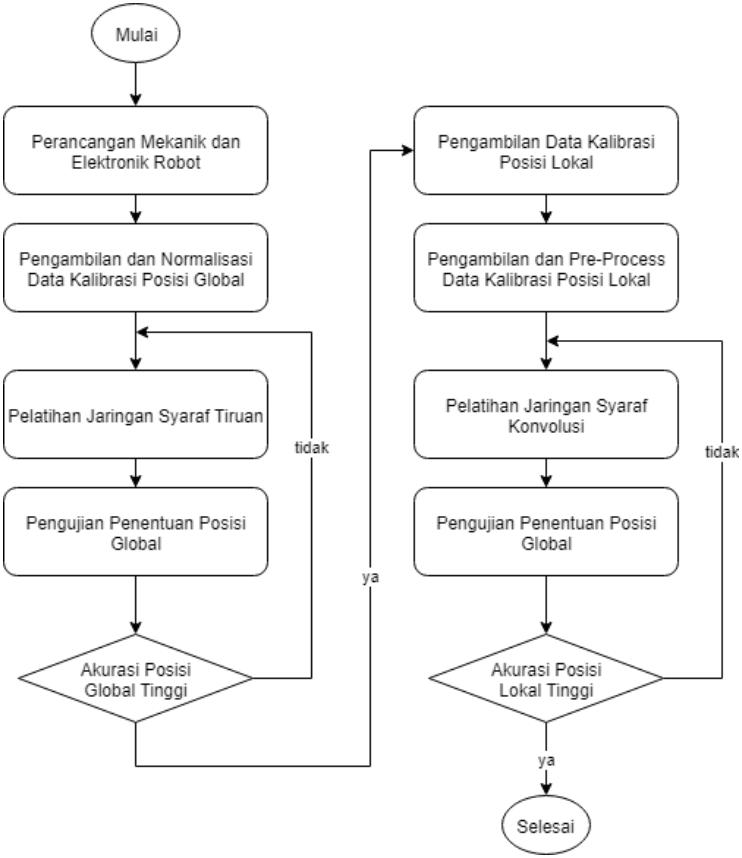
Bab ini membahas tentang perancangan sistem penentuan posisi robot sepak bola beroda berdasarkan pengindraan visual. Perancangan sistem yang dibahas meliputi perancangan mekanik, elektronik, dan program. Perancangan mekanik meliputi pembuatan platform robot sepak bola beroda. Perancangan elektronik meliputi pembuatan rangkaian elektronika, pemasangan sensor dan antarmuka, serta pemasangan *mini-PC* pada robot. Perancangan program meliputi komunikasi antara mikrokontroler dengan *mini-PC*, pengolahan citra digital, jaringan syaraf tiruan dan komunikasi antara robot dan komputer pengendali.

Sistem utama yang dirancang dibagi menjadi dua tahap, yaitu penentuan posisi global dan penentuan posisi lokal. Setiap tahap penentuan posisi dilakukan pengambilan data kalibrasi yang kemudian digunakan untuk melatih jaringan syaraf tiruan. Diagram alur perancangan sistem penentuan posisi ditunjukkan pada **Gambar 3.1** dan diagram sistem secara keseluruhan ditunjukkan pada **Gambar 3.2**.

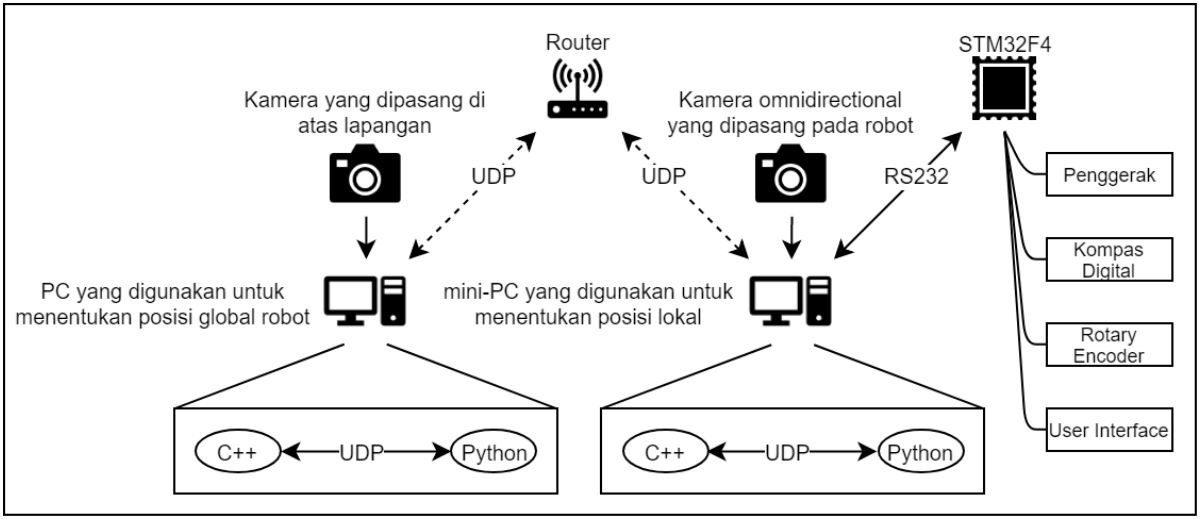
Penentuan posisi global dilakukan menggunakan kamera yang dipasang di atas lapangan. Kamera tersebut mengambil citra lapangan dan robot secara keseluruhan dari atas. Data kalibrasi posisi global robot didapatkan dengan cara mencatat secara manual koordinat centimeter robot pada lapangan dan koordinat pixel robot pada citra. Kemudian sebuah jaringan syaraf tiruan dilatih menggunakan data kalibrasi tersebut.

Penentuan posisi lokal dilakukan menggunakan kamera *omnidirectional* yang terpasang pada robot. Kamera tersebut mengambil citra lapangan di sekitar robot dengan sudut pandang 360 derajat. Data kalibrasi posisi lokal robot didapatkan dengan cara menjalankan robot dengan dipandu oleh posisi global bersamaan dengan mengambil citra lapangan menggunakan kamera *omidirectional*. Kemudian sebuah jaringan syaraf konvolusi dilatih menggunakan data kalibrasi tersebut. Algoritma jaringan syaraf konvolusi dipilih karena tidak perlu mendefinisikan fitur citra tertentu yang digunakan sebagai masukan.

Dengan menggunakan jaringan syaraf konvolusi, maka tidak perlu memprogram algoritma tambahan agar sistem penentuan posisi dapat bekerja. Keandalan algoritma penentuan posisi kemudian tergantung pada akurasi data kalibrasi yang dimasukkan pada proses pelatihan jaringan syaraf konvolusi.



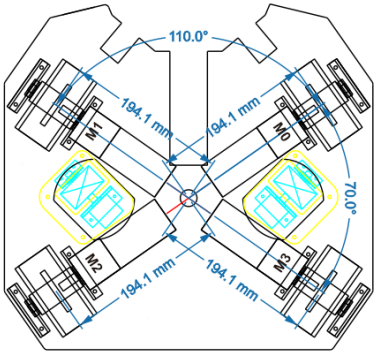
Gambar 3.1 Diagram alur perancangan sistem penentuan posisi robot



Gambar 3.2 Diagram sistem penentuan posisi

3.1 Perancangan Mekanik dan Elektronik

Perancangan mekanik dan elektronik robot merupakan tahapan awal yang dilakukan sebelum perancangan sistem penentuan posisi robot dikerjakan. Supaya penelitian berjalan dengan lancar maka bagian mekanik dan elektronik harus dapat bekerja dengan baik, termasuk komunikasi antara *mini-PC* dan mikrokontroler, pembacaan sensor, gerakan dasar robot, dan antarmuka pengguna.



Gambar 3.3 Desain penggerak robot sepak bola beroda

3.1.1 Perancangan Mekanik

Sebagian besar bagian mekanik robot dikerjakan dengan mesin laser CNC supaya hasil potongan pelat yang dihasilkan bersih dan rapi. Bahan yang digunakan untuk mekanik robot adalah aluminium. Bagian struktur robot yang membutuhkan kekuatan lebih seperti *spacer* dan pelindung robot menggunakan bahan *stainless steel*/baja.

Penggerak robot menggunakan mekanisme *holonomic drive* dengan empat roda omni. Konfigurasi roda robot dibuat tidak simetris dengan mengutamakan kecepatan dan traksi gerakan robot pada sumbu-y. Masing-masing roda omni disusun dengan beda sudut 70 derajat dan 110 derajat. Desain mekanik penggerak robot ditunjukkan pada **Gambar 3.3**. Kecepatan positif motor menggerakkan robot berlawanan arah jarum jam. Kecepatan masing-masing motor diperoleh menggunakan rumus sebagai berikut:

$$\begin{bmatrix} \dot{\theta}_{M0} \\ \dot{\theta}_{M1} \\ \dot{\theta}_{M2} \\ \dot{\theta}_{M3} \end{bmatrix} = \begin{bmatrix} -\sin(35°) & \cos(35°) & 1 \\ -\sin(145°) & \cos(145°) & 1 \\ -\sin(-35°) & \cos(-35°) & 1 \\ -\sin(-145°) & \cos(-145°) & 1 \end{bmatrix} \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\emptyset} \end{bmatrix} \quad (3.1)$$

Pseudo-code untuk mendapatkan kecepatan masing-masing robot adalah sebagai berikut:

```
m[0] = (vx * -0.5735) + (vy * 0.8191) + vw;
m[1] = (vx * -0.5735) + (vy * -0.8191) + vw;
m[2] = (vx * 0.5735) + (vy * -0.8191) + vw;
m[3] = (vx * 0.5735) + (vy * 0.8191) + vw;
```

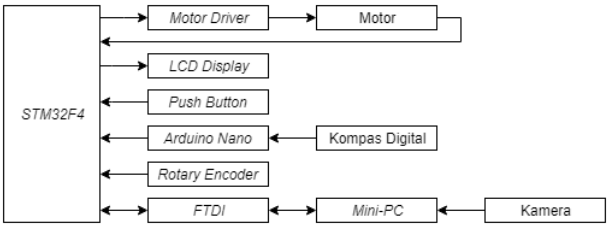
Kamera *omnidirectional* dipasang pada bagian kepala robot dengan sumbu-x sensor kamera berada sejajar dengan sumbu-y robot. Pemasangan kamera seperti ini bertujuan agar jarak pandang citra ke depan dan belakang robot lebih jauh karena pergerakan robot cenderung ke arah tersebut. Kamera dipasang pada ketinggian 54 centimeter dari lantai.

3.1.2 Perancangan Elektronik

Perancangan elektronik robot meliputi kontrol motor penggerak, komunikasi antara mikrokontroler dan *mini-PC*, dan komponen antarmuka pengguna.

- **Kontrol motor.** Mikrokontroler STM32F4 bertugas melakukan komputasi dan kontrol *closed-loop* kecepatan motor. Kontrol *closed-loop* tersebut dapat dicapai dengan memanfaatkan *hall-effect rotary encoder* yang terpasang pada poros motor. Motor yang digunakan adalah *PG45* dengan rasio gigi 1:19 *planetary gear*. Kontrol *closed-loop* dimaksudkan agar gerakan robot sesuai dengan arah vektor kecepatan yang diperintahkan.
- **Komunikasi serial**. Mikrokontroler dihubungkan ke *mini-PC* dengan protokol serial *RS232* menggunakan pengubah *FTDI*. Komunikasi serial antara mikrokontroler dan *mini-PC* memanfaatkan fitur *DMA* sehingga tidak membebani komputasi utama pada mikrokontroler. Data yang dikirimkan sebesar 64 byte berisi informasi kecepatan, *odometry*, dan nilai pembacaan sensor.
- **Antarmuka pengguna.** Sebuah *LCD display* dipasang pada robot untuk menunjukkan status algoritma yang sedang berjalan. Lima tombol mekanik juga dipasang pada robot agar dapat digunakan untuk memberi masukan perintah secara manual atau untuk mengatur parameter kontrol.

Diagram blok sistem elektronik robot secara lengkap ditunjukkan pada **Gambar 3.4**.



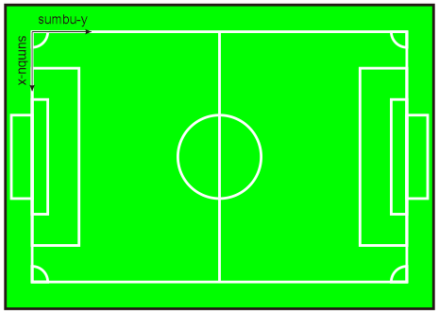
Gambar 3.4 Diagram blok sistem elektronik robot

3.2 Penentuan Posisi Global

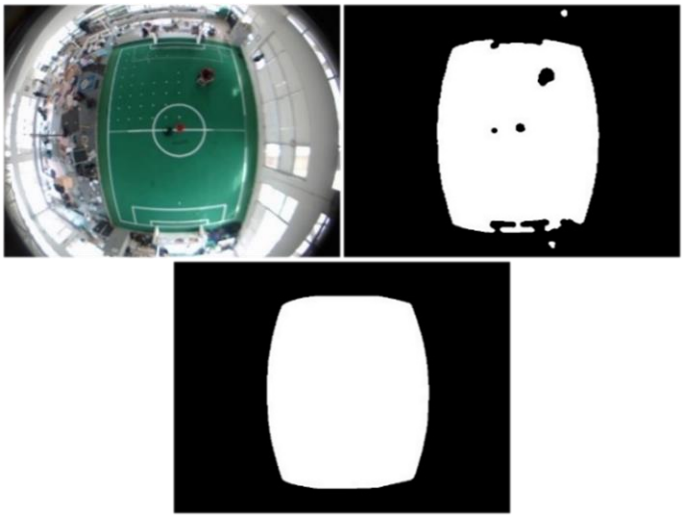
Sistem penentuan posisi global dibuat dengan tujuan agar posisi robot pada lapangan dapat diketahui menggunakan sensor eksternal. Sensor yang digunakan dalam menentukan posisi global tersebut berupa kamera yang dipasang di atas lapangan pada ketinggian 4 meter dari lantai. Kamera tersebut dipasangi lensa *fish-eye* untuk memperlebar sudut pandangnya menjadi 130 derajat. Citra dari kamera disegmentasi dan dinormalisasi untuk menghilangkan bagian citra selain lapangan. Data kalibrasi posisi global kemudian diambil secara manual dengan mencatat koordinat pixel dan koordinat centimeter posisi robot. Sebuah jaringan syaraf tiruan kemudian dilatih menggunakan data kalibrasi tersebut.

3.2.1 Sistem Koordinat Robot

Sistem koordinat yang digunakan adalah sistem koordinat Cartesius. Titik pusat koordinat berada pada sisi kiri belakang lapangan dengan gawang lawan berada di bagian depan. Dimensi lapangan yang digunakan sebesar 6x9 meter mengacu pada buku panduan Kontes Robot Sepak Bola Indonesia Divisi Beroda [15]. Sistem koordinat posisi robot diilustrasikan pada **Gambar 3.5**.



Gambar 3.5 Sistem koordinat posisi robot



Gambar 3.6 Segmentasi warna dan kontur citra lapangan

3.2.2 Pengolahan Citra Digital

Citra lapangan dalam ruang warna RGB diubah menjadi ruang warna HSV untuk kemudian dilakukan segmentasi warna dan segmentasi kontur pada citra tersebut. Keluaran yang dihasilkan dari segmentasi adalah citra biner dengan wana putih merepresentasikan lapangan dan warna hitam merepresentasikan objek selain lapangan. Segmentasi warna dan kontur pada citra ditunjukkan pada **Gambar 3.6**.

Segmentasi warna dilakukan dengan memberikan nilai ambang pada citra HSV. *Pseudo-code* untuk segmentasi warna pada citra adalah sebagai berikut:

```
cvtColor(frame, hsv, CV_BGR2HSV);

if (hl > hh)
{
  Mat dstA; Mat dstB;
  inRange(hsv, Scalar(hl, sl, vl), Scalar(180, sh, vh),
    dstA);
  inRange(hsv, Scalar(0, sl, vl), Scalar(hh, sh, vh),
    dstB);
  bitwise_or(dstA, dstB, threshold);
}
```

```
smorphologyEx(threshold, threshold, MORPH_OPEN,
  getStructuringElement(MORPH_ELLIPSE, Size(7, 7)));
morphologyEx(threshold, threshold, MORPH_CLOSE,
  getStructuringElement(MORPH_ELLIPSE, Size(11, 11)));
```

Segmentasi kontur dilakukan dengan mencari semua kontur yang mungkin pada citra biner hasil segmentasi warna. Kemudian dari masing-masing kontur tersebut dicari *convex hull*-nya dan dipilih *convex hull* dengan luasan terbesar. *Pseudo-code* untuk segmentasi kontur pada citra biner adalah sebagai berikut:

```
//-----Mencari Contour
vector<Mat> contours;
vector<Vec4i> hierarchy;
findContours(threshold, contours, hierarchy, 3, 2);

//-----Mencari Hull
vector<Mat> hull(contours.size());
for (int i = 0; i < contours.size(); i++)
  convexHull(Mat(contours[i]), hull[i]);

//-----Mencari Hull Terbesar
int largest_hull_index = 0;
int largest_hull_area = 0;
for (int i = 0; i < hull.size(); i++)
  if (contourArea(hull[i]) > largest_hull_area)
  {
    largest_hull_area = contourArea(hull[i]);
    largest_hull_index = i;
  }

threshold = Scalar(0);
drawContours(threshold, hull, largest_hull_index,
  Scalar(255), -1);
```

Dari citra biner hasil segmentasi dibuat persegi panjang virtual untuk menyelubungi citra biner tersebut. Citra lapangan kemudian dipotong menurut persegi panjang untuk mengambil bagian lapangan saja. Hasil potongan citra selanjutnya dinormalisasi resolusinya menjadi 480x640 pixel. Normalisasi citra lapangan ditunjukkan pada **Gambar 3.7**. *Pseudo-code* untuk normalisasi citra adalah sebagai berikut:

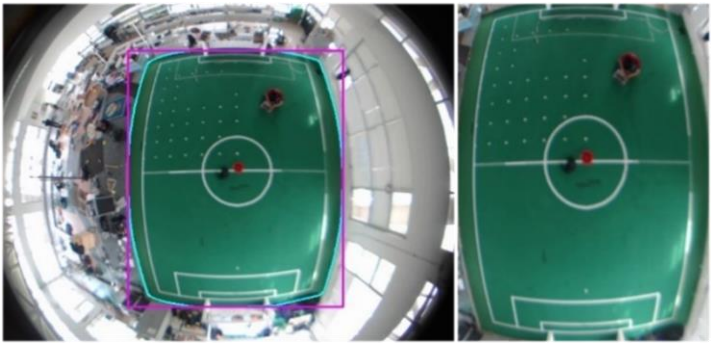
```
Rect roi = boundingRect(threshold);
field_frame = frame(roi);
resize(field_frame, field_frame, Size(480, 640));
```

Segmentasi dilakukan pada citra hasil normalisasi untuk menentukan koordinat pixel robot. Kepala robot (pada saat pengambilan data kalibrasi diganti menggunakan tongkat) diberi penanda berwarna merah agar kontras dengan warna hijau lapangan di sekitarnya. Langkah-langkah segmentasi citra robot sama dengan segmentasi citra lapangan hingga tahap mencari kontur dengan luasan terbesar. Koordinat pixel robot kemudian didapatkan dari pusat lingkaran virtual yang dibuat mengelilingi kontur tersebut. *Pseudo-code* untuk mendapatkan koordinat pixel robot adalah sebagai berikut:

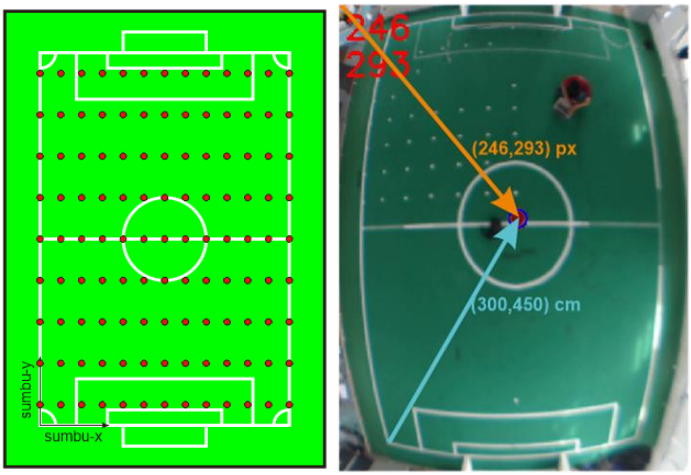
```
Point2f center; float radius;
minEnclosingCircle(contours[largest_contour_index], center,
  radius);

robot_x = center.x;
robot_y = center.y;
```

3.2.3 Pengambilan Data Kalibrasi

Data kalibrasi diambil dengan cara mencatat koordinat pixel robot dan koordinat centimeter robot pada lapangan. Untuk mempermudah pengambilan data, peran robot digantikan dengan tongkat dengan tinggi yang sama dengan tinggi robot dan dipasangi penanda berwarna merah.Data kalibrasi diambil mulai koordinat $(0,50)$ centimeter sampai koordinat $(600,850)$ centimeter. Jeda jarak titik pengukuran pada sumbu-x sebesar 50cm dan pada sumbu-y sebesar 100cm. Titik-titik pengambilan data kalibrasi diilustrasikan pada **Gambar 3.8**.



Gambar 3.7 Normalisasi citra lapangan



Gambar 3.8 Pengambilan data kalibrasi posisi global

Pada **Gambar 3.8** juga ditunjukkan pengambilan data kalibrasi posisi global robot. Koordinat centimeter robot berada pada (300,450) centimeter dan koordinat pixel robot berada pada (246,293) pixel. Seluruh 117 koordinat kalibrasi disimpan dalam file .csv untuk kemudian digunakan untuk proses pelatihan jaringan syaraf tiruan.

3.2.4 Pelatihan Jaringan Syaraf Tiruan

Jaringan syaraf tiruan yang digunakan memiliki 3 *layer* (2 *hidden layer* dan 1 *output layer*). *Node* pada masing-masing *hidden layer* sejumlah 25 *node* dan 2 *node* untuk *input layer* dan *output layer*. Fungsi aktivasi pada *hidden layer* adalah log sigmoid dan pada *output layer* adalah linier. *Pseudo-code* untuk membuat arsitektur jaringan syaraf tiruan adalah sebagai berikut:

```
model = Sequential()

model.add(Dense(units=25, activation='sigmoid',
  input_dim=2))
model.add(Dense(units=25, activation='sigmoid'))
model.add(Dense(units=2, activation='linear'))

cnn_model.compile(optimizer='Adam',
  loss='mse', metrics=['acc'])
```

Arsitektur jaringan syaraf tiruan beserta jumlah variabel (keluaran, *weight* dan *bias*) pada masing-masing *layer* adalah sebagai berikut:

```
Layer (type)         Output Shape     Param #
=================================================
dense_1 (Dense)      (None, 25)       75
_________________________________________________
dense_2 (Dense)      (None, 25)       650
_________________________________________________
dense_3 (Dense)      (None, 2)        52
=================================================
Total params: 777
Trainable params: 777
Non-trainable params: 0
```

Normalisasi dilakukan pada 117 data kalibrasi sebelum proses pelatihan dimulai. Normalisasi dilakukan dengan cara membagi data dengan seribu sehingga range data kalibrasi berada pada 0,0 ~ 1,0. Normalisasi pada data kalibrasi ditunjukkan pada **Gambar 3.9**. Kolom **x'** dan **y'** menunjukkan koordinat pixel robot sedangkan kolom **x** dan **y** menunjukkan koordinat centimeter robot.

Proses pelatihan berjalan menggunakan GPU *NVIDIA GeForce 840M*. Nilai *weight* dan *bias* yang menghasilkan nilai *error* terkecil pada setiap iterasi pelatihan disimpan pada file .hdf5. Pelatihan dapat ditunda dan dilanjutkan menggunakan file .hdf5 tersebut. **Gambar 3.10** menunjukkan file .hdf5 berisi *weight* dan *bias* pada iterasi 37589 sampai 37688 yang menghasilkan *error* 0,000040 sampai 0,000039.

```
In [14]: dataset.head()
```

Out[14]:

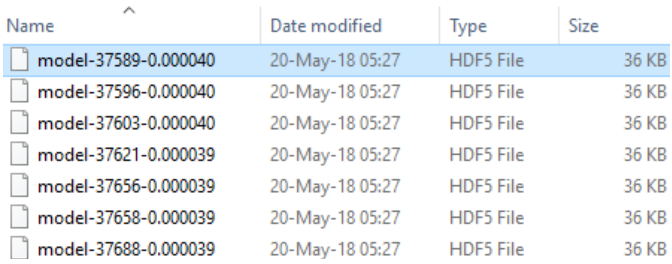
	x'	y'	x	y
0	46	603	0	50
1	71	611	50	50
2	102	619	100	50
3	135	627	150	50
4	168	629	200	50

```
In [16]: dataset.head()
```

Out[16]:

	x'	y'	x	y
0	0.046	0.603	0.00	0.05
1	0.071	0.611	0.05	0.05
2	0.102	0.619	0.10	0.05
3	0.135	0.627	0.15	0.05
4	0.168	0.629	0.20	0.05

Gambar 3.9 Normalisasi data kalibrasi



Name	Date modified	Type	Size
model-37589-0.000040	20-May-18 05:27	HDF5 File	36 KB
model-37596-0.000040	20-May-18 05:27	HDF5 File	36 KB
model-37603-0.000040	20-May-18 05:27	HDF5 File	36 KB
model-37621-0.000039	20-May-18 05:27	HDF5 File	36 KB
model-37656-0.000039	20-May-18 05:27	HDF5 File	36 KB
model-37658-0.000039	20-May-18 05:27	HDF5 File	36 KB
model-37688-0.000039	20-May-18 05:27	HDF5 File	36 KB

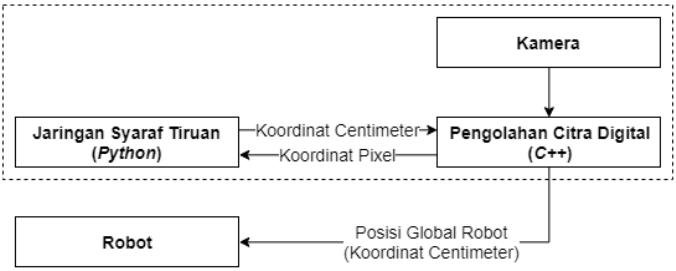
Gambar 3.10 *Weight* dan *bias* dalam file .hdf5

3.2.5 Implementasi Jaringan Syaraf Tiruan

Pengimplementasian jaringan syaraf tiruan untuk menentukan posisi global robot dilakukan dengan cara menjalankan dua program secara bersamaan. Satu program dalam bahasa *C++* digunakan untuk pengolahan citra digital dan satu program lainnya dalam bahasa *Python* digunakan untuk *feedforward* jaringan syaraf tiruan. Kedua program tersebut dihubungkan menggunakan protokol UDP dalam jaringan *localhost*.

Program pengolah citra digital mengirimkan data koordinat pixel robot yang kemudian dijadikan sebagai masukan jaringan syaraf tiruan. Program *feedforward* jaringan syaraf tiruan selanjutnya mengirimkan keluaran berupa koordinat centimeter robot kembali ke program pengolah citra digital. Dari program pengolah citra digital, data posisi global robot dikirimkan kepada robot menggunakan protokol UDP dalam jaringan intranet.

Diagram blok implementasi jaringan syaraf tiruan untuk menentukan posisi global robot ditunjukkan pada **Gambar 3.11**.



Gambar 3.11 Diagram blok implementasi jaringan syaraf tiruan

3.3 Penentuan Posisi Lokal

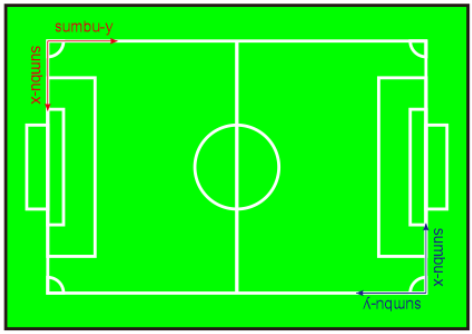
Sistem penentuan posisi lokal dibuat dengan tujuan agar posisi robot pada lapangan dapat diketahui menggunakan sensor internal robot yang tidak terpengaruh oleh pembacaan sensor sebelumnya. Sensor yang digunakan dalam menentukan posisi lokal robot adalah kamera *omnidirecitonal* yang terpasang pada robot. Citra dari kamera disegmentasi untuk menghilangkan bagian citra selain lapangan. Data kalibrasi posisi lokal kemudian diambil secara otomatis dengan cara menjalankan robot dengan jalur tertentu sambil merekam citra lapangan yang telah disegmentasi. Sebuah jaringan syaraf konvolusi kemudian dilatih menggunakan data kalibrasi tersebut.

3.3.1 Sistem Koordinat Robot

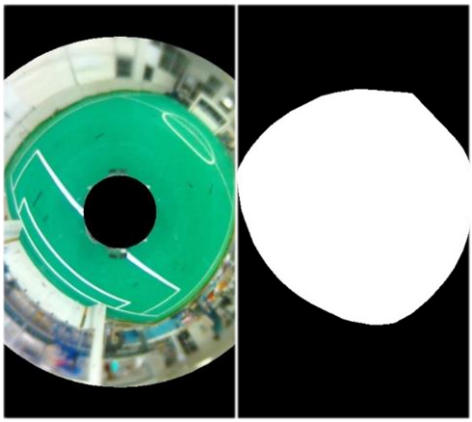
Sistem koordinat yang digunakan adalah sistem koordinat Cartesius sama dengan sistem koordinat yang digunakan pada sistem penentuan posisi lokal. Untuk mendapatkan data kalibrasi yang lebih beragam maka titik pusat koordinat diputar 180 derajat dan dipindah ke sisi lapangan yang berseberangan sehingga menghasilkan sistem koordinat yang sama namun dengan citra lapangan yang berbeda. Modifikasi sistem koordinat yang digunakan dalam penentuan posisi lokal diilustrasikan pada **Gambar 3.12**.

3.3.2 Pengolahan Citra Digital

Pengolahan citra digital dilakukan untuk melakukan segmentasi pada lapangan sehingga objek lain yang bukan merupakan lapangan diabaikan oleh sistem. Proses segmentasi lapangan pada sistem penentuan posisi lokal hampir sama dengan proses segmentasi lapangan pada sistem penentuan posisi global.



Gambar 3.12 Modifikasi sistem koordinat robot



Gambar 3.13 Segmentasi citra lapangan

Citra digital dalam ruang warna RGB diubah ke dalam ruang warna HSV dan diberi nilai ambang sehingga dihasilkan citra biner lapangan. Dari citra biner tersebut dibuat *convex hull* dari kontur yang memiliki luasan terbesar. *Convex hull* hasil segmentasi warna lapangan ditunjukkan pada **Gambar 3.13**.

Citra RGB lapangan dan citra biner hasil segmentasi warna lapangan kemudian dipotong menjadi bentuk persegi. Pemotongan dimaksudkan agar resolusi citra menjadi simetris. Resolusi citra yang simetris sangat penting untuk tahap pengolahan citra digital selanjutnya, yaitu transformasi rotasi pada citra. Citra RGB kemudian diberi *masking* menurut citra biner lapangan dan kembali diberi *masking* mengikuti bentuk lingkaran virtual dengan diameter sama dengan lebar dan tinggi citra. Pemotongan citra ditunjukkan pada **Gambar 3.14**.

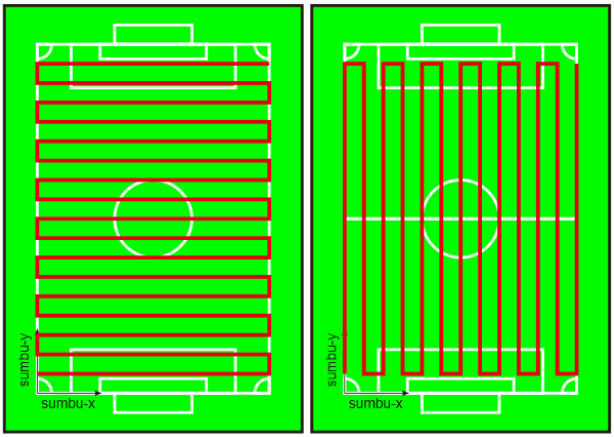
Gambar 3.14 Pemotongan citra lapangan

Gambar 3.15 Transformasi rotasi citra lapangan

Transformasi rotasi dilakukan pada citra yang sudah diberi *masking*. Besarnya rotasi tergantung pada orientasi robot. Transformasi rotasi dimaksudkan agar robot seolah-olah selalu menghadap depan (menghadap lapangan lawan). Pseudo-code untuk melakukan transformasi rotasi pada citra adalah sebagai berikut:

```
Mat M = getRotationMatrix2D(Point(center_x, center_y),
  kompas - 90, 1);
warpAffine(field, field, M, Size(x, y));
```

Transformasi rotasi pada citra lapangan dengan orientasi robot menghadap 135 derajat terhadap sumbu-x lapangan ditunjukkan pada **Gambar 3.15**.

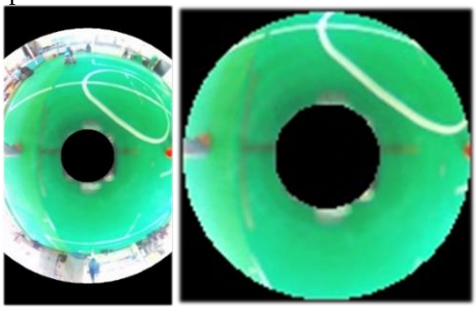


Gambar 3.16 Jalur pengambilan data kalibrasi posisi lokal

3.3.3 Dimensi Maksimal Lapangan

Citra masukan sistem memiliki resolusi sebesar 720x1280 pixel (rasio 9:16). Kamera pada sumbu-y memiliki sudut pandang yang luas sehingga seluruh citra lapangan pada sumbu-y tertangkap oleh kamera. Sebaliknya pada sumbu-x, kamera memiliki sudut pandang yang terbatas yang membuat jarak pandang kamera juga terbatas. Citra masukan tersebut diolah dan dipotong pada sumbu-y menyesuaikan dengan resolusi sumbu-x. Hasil akhir dari pengolahan citra masukan berupa citra keluaran dengan resolusi 120x120 pixel (rasio 1:1). Pemotongan citra berakibat pada penurunan jarak pandang kamera sumbu-y menjadi sama dengan jarak pandang kamera sumbu-x seperti ditunjukkan pada **Gambar 3.17**. Maka kemudian jarak pandang kamera pada sumbu-x dapat digunakan sebagai dasar penentuan dimensi lapangan terluas yang mungkin untuk sistem penentuan posisi lokal.

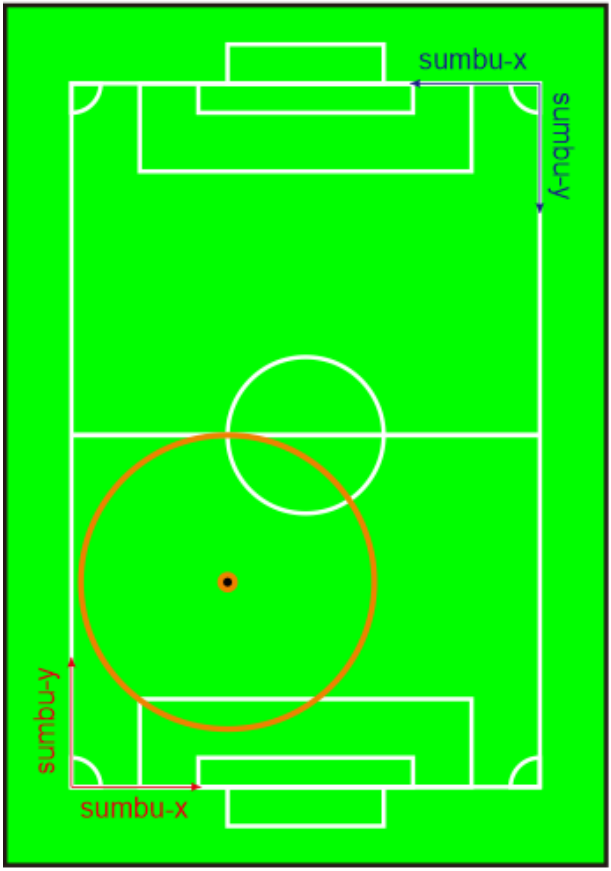
Pengukuran dilakukan untuk mengetahui jarak pandang kamera pada sumbu-x dan diketahui bahwa jarak pandang kamera pada sumbu-x sejauh 2,5 meter. Proses pengukuran jarak pandang kamera ditunjukkan pada **Gambar 3.18**.



Gambar 3.17 Pengolahan citra lapangan

Gambar 3.18 Proses pengukuran jarak pandang kamera

Agar sistem penentuan posisi lokal dapat berjalan dengan baik, pola marka lapangan dengan jarak terdekat pada lapangan harus berada dalam jangkauan kamera. Pola marka lapangan terdekat yang dimaksud adalah marka garis tengah lapangan dan marka kotak penalti lapangan; atau minimal ada satu pola marka tertentu yang terjangkau oleh jarak pandang kamera. Kemudian dimensi maksimal lapangan yang disarankan agar sistem penentuan posisi lokal berjalan dengan baik adalah 8x12 meter. Ilustrasi jangkauan kamera agar sistem penentuan posisi lokal berjalan dengan baik diilustrasikan pada **Gambar 3.19**.



Gambar 3.19 Ilustrasi dimensi maksimal lapangan yang disarankan

3.3.4 Pengambilan Data Kalibrasi

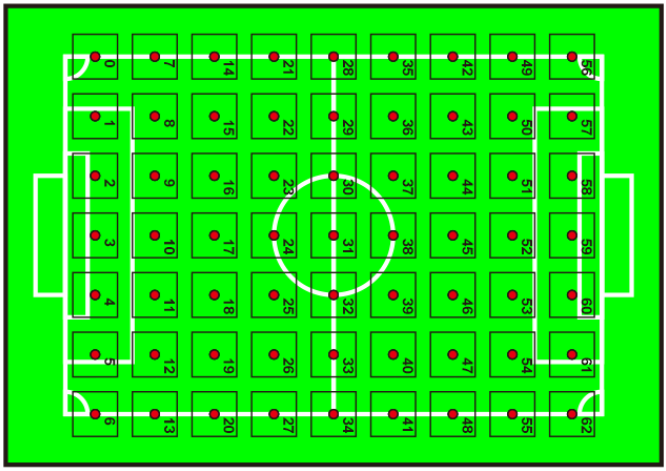
Data kalibrasi didapatkan dengan cara menjalankan robot secara otomatis dengan jalur tertentu sambil merekam citra lapangan dan koordinat robot. Hasil perekaman citra lapangan disimpan dalam file .avi dengan resolusi 360x360 pixel. Jalur pengambilan data kalibrasi posisi lokal dibuat mengular ditunjukkan pada **Gambar 3.16**. Kontrol posisi untuk menyusuri jalur menggunakan kontrol proporsional dan dibuat independen untuk kontrol posisi sumbu-x dan kontrol posisi sumbu-y.

Pseudo-code untuk merekam citra lapangan dan mencatat koordinat robot adalah sebagai berikut:

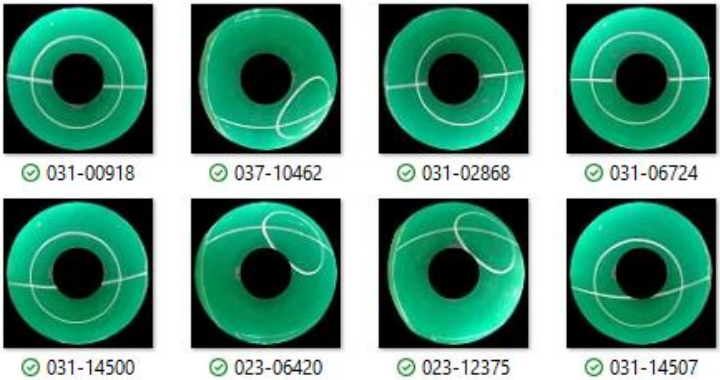
```
//-----Recorder and Logger
ofstream logger;
VideoWriter recorder;
unsigned char rec_status;
unsigned char previous_rec_status;

//Membuka file untuk merekam
if (rec_status && !previous_rec_status)
{
  string filename;
  filename = "C:\\IRIS\\cnn_log\\" +
    ofGetTimestampString("%H%M") + ".txt";
  logger.open(filename);
  filename = "C:\\IRIS\\cnn_log\\" +
    ofGetTimestampString("%H%M") + ".avi";
  recorder.open(filename, CV_FOURCC('M', 'J', 'P', 'G'),
    30, Size(x, x));
}
//Menutup file yang sudah terekam
if (!rec_status && previous_rec_status)
{
  logger.close();
  recorder.release();
}
//Menyimpan status terakhir perekaman
previous_rec_status = rec_status;

//Merekam CNN
if (rec_status)
{
  logger << posisi_x_global << '\t' << posisi_y_global
    << endl;
  recorder.write(field);
}
```

Pseudo-code kontrol proporsional posisi robot adalah sebagai berikut:

```
bool routine::jalan_global_posisi_sudut(
  float _target_x, float _target_y,
  float _target_sudut, int _vpos,
  int _vsud
)
{
  float error_x = _target_x - posisi_x_global;
  float output_x = KP_POS * error_x;
  float error_y = _target_y - posisi_y_global;
  float output_x = KP_POS * error_y;

  float error_sudut = _target_sudut – kompas;
  if (error_sudut > 180) error_sudut -= 360;
  if (error_sudut < -180) error_sudut += 360;
  float output_sudut = KP_SUD * error_sudut;

  jalan_manual_posisi(output_x, output_y,
    output_sudut);

  if (ofDist(posisi_x_global, posisi_y_global,
    _target_x, _target_y) < 20)
    return TRUE;
  else
    return FALSE;
}
```



Gambar 3.20 Pengelompokan citra lapangan menurut posisi robot



Gambar 3.21 File .jpg hasil pengelompokan citra

Video yang berisi citra lapangan untuk kalibrasi posisi lokal kemudian dicuplik setiap frame-nya dan dikelompokkan menjadi 63 kelas. Pengelompokan tersebut berdasarkan posisi robot saat merekam citra lapangan. Masing-masing kelas merujuk pada titik-titik tertentu di lapangan dengan toleransi sebesar 37.5cm dan dengan jeda jarak 100cm pada sumbu-x dan sumbu-y. Pengelompokan frame citra lapangan ditunjukkan pada **Gambar 3.20**. Kemudian resolusi masing-masing frame diubah menjadi 120x120 pixel dan disimpan dalam file .jpg dengan nama file sesuai dengan nomor kelasnya. Beberapa file .jpg hasil pengelompokan citra ditunjukkan pada **Gambar 3.21**.

3.3.5 Pelatihan Jaringan Syaraf Konvolusi

Penentuan posisi lokal robot menggunakan jaringan syaraf konvolusi atau *Convolutional Neural Network (CNN)*. Dalam sistem penentuan posisi robot ini, jaringan syaraf konvolusi berfungsi untuk melakukan klasifikasi citra lapangan terhadap 63 kelas posisi robot. Algoritma jaringan syaraf konvolusi dipilih karena tidak perlu mendefinisikan fitur citra yang akan dijadikan masukan sistem. Jaringan syaraf konvolusi akan menentukan sendiri fitur yang akan digunakan ketika proses pelatihan berlangsung. Keandalan jaringan syaraf konvolusi bergantung pada akurasi data kalibrasi. Jaringan syaraf konvolusi juga lebih tahan terhadap *noise* dengan adanya *pooling layer*.

Arsitektur jaringan syaraf konvolusi yang digunakan terdiri dari empat *convolution layer* dan *max pooling layer* ditambahkan pada setiap convolution layer tersebut. Keluaran dari *convolution layer* selanjutnya dimasukkan jaringan syaraf tiruan biasa yang terdiri dari satu *hidden layer* dengan *node* sebanyak 256 buah. Semua *hidden layer* pada jaringan syaraf konvolusi dan jaringan syaraf tiruan menggunakan fungsi aktivasi *ReLU* kecuali *output layer* jaringan syaraf tiruan menggunakan fungsi aktivasi *softmax*. *Pseudo-code* untuk membuat jaringan syaraf konvolusi adalah sebagai berikut:

```
cnn_model = Sequential()

cnn_model.add(Conv2D(filters=64, kernel_size=4,
  activation='relu', input_shape=(120,120,3)))
cnn_model.add(MaxPooling2D(pool_size=2))
cnn_model.add(Dropout(rate=0.25))

cnn_model.add(Conv2D(filters=64, kernel_size=4,
  activation='relu'))
cnn_model.add(MaxPooling2D(pool_size=2))
cnn_model.add(Dropout(rate=0.25))

cnn_model.add(Conv2D(filters=64, kernel_size=4,
  activation='relu'))
cnn_model.add(MaxPooling2D(pool_size=2))
cnn_model.add(Dropout(rate=0.25))

cnn_model.add(Conv2D(filters=64, kernel_size=4,
  activation='relu'))
cnn_model.add(MaxPooling2D(pool_size=2))
cnn_model.add(Dropout(rate=0.25))

cnn_model.add(Flatten())

cnn_model.add(Dense(units=256, activation='relu'))
cnn_model.add(Dense(units=63, activation='softmax'))

cnn_model.compile(optimizer='Adam',
  loss='categorical_crossentropy', metrics=['acc'])
```

Arsitektur jaringan syaraf konvolusi beserta jumlah variabel (keluaran, *weight* dan *bias*) pada masing-masing *layer* adalah sebagai berikut:

```
Layer (type)         Output Shape                  Param #
=================================================================
conv2d_1 (Conv2D)    (None, 117, 117, 64)          3136
_________________________________________________________________
max_pooling2d_1      (None, 58, 58, 64)            0
_________________________________________________________________
dropout_1 (Dropout)  (None, 58, 58, 64)            0
_________________________________________________________________
conv2d_2 (Conv2D)    (None, 55, 55, 64)            65600
_________________________________________________________________
max_pooling2d_2      (None, 27, 27, 64)            0
_________________________________________________________________
dropout_2 (Dropout)  (None, 27, 27, 64)            0
_________________________________________________________________
conv2d_3 (Conv2D)    (None, 24, 24, 64)            65600
_________________________________________________________________
max_pooling2d_3      (None, 12, 12, 64)            0
_________________________________________________________________
dropout_3 (Dropout)  (None, 12, 12, 64)            0
_________________________________________________________________
conv2d_4 (Conv2D)    (None, 9, 9, 64)              65600
_________________________________________________________________
max_pooling2d_4      (None, 4, 4, 64)              0
_________________________________________________________________
dropout_4 (Dropout)  (None, 4, 4, 64)              0
_________________________________________________________________
flatten_1 (Flatten)  (None, 1024)                  0
_________________________________________________________________
dense_1 (Dense)      (None, 256)                   262400
_________________________________________________________________
dense_2 (Dense)      (None, 63)                    16191
=================================================================
Total params: 478,527
Trainable params: 478,527
Non-trainable params: 0
```

Keseluruhan data kalibrasi yang berhasil dikumpulkan sejumlah 15.447 pasang citra dan kelas posisi robot. Data kalibrasi yang ada dipecah menjadi dua bagian, 10% data kalibrasi digunakan untuk proses pelatihan dan 90% data kalibrasi lainnya digunakan untuk proses validasi. Data kalibrasi tersebut dinormalisasi dengan cara membagi intensitas pixel dengan 255. Kelas data kalibrasi dikodekan dengan teknik *One-Hot Encoding*. Variabel kelas data kalibrasi

diubah dimensinya dari 1 dimensi menjadi 63 dimensi. Indeks yang bersesuaian dengan nilai kelas bernilai satu dan lainnya bernilai nol. *One-Hot Encoding* untuk 5 kelas data kalibrasi diilustrasikan pada **Tabel 3.1**.

Tabel 3.1 *One-Hot Encoding*

Kelas	Kelas (*One-Hot Encoding*)				
0	0	0	0	0	1
1	0	0	0	1	0
2	0	0	1	0	0
3	0	1	0	0	0
4	1	0	0	0	0

Proses pelatihan berjalan menggunakan GPU *NVIDIA GeForce GTX 1070*. Nilai *weight* dan *bias* yang menghasilkan nilai *error* terkecil pada setiap iterasi pelatihan disimpan pada file .hdf5. Pelatihan dapat ditunda dan dilanjutkan menggunakan file .hdf5 tersebut.

3.3.6 Implementasi Jaringan Syaraf Konvolusi

Pengimplementasian jaringan syaraf konvolusi untuk menentukan posisi lokal robot dilakukan dengan cara menjalankan dua program secara bersamaan. Satu program dalam bahasa *C++* digunakan untuk pengolahan citra digital dan satu program lainnya dalam bahasa *Python* digunakan untuk *feedforward* jaringan syaraf konvolusi. Kedua progra tersebut dihubungkan menggunakan protokol UDP dalam jaringan *localhost.*

Program pengolah citra digital mengirimkan data nilai pixel citra lapangan yang kemudian dijadikan masukan jaringan syaraf konvolusi. Nilai pixel yang dikirimkan ke program *feedforward* adalah sebesar 43.200 byte (120 pixel x 120 pixel x 3 kanal). Program *feedforward* kemudian mengirimkan keluaran jaringan syaraf konvolusi berupa derajat kepercayaan dari 63 kelas posisi terhadap citra yang dirambatkan kembali ke program *C++*. Posisi lokal robot kemudian dipilih dari kelas yang memiliki derajat kepercayaan paling besar.

.....Halaman ini sengaja dikosongkan.....

BAB 4
PENGUJIAN DAN ANALISA

Bab ini membahas mengenai pengujian dan analisa terhadap sistem yang sudah dirancang dan dibuat. Bab ini bertujuan untuk mengetahui apakah sistem dapat menyelesaikan masalah yang telah dirumuskan pada tugas akhir ini. Pengujian yang dilakukan meliputi pengujian sistem penentuan posisi global dan sistem penentuan posisi lokal robot. Pengujian sistem dilakukan dengan mengikuti batasan-batasan masalah yang telah dirumuskan pada tugas akhir ini.

4.1 Pengujian Posisi Global

Pengujian sistem penentuan posisi global dilakukan untuk mengetahui tingkat akurasi jaringan syaraf tiruan dalam mengestimasi koordinat centimeter robot yang sebenarnya terhadap koordinat pixel robot pada citra digital. Sebelum pengujian dilakukan, nilai ambang HSV diatur terlebih dahulu agar hasil segmentasi lapangan dan robot maksimal. Besaran nilai ambang HSV bergantung pada tingkat intensitas cahaya pada ruangan pengujian sehingga lapangan dan robot tersegmentasi dengan baik.

Data pengujian diperoleh dengan cara menempatkan robot pada titik-titik pada lapangan yang koordinatnya sudah diketahui. Kemudian koordinat centimeter robot keluaran dari jaringan syaraf tiruan dicatat sebagai hasil pengukuran.

4.1.1 Pengujian Segmentasi Lapangan

Pengujian segmentasi lapangan dilakukan di lapangan pertandingan robot sepak bola beroda Ruang Workshop Gedung Pusat Robotika. Pengujian dilakukan pada waktu malam untuk mendapatkan intensitas cahaya yang sama setiap hari. Citra lapangan dalam ruang wana RGB ditunjukkan pada **Gambar 4.1**. Citra biner hasil segmentasi warna lapangan dan *convex hull* yang diperoleh dari citra biner lapangan ditunjukkan pada **Gambar 4.2**. Hasil pemotongan dan normalisasi citra lapangan ditunjukkan pada **Gambar 4.3**.

Gambar 4.1 Citra lapangan dalam ruang warna RGB

Gambar 4.2 Citra biner dan *convex hull* lapangan

Gambar 4.3 Citra hasil pemotongan dan normalisasi

Gambar 4.4 Citra biner dan penentuan koordinat pixel robot

4.1.2 Pengujian Segmentasi Robot

Pengujian segmentasi robot dilakukan bersamaan dengan pengujian segmentasi lapangan. Kepala robot diberi penanda berwarna merah dan robot dalam keadaan diam. Citra biner hasil segmentasi warna robot dan hasil penentuan koordinat pixel robot ditunjukkan pada **Gambar 4.4**.

4.1.3 Pengujian *Feedforward* Jaringan Syaraf Tiruan

Pengujian posisi global robot dilakukan pada lima belas titik acak pada lapangan. Kemudian data posisi global keluaran jaringan syaraf tiruan dibandingkan dengan koordinat robot yang sebenarnya. Hasil pengujian sistem penentuan posisi global robot ditunjukkan pada **Tabel 4.1**. Dari pengujian, dapat diketahui bahwa sistem penentuan posisi global yang sudah dibuat memiliki rata-rata kesalahan sebesar 6.25cm.

Tabel 4.1 Data pengujian posisi global

Koordinat (cm)		Koordinat Terukur (cm)		*Error* (cm)		*Error* Koordinat (cm)
X	y	x	y	x	Y	
25	350	26	357	-0.62	-7.14	7.17
25	425	25	430	-0.28	-4.77	4.77
50	75	45	75	4.73	0.49	4.76
100	125	100	125	0.14	0.18	0.23
150	500	157	500	-6.52	0.47	6.54
175	75	177	78	-2.45	-2.73	3.67
175	225	180	229	-5.38	-4.38	6.94
250	75	254	76	-4.50	-0.87	4.58
250	375	251	380	-1.21	-5.21	5.35
400	175	407	178	-6.89	-2.70	7.40
475	225	482	228	-7.13	-2.99	7.73
500	575	507	580	-6.71	-5.24	8.51
575	250	584	250	-9.04	-0.04	9.04
575	350	581	354	-6.08	-4.24	7.41
600	350	603	352	-2.84	-1.75	3.33
RMSE (cm)				**5.11**	**3.59**	**6.25**

4.2 Pengujian Posisi Lokal

Pengujian sistem penentuan posisi lokal dilakukan untuk mengetahui tingkat akurasi jaringan syaraf konvolusi dalam mengestimasi koordinat centimeter robot yang sebenarnya terhadap citra digital yang dirambatkan. Sebelum pengujian dilakukan, nilai ambang HSV diatur terlebih dahulu agar hasil segmentasi lapangan maksimal. Besaran nilai ambang HSV bergantung pada tingkat intensitas cahaya pada ruangan pengujian sehingga lapangan tersegmentasi dengan baik.

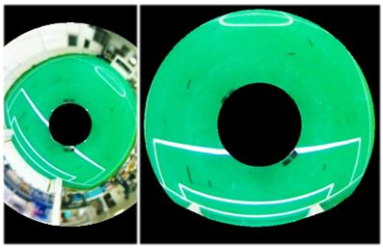
Data pengujian diperoleh dengan cara menjalankan robot secara otomatis pada 63 koordinat data kalibrasi posisi lokal dan dijaga pada koordinat tersebut selama 15 detik. Kemudian kelas koordinat keluaran jaringan syaraf konvolusi dengan derajat kepercayaan paling besar dibandingkan dengan kelas koordinat yang sebenarnya. Jumlah estimasi benar dan jumlah estimasi salah dicatat sebagai hasil pengukuran.

4.2.1 Pengujian Segmentasi Lapangan

Pengujian segmentasi lapangan dilakukan pada saat robot dalam keadaan diam. Sebelumnya lapangan sudah dibersihkan dari objek-objek yang dimungkinkan mengganggu proses pengolahan citra. Citra masukan dan keluaran segmentasi lapangan ditunjukkan padan **Gambar 4.5**.

4.2.2 Pengujian *Feedforward* Jaringan Syaraf Konvolusi

Pengujian sistem penentuan posisi lokal dilakukan dengan dua variasi, yaitu robot keadaan diam dan robot berjalan memutari koordinat dengan radius 25cm dan kecepatan 15cm/s. Hasil pengujian posisi lokal saat robot diam ditunjukkan pada **Tabel 4.2**. Hasil pengujian posisi lokal saat robot berjalan ditunjukkan pada **Tabel 4.3**. Dari pengujian, dapat diketahui bahwa akurasi sistem penentuan posisi lokal yang dibuat memiliki akurasi 100% pada saat robot diam dan 99,9% pada saat robot berjalan dengan kecepatan 15cm/s.



Gambar 4.5 Citra masukan dan keluaran segmentasi lapangan

Tabel 4.2 Data pengujian posisi lokal saat robot diam

Kelas	Benar	Salah	Akurasi	Kelas	Benar	Salah	Akurasi	Kelas	Benar	Salah	Akurasi
0	642	0	1	21	544	0	1	42	535	0	1
1	617	0	1	22	535	0	1	43	529	0	1
2	600	0	1	23	545	0	1	44	545	0	1
3	563	0	1	24	562	0	1	45	528	0	1
4	570	0	1	25	567	0	1	46	530	0	1
5	563	0	1	26	554	0	1	47	541	0	1
6	560	0	1	27	570	0	1	48	531	0	1
7	571	0	1	28	537	0	1	49	587	0	1
8	561	0	1	29	542	0	1	50	530	0	1
9	546	0	1	30	538	0	1	51	549	0	1
10	560	0	1	31	545	0	1	52	582	0	1
11	556	0	1	32	541	0	1	53	648	0	1
12	566	0	1	33	534	0	1	54	606	0	1
13	575	0	1	34	532	0	1	55	545	0	1
14	568	0	1	35	534	0	1	56	533	0	1
15	548	0	1	36	535	0	1	57	533	0	1
16	544	0	1	37	542	0	1	58	531	0	1
17	554	0	1	38	539	0	1	59	538	0	1
18	544	0	1	39	522	0	1	60	528	0	1
19	544	0	1	40	552	0	1	61	546	0	1
20	547	0	1	41	550	0	1	62	540	0	1
Total											
Benar: 34854				Salah: 0				Akurasi: 1,000			

Tabel 4.3 Data pengujian posisi lokal saat robot berjalan

Kelas	Benar	Salah	Akurasi	Kelas	Benar	Salah	Akurasi	Kelas	Benar	Salah	Akurasi
0	435	0	1	21	401	0	1	42	327	3	0.99
1	448	0	1	22	392	0	1	43	429	0	1
2	429	0	1	23	401	0	1	44	432	0	1
3	420	0	1	24	392	0	1	45	439	0	1
4	461	0	1	25	401	0	1	46	430	0	1
5	491	0	1	26	394	0	1	47	443	0	1
6	449	0	1	27	433	0	1	48	430	0	1
7	439	0	1	28	428	0	1	49	410	16	0.96
8	428	0	1	29	416	0	1	50	422	0	1
9	423	0	1	30	409	0	1	51	344	0	1
10	414	0	1	31	434	0	1	52	343	0	1
11	396	0	1	32	422	0	1	53	400	0	1
12	390	0	1	33	440	0	1	54	415	0	1
13	391	0	1	34	439	0	1	55	358	0	1
14	397	4	0.99	35	419	0	1	56	368	0	1
15	395	0	1	36	422	0	1	57	357	0	1
16	397	0	1	37	425	0	1	58	373	0	1
17	395	0	1	38	532	0	1	59	357	0	1
18	397	0	1	39	432	0	1	60	365	0	1
19	404	0	1	40	426	0	1	61	368	0	1
20	396	0	1	41	430	0	1	62	360	0	1
Total											
Benar: 25853				Salah: 23				Akurasi: 0,999			

.....Halaman ini sengaja dikosongkan.....

BAB 5
KESIMPULAN

Kesimpulan yang dapat diambil dari pembuatan sistem penentuan posisi robot sepak beroda berdasarkan pengindraan visual adalah sebagai berikut:

Sistem penentuan posisi global mampu memberikan data posisi dengan rata-rata *error* sebesar 6,25cm. Namun sistem ini sering kali gagal mendeteksi robot jika robot berada di pinggir lapangan. Hal ini dikarenakan kepala robot yang keluar dari jangkauan segmentasi lapangan. Kegagalan deteksi robot dapat diperbaiki dengan menggabungkan data posisi *odometry* dan data posisi global.

Sistem penentuan posisi lokal mampu memberikan data posisi dengan akurasi 100% saat robot diam dan 99,9% saat robot berjalan dengan kecepatan 15cm/s. Ketelitian sistem penentuan posisi lokal adalah sebesar 100cm. Ketelitian sistem dapat diperbaiki lagi dengan menambah titik pengambilan data kalibrasi.

.....Halaman ini sengaja dikosongkan.....

DAFTAR PUSTAKA

[1] "Data Science, Machine Learning, Data Visualization, Artificial Intelligence, Deep Learning - Google Trends," *Google Trends*. [Daring]. Tersedia pada: https://trends.google.co.id/trends/explore?date=today%205-y&q=Data%20Science,Machine%20Learning,Data%20Visualization,Artificial%20Intelligence,Deep%20Learning. [Diakses: 23-Apr-2018].

[2] A. Giusti *dkk.*, "A Machine Learning Approach to Visual Perception of Forest Trails for Mobile Robots," *IEEE Robotics and Automation Letters*, vol. 1, no. 2, hlm. 661–667, Jul 2016.

[3] J. Jin, M. Li, dan L. Jin, "Data Normalization to Accelerate Training for Linear Neural Net to Predict Tropical Cyclone Tracks," *Mathematical Problems in Engineering*, vol. 2015, hlm. 1–8, 2015.

[4] K.-S. Oh dan K. Jung, "GPU implementation of neural networks," *Pattern Recognition*, vol. 37, no. 6, hlm. 1311–1314, Jun 2004.

[5] R. C. Gonzalez dan R. E. Woods, *Digital image processing*, 3. ed., internat. ed. Upper Saddle River, NJ: Pearson, 2010.

[6] "OpenCV," *Wikipedia*. 09-Apr-2018.

[7] T. Munakata, *Fundamentals of the new artificial intelligence: neural, evolutionary, fuzzy and more*, 2. ed. London: Springer, 2008.

[8] "CS231n Convolutional Neural Networks for Visual Recognition." [Daring]. Tersedia pada: http://cs231n.github.io/neural-networks-3/. [Diakses: 09-Mei-2018].

[9] M. Abadi *dkk.*, "TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems," *arXiv:1603.04467 [cs]*, Mar 2016.

[10] "Keras Documentation." [Daring]. Tersedia pada: https://keras.io/. [Diakses: 16-Mei-2018].

[11] Y. Prakoso, "Desain Dan Implementasi Pengukuran Posisi Bola Menggunakan Kamera 360 Derajat Pada Robot Sepak Bola.," Institut Teknologi Sepuluh Nopember, 2017.

[12] A. Rachmawan, "Penentuan Posisi Robot Sepak Bola Beroda Menggunakan Rotary Encoder dan Kamera," Institut Teknologi Sepuluh Nopember, Surabaya, 2017.

[13] R. Dogaru dan I. Dogaru, "Optimization of GPU and CPU acceleration for neural networks layers implemented in python," 2017, hlm. 1–6.

[14] J. Cunha *dkk.*, "CAMBADA@Home'2012: Team Description Paper," dalam *Transverse Activity on Intelligent Robotics IEETA/DETI*, 2013.
[15] *Buku Panduan Kontes Robot Sepak Bola Indonesia Divisi Beroda 2018*. Kementrian Riset, Teknologi dan Perguruan Tinggi, 2017.

LAMPIRAN

A. Data Kalibrasi Posisi Global

No.	x	y	x'	y'	No.	x	y	x'	y'
1	0	50	46	603	27	0	250	15	478
2	50	50	71	611	28	50	250	42	483
3	100	50	102	619	29	100	250	77	490
4	150	50	135	627	30	150	250	114	493
5	200	50	168	629	31	200	250	155	497
6	250	50	206	630	32	250	250	198	500
7	300	50	241	630	33	300	250	242	500
8	350	50	277	630	34	350	250	286	498
9	400	50	315	625	35	400	250	328	493
10	450	50	346	619	36	450	250	368	488
11	500	50	378	610	37	500	250	403	481
12	550	50	405	603	38	550	250	437	475
13	600	50	433	596	39	600	250	463	470
14	0	150	28	548	40	0	350	3	395
15	50	150	56	555	41	50	350	31	400
16	100	150	89	561	42	100	350	67	403
17	150	150	124	568	43	150	350	106	403
18	200	150	161	576	44	200	350	150	406
19	250	150	202	578	45	250	350	195	406
20	300	150	241	577	46	300	350	240	407
21	350	150	283	574	47	350	350	288	405
22	400	150	321	571	48	400	350	334	404
23	450	150	358	563	49	450	350	375	400
24	500	150	392	554	50	500	350	411	395
25	550	150	422	548	51	550	350	445	393
26	600	150	450	538	52	600	350	472	389

No.	x	y	x'	y'	No.	x	y	x'	y'
53	0	450	2	308	79	0	650	17	147
54	50	450	29	306	80	50	650	45	141
55	100	450	64	303	81	100	650	78	132
56	150	450	104	303	82	150	650	115	126
57	200	450	148	300	83	200	650	154	120
58	250	450	194	300	84	250	650	195	116
59	300	450	242	300	85	300	650	238	115
60	350	450	290	300	86	350	650	280	116
61	400	450	334	300	87	400	650	321	120
62	450	450	375	300	88	450	650	359	125
63	500	450	413	300	89	500	650	395	130
64	550	450	445	300	90	550	650	426	137
65	600	450	472	300	91	600	650	454	144
66	0	550	5	222	92	0	750	33	84
67	50	550	33	218	93	50	750	59	75
68	100	550	67	212	94	100	750	92	67
69	150	550	106	207	95	150	750	125	59
70	200	550	148	200	96	200	750	162	54
71	250	550	193	200	97	250	750	200	51
72	300	550	239	200	98	300	750	238	50
73	350	550	285	200	99	350	750	277	52
74	400	550	328	200	100	400	750	314	56
75	450	550	370	204	101	450	750	348	60
76	500	550	403	207	102	500	750	383	67
77	550	550	438	214	103	550	750	412	76
78	600	550	467	218	104	600	750	438	82

No.	x	y	x'	y'
105	0	850	50	34
106	50	850	75	26
107	100	850	103	18
108	150	850	135	11
109	200	850	167	7
110	250	850	203	6
111	300	850	238	4
112	350	850	273	5
113	400	850	307	8
114	450	850	339	12
115	500	850	369	19
116	550	850	397	25
117	600	850	420	34

B. *Pseudo-code* pelatihan jaringan syaraf konvolusi

```
import os

import numpy as np
import pandas as pd
import cv2 as cv

import matplotlib.pyplot as plt

from sklearn.model_selection import train_test_split

from keras.models import Sequential, load_model
from keras.layers import Conv2D, MaxPooling2D, Dense,
  Dropout, Flatten
from keras.optimizers import Adam
from keras.callbacks import ModelCheckpoint
from keras.utils import to_categorical

images_path = './images/'

# Jumlah keseluruhan gambar yang ada dalam folder
n_images = len(os.listdir(images_path))
# Jumlah pixel dalam satu gambar
n_pixels = 120 * 120 * 3
```

```
# Menyiapkan buffer untuk membuat entry dataset
dataset = np.zeros((n_images, n_pixels + 1),
  dtype=np.uint8)

filelist = os.listdir(images_path)

for i in range(n_images):
    filename = filelist[i]

    dataset[i, 0] = int(filename[:3])

    image = cv.imread(images_path + filename)
    image = cv.resize(image, (120, 120))
    image = image.flatten()

    dataset[i, 1:] = image[0:]

# Membuat list untuk header
header = list()
# Menambah header untuk class gambar
header.append('class')
# Menambah header untuk pixel gambar
for i in range(n_pixels):
    header.append('pixel' + str(i))

df = pd.DataFrame(dataset, columns=header)
del dataset

# Memecah dataset untuk pelatihan dan validasi
train_df, test_df = train_test_split(df, test_size=0.1)
del df

train_data = np.array(train_df, dtype=np.float32)
test_data = np.array(test_df, dtype=np.float32)

del train_df
del test_df

# Normalisasi
x_train = train_data[:, 1:] / 255
y_train = train_data[:, 0]

del train_data

# Normalisasi
x_test = test_data[:, 1:] / 255
y_test = test_data[:, 0]

del test_data
```

```
x_train = x_train.reshape((x_train.shape[0], 120, 120, 3))
x_test = x_test.reshape((x_test.shape[0], 120, 120, 3))
y_train = to_categorical(y_train)
y_test = to_categorical(y_test)

print('x_train shape: {}'.format(x_train.shape))
print('x_test shape: {}'.format(x_test.shape))
print('y_train shape: {}'.format(y_train.shape))
print('y_test shape: {}'.format(y_test.shape))

cnn_model = Sequential()

cnn_model.add(Conv2D(filters=64, kernel_size=4,
  activation='relu', input_shape=(120,120,3)))
cnn_model.add(MaxPooling2D(pool_size=2))
cnn_model.add(Dropout(rate=0.25))

cnn_model.add(Conv2D(filters=64, kernel_size=4,
  activation='relu'))
cnn_model.add(MaxPooling2D(pool_size=2))
cnn_model.add(Dropout(rate=0.25))

cnn_model.add(Conv2D(filters=64, kernel_size=4,
  activation='relu'))
cnn_model.add(MaxPooling2D(pool_size=2))
cnn_model.add(Dropout(rate=0.25))

cnn_model.add(Conv2D(filters=64, kernel_size=4,
  activation='relu'))
cnn_model.add(MaxPooling2D(pool_size=2))
cnn_model.add(Dropout(rate=0.25))

cnn_model.add(Flatten())

cnn_model.add(Dense(units=256, activation='relu'))
cnn_model.add(Dense(units=63, activation='softmax'))

cnn_model.compile(optimizer='Adam',
  loss='categorical_crossentropy', metrics=['acc'])

cnn_model.summary()

modelcheckpoint = ModelCheckpoint(
    filepath='./models/model-{epoch:d}-{loss:f}
      {acc:f}.hdf5',
    monitor='loss',
    save_best_only=True,
    period=1
)
```

```
cnn_model.fit(
    x=x_train,
    y=y_train,
    validation_data=(x_test, y_test),
    epochs=999999,
    callbacks=[modelcheckpoint]
)
```

C. Data Kalibrasi Posisi Lokal

Data kalibrasi posisi lokal terdiri dari 15.447 file .jpg dengan ukuran keseluruhan sebesar 213MB. Data kalibrasi tersebut tidak dapat dilampirkan dalam buku sehingga dilampirkan melalui tautan https://drive.google.com/file/d/1NFL_cJ80G2fCuQjMsorOMN-1DKeF0Syn/view?usp=sharing.

D. *Pseudo-code feedforward* jaringan syaraf tiruan

```
import socket
import keras
import numpy as np

udp_tx = ('127.0.0.1', 4040)
udp_rx = ('0.0.0.0', 4041)

# merangkai otak yang pintar
model = keras.models.load_model('./model.hdf5')

# membuat jalan menuju dunia luar
udp = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
# mendengarkan apa yang mereka bicarakan
udp.bind(udp_rx)

print('===== Global Position Estimator =====')

while True:
    data_rx, address_rx = udp.recvfrom(8)

    input = np.fromstring(data_rx[0:8], np.float32)
    input = input.reshape((1, 2)) / 1000

    output = model.predict(input)
    output = output.reshape((2)) * 1000

    udp.sendto(output.tobytes(), udp_tx)
```

BIODATA PENULIS

Penulis dilahirkan dengan nama Pandu Surya Tantra pada tanggal 21 Oktober 1997 di Kediri, Jawa Timur. Penulis lulus dari SMP Negeri 1 Kediri pada tahun 2012 dan lulus dari SMA Negeri 2 Kediri pada tahun 2014. Saat ini penulis tengah menempuh semester akhir pendidikan Strata Satu di Departemen Teknik Elektro, Bidang Studi Elektronika, Institut Teknologi Sepuluh Nopember, Surabaya. Saat menjalani masa perkuliahan, penulis aktif dalam kegiatan Workshop UKM Robotika ITS. Selain itu, penulis juga aktif sebagai anggota Tim Robot ITS dan telah mendapatkan berbagai prestasi baik di tingkat regional maupun nasional. Sejalan dengan semua kegiatan tersebut, penulis juga aktif sebagai asisten Laboratorium Elektronika Dasar B-202. Penulis sangat tertarik dengan riset di bidang robotika dan kecerdasan buatan. Penulis memiliki cita-cita mengembangkan teknologi robotika Indonesia agar dapat bersaing di tingkat internasional.

e-mail : pandustantra@gmail.com
LinkedIn : linkedin.com/in/pandustantra

.....Halaman ini sengaja dikosongkan.....