



UNIVERSITAS INDONESIA

**PERANCANGAN DAN IMPLEMENTASI SISTEM
PENDETEKSIAN DAN PELACAKAN OBJEK BOLA
BERDASARKAN JARINGAN SARAF TIRUAN**

SKRIPSI

FATAH ABDUL WAHAB

1506673681

FAKULTAS TEKNIK UNIVERSITAS INDONESIA

DEPARTEMEN TEKNIK ELEKTRO

PROGRAM STUDI TEKNIK KOMPUTER

DEPOK

JANUARI 2021



UNIVERSITAS INDONESIA

**PERANCANGAN DAN IMPLEMENTASI SISTEM
PENDETEKSIAN DAN PELACAKAN OBJEK BOLA
BERDASARKAN JARINGAN SARAF TIRUAN**

SKRIPSI

**Diajukan sebagai salah satu syarat untuk memperoleh gelar
Sarjana Ilmu Komputer**

Fatah Abdul Wahab

1506673681

FAKULTAS TEKNIK UNIVERSITAS INDONESIA

DEPARTEMEN TEKNIK ELEKTRO

PROGRAM STUDI TEKNIK KOMPUTER

DEPOK

JANUARI 2021

HALAMAN PERNYATAAN ORISINALITAS

**Skripsi ini merupakan hasil dari karya saya sendiri
dan semua sumber baik yang dikutip maupun dirujuk
telah saya nyatakan dengan benar.**

Nama : Fatah Abdul Wahab

NPM : 1506673681

Tanda Tangan :



Tanggal : 23 Januari 2021

LEMBAR PENGESAHAN

Skripsi ini diajukan oleh:

Nama : Fatah Abdul Wahab

NPM : 1506673681

Program Studi : Teknik Komputer

Judul Skripsi : Perancangan dan Implementasi Sistem Pendeteksian dan
Pelacakan Objek Bola Berdasarkan Jaringan Saraf Tiruan

Telah berhasil dipertahankan di hadapan Dewan Penguji dan diterima sebagai bagian persyaratan yang diperlukan untuk memperoleh gelar Sarjana Teknik pada program Studi Teknik Komputer, Fakultas Teknik, Universitas Indonesia

DEWAN PENGUJI

Pembimbing : Dr. Prima Dewi Purnamasari, S.T., M.T., M.Sc.



Penguji : I Gde Dharma Nugraha, S.T., M.T., Ph.D.



Penguji : Dr. Eng. Mia Rizkinia, S.T, M.T.



Ditetapkan di : Depok

Tanggal : 23 Januari 2021

KATA PENGANTAR

Puji syukur penulis panjatkan kepada Allah SWT, karena atas berkat dan rahmat-Nya penulis dapat menyelesaikan skripsi ini. Penulisan skripsi ini dilakukan dalam rangka memenuhi salah satu syarat untuk memperoleh gelar Sarjana Program Studi Teknik Komputer pada Fakultas Teknik Universitas Indonesia. Penulis menyadari bahwa tanpa adanya bantuan dari berbagai pihak, akan sangat sulit bagi penulis untuk menyelesaikan skripsi ini. Oleh karena itu, penulis mengucapkan terima kasih kepada :

1. Ibu Prima Dewi Purnamasari S.T., M.Sc. selaku dosen pembimbing yang telah menyediakan waktu, tenaga, dan pikiran untuk membimbing penulis dalam penyusunan skripsi ini.
2. Kedua Orangtua dan keluarga penulis yang telah memberikan banyak bantuan berupa dukungan material dan moral.
3. Seluruh rekan anggota Tim Robotika UI yang telah memberikan kesempatan bagi penulis untuk belajar, bermain, bersedih, dan berbahagia bersama.
4. Segenap teman-teman Teknik Komputer UI angkatan 2015 atas masukan, dukungan, dan ilmu untuk penulis dalam menyelesaikan skripsi ini.

Akhir kata, saya berharap Allah SWT berkenan membalas segala kebaikan semua pihak yang telah membantu. Semoga skripsi ini membawa manfaat bagi pengembangan ilmu pengetahuan dan teknologi untuk selanjutnya.

Depok, 24 Januari 2021

Fatah Abdul Wahab

**HALAMAN PERNYATAAN PERSETUJUAN PUBLIKASI TUGAS
AKHIR UNTUK KEPENTINGAN AKADEMIS**

Sebagai sivitas akademik Universitas Indonesia, saya yang bertanda tangan di bawah ini:

Nama : Fatah Abdul Wahab

NPM : 1506673681

Program Studi : Teknik Komputer

Departemen : Teknik Elektro

Fakultas : Teknik

Jenis karya : Skripsi

Demi pengembangan ilmu pengetahuan, menyetujui untuk memberikan kepada Universitas Indonesia **Hak Bebas Royalti Noneksklusif (*Non-exclusive Royalty-Free Right*)** atas karya ilmiah saya yang berjudul:

**PERANCANGAN DAN IMPLEMENTASI SISTEM
PENDETEKSIAN DAN PELACAKAN OBJEK BOLA
BERDASARKAN JARINGAN SARAF TIRUAN**

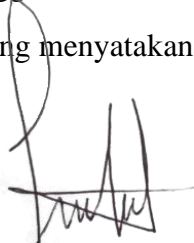
beserta perangkat yang ada (jika diperlukan). Dengan Hak Bebas Royalti Noneksklusif ini Universitas Indonesia berhak menyimpan, mengalihmedia/formatkan, mengelola dalam bentuk pangkalan data (*database*), merawat, dan mempublikasikan tugas akhir saya selama tetap mencantumkan nama saya sebagai penulis/pencipta dan sebagai pemilik Hak Cipta.

Demikian pernyataan ini saya buat dengan sebenarnya.

Dibuat di : Depok

Pada tanggal : 23 Januari 2021

Yang menyatakan,



(Fatah Abdul Wahab)

ABSTRAK

Nama : Fatah Abdul Wahab
Program Studi : Teknik Komputer
Judul : Perancangan dan Implementasi Sistem Pendeteksian dan Pelacakan Objek Bola Berdasarkan Jaringan Saraf Tiruan

Computer vision merupakan cabang dari bidang ilmu kecerdasan buatan yang mempelajari bagaimana sebuah komputer dapat memahami suatu gambar yang diberikan. Salah satu contoh nyata dari penerapan *computer vision* adalah pengenalan objek bola pada robot sepak bola. Salah satu tantangan yang dapat menyulitkan robot dalam mendeteksi bola adalah warna bola yang sebagian besar berwarna putih. Hal ini menjadi tantangan karena warna putih sangat rentan terhadap *noise*. Metode tradisional yang hanya dapat mendeteksi satu bentuk saja tidak cukup untuk memenuhi kebutuhan tersebut, karenanya digunakan pendeteksian berdasarkan *machine learning*. Salah satu metode pengenalan objek berdasarkan *machine learning* yang sering digunakan adalah metode Jaringan Saraf Tiruan. Pada tulisan ini, sistem penglihatan robot sepak bola untuk mengenali objek bola dirancang menggunakan metode jaringan saraf tiruan dengan *library* pengolahan citra OpenCV dalam bahasa pemrograman C++. Berdasarkan pengujian kinerja sistem dalam mendeteksi bola pada gambar mendapatkan nilai *accuracy* sebesar 0.9987, nilai *precision* sebesar 0.8055, nilai *recall* sebesar 0.7, dan FPS sebesar 6. Sedangkan kinerja sistem pembanding dengan menggunakan SVM pada dataset yang sama mendapatkan nilai *accuracy* sebesar 0.988, nilai *precision* sebesar 0.167, nilai *recall* sebesar 0.966, dan FPS sebesar 7,7. Setelah kedua metode dibandingkan dapat disimpulkan bahwa metode jaringan saraf tiruan dapat mendeteksi bola lebih akurat berdasarkan nilai *F-Score* yang didapatkan yaitu 0.749 pada sistem yang dibuat berbanding dengan 0.285 pada sistem pembanding, namun memerlukan waktu proses yang lebih lama.

Kata Kunci:

Computer vision, robot sepak bola, deep learning, jaringan saraf tiruan, sistem pengenalan objek, SVM, OpenCV, C++.

ABSTRACT

Name : Fatah Abdul Wahab
Study Program : Teknik Komputer
Title : Design and Implementation of Object Detection and Tracking Systems for Ball Object Based on Neural Network

Computer vision is a branch of the field of artificial intelligence that studies how a computer can understand a given image. An example of the application of computer vision is detecting a ball object on a soccer robot. One of the challenges that can make it difficult for the robot to detect the ball is the color of the ball, which is mostly white. This becomes a challenge because white is very susceptible to noise. Traditional methods that can only detect one form are not sufficient to meet these needs, therefore detection based on machine learning is used. One of the object detection methods based on machine learning that is often used is the Artificial Neural Network method. In this paper, the system to detect ball object is implemented using an artificial neural network method with the OpenCV image processing library in the C ++ programming language. Based on testing the performance of the system at detecting ball have the accuracy value of 0.9987, precision value of 0.8055, recall value of 0.7, and FPS of 6. While the performance of the comparison system using SVM on the same dataset gets accuracy value of 0.988, precision value of 0.167, recall value of 0.966, and FPS of 7.7. After the two methods were compared, it can be concluded that the artificial neural network method can detect the ball more accurately based on the F-Score value obtained, which is 0.749 compared to 0.285, but it requires a longer processing time.

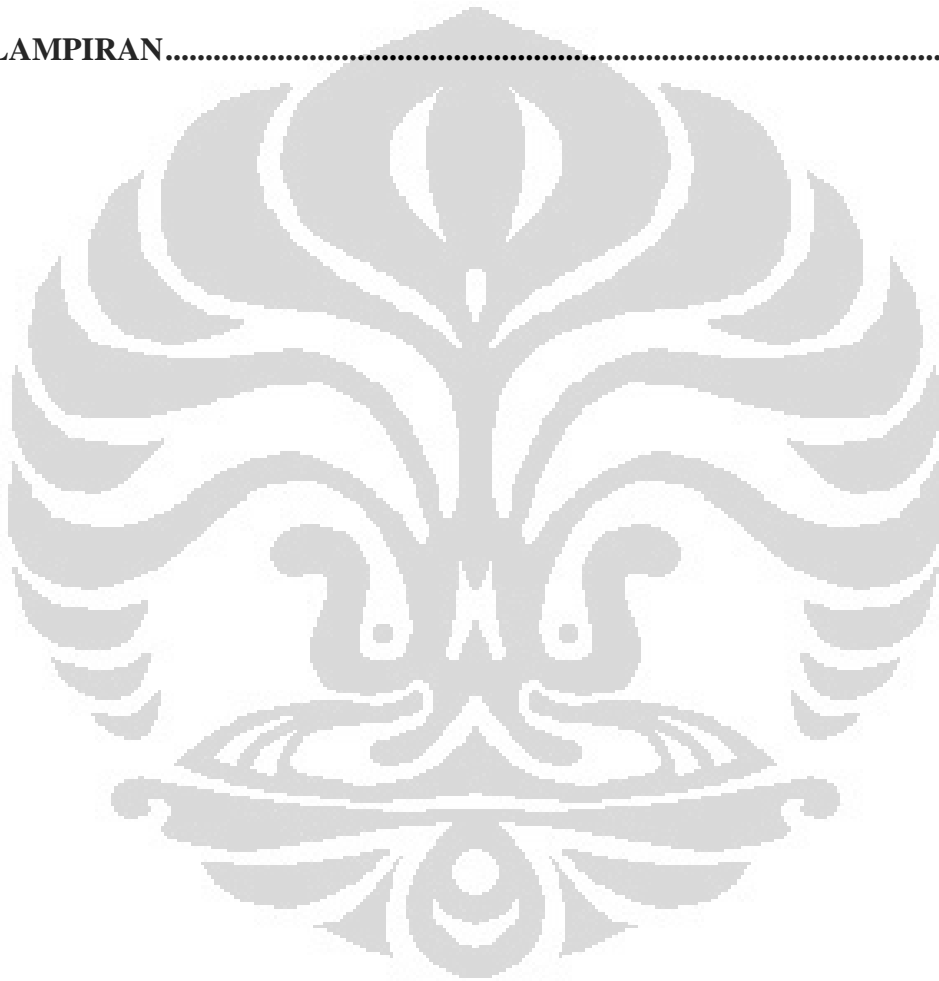
Key words:

Computer vision, soccer robot, deep learning, artificial neural networks, object recognition systems, SVM, OpenCV, C ++.

DAFTAR ISI

HALAMAN PERNYATAAN ORISINALITAS	ii
LEMBAR PENGESAHAN	iii
KATA PENGANTAR.....	iv
HALAMAN PERNYATAAN PERSETUJUAN PUBLIKASI TUGAS AKHIR UNTUK KEPENTINGAN AKADEMIS	v
ABSTRAK	vi
ABSTRACT	vii
DAFTAR ISI.....	viii
DAFTAR GAMBAR.....	x
BAB 1 PENDAHULUAN	1
1.1. Latar Belakang	1
1.2. Tujuan Penelitian	3
1.3. Batasan Penelitian	4
1.4. Metodologi Penelitian	4
1.5. Sistematika Penulisan	5
BAB 2 LANDASAN TEORI	6
2.1. Jaringan Saraf Tiruan	6
2.2. <i>Histogram of Oriented Gradients (HOG)</i>	8
2.3. <i>Multiscale Sliding Window</i>	8
2.4. <i>Support Vector Machine (SVM)</i>	9
2.5. OpenCV	10
2.6. FANN (Fast Artificial Neural Network)	10
BAB 3 PERANCANGAN SISTEM	13
3.1. Perancangan Kebutuhan Sistem	13
3.2. Perancangan Perangkat Lunak	14
3.2.1. Perancangan Pembelajaran Jaringan Saraf Tiruan	15
3.2.2. Perancangan Proses Pendeteksian dan Pelacakan	17
3.3. Perancangan Pengumpulan Data	21
3.4. Perancangan Pengujian	23
3.4.1. Perancangan Pengujian Program Pembelajaran	23
3.4.2. Perancangan Pengujian Program Pendeteksian dan Pelacakan ..	24

BAB 4 PENGUJIAN DAN ANALISA	27
4.1. Pengujian Program Pembelajaran	27
4.1.1. Pengujian Variasi Jumlah <i>Node</i> Pada <i>Hidden Layer</i>	27
4.1.2. Pengujian Variasi Laju Pembelajaran	28
4.1.3. Pengujian Variasi Momentum Pembelajaran.....	29
4.2. Pengujian Program Pendeteksian Bola	29
BAB 5 KESIMPULAN	35
DAFTAR PUSTAKA	36
LAMPIRAN.....	38



DAFTAR GAMBAR

Gambar 1.1 Contoh robot sepak bola yang sedang bertanding.....	1
Gambar 2.1 Ilustrasi <i>node</i> pada jaringan saraf tiruan	6
Gambar 2.2 Contoh visualisasi dari SVM.....	9
Gambar 3.1 Diagram blok proses pembelajaran BPNN	15
Gambar 3.2 Gambar Alur Proses Pendeteksian	18
Gambar 3.3 Ilustrasi Pergerakan Kotak Deteksi	19
Gambar 4.1 Contoh hasil pengujian pendeteksian dengan JST	32
Gambar 4.2 Contoh hasil pengujian pendeteksian dengan SVM.....	33
Gambar 4.3 Contoh proses pelacakan bola	34

BAB 1

PENDAHULUAN

1.1. Latar Belakang

Artificial Intelligence (AI) atau kecerdasan tiruan adalah salah satu bidang ilmu yang sedang mengalami perkembangan pesat saat ini. Kecerdasan tiruan adalah kemampuan sebuah sistem yang telah diprogram untuk dapat belajar dan berpikir seperti manusia [1]. Dengan kemampuan tersebut, sistem dapat digunakan untuk menyelesaikan berbagai masalah seperti layaknya manusia. Salah satu contoh dimana kecerdasan tiruan dapat digunakan untuk menyelesaikan masalah adalah pada bidang *computer vision* atau penglihatan komputer.

Computer vision merupakan bidang ilmu yang mempelajari bagaimana sebuah komputer dapat memahami suatu gambar yang diberikan [2]. Penerapan dari *computer vision* yang sering digunakan adalah untuk mengenali suatu objek tertentu. Agar dapat mengenali suatu objek, komputer harus mengetahui ciri-ciri dari objek yang ingin dilihat. Dalam hal ini kecerdasan tiruan digunakan oleh komputer untuk mempelajari ciri-ciri dari suatu objek tersebut, sehingga ketika komputer melihat objek dengan ciri yang sama, komputer akan mengenalinya. Salah satu contoh nyata dari penerapan ini adalah pengenalan objek bola pada robot sepak bola.



Gambar 1.1 Contoh robot sepak bola yang sedang bertanding

Seperti namanya, robot sepak bola adalah salah satu jenis robot yang dapat melakukan pertandingan sepak bola seperti manusia. Tugas utama dari robot sepak bola adalah untuk mendeteksi bola lalu membawa atau menendang bola tersebut sampai masuk ke gawang lawan. Pada proses pendeteksian bola inilah *computer vision* digunakan sehingga robot dapat mengenali objek bola tersebut dan menjejarnya.

Salah satu tantangan yang dapat menyulitkan robot dalam mendeteksi bola adalah warna bola yang sebagian besar berwarna putih. Hal ini menjadi tantangan karena warna putih sangat rentan terhadap *noise*, dimana banyak warna putih lainnya yang dapat ditemukan di lapangan dan sekitarnya, sehingga pendeteksian tidak cukup hanya mengandalkan warna dan harus dapat mengenali bentuk bola. Sistem pengenalan bentuk yang dibuat harus fleksibel serta akurat, karena terdapat kemungkinan bahwa bola terhalang benda lain sehingga bentuk bola dapat terlihat berubah dan tidak bulat sempurna. Metode tradisional yang hanya dapat mendeteksi satu bentuk saja tidak cukup untuk memenuhi kebutuhan tersebut, karenanya digunakan pendeteksian berdasarkan *machine learning*.

Salah satu metode pengenalan objek berdasarkan *machine learning* yang sering digunakan adalah metode Jaringan Saraf Tiruan. Jaringan Saraf Tiruan (JST) adalah sistem komputasi yang dibentuk berdasarkan cara kerja neuron pada manusia [3]. Metode ini dapat melakukan pembelajaran terhadap suatu masalah, sehingga masalah yang sulit diselesaikan dengan persamaan matematis dapat terselesaikan. JST juga sangat terkenal untuk digunakan pada *computer vision*. Dengan proses pembelajaran gambar, JST dapat mengenali suatu objek yang telah dipelajari dengan tingkat akurasi yang cukup tinggi. Dengan sistem pembelajaran ini, robot diharapkan dapat mengenali bola dengan lebih akurat dibandingkan sistem pendeteksian tradisional.

Pembelajaran gambar dengan Jaringan Saraf Tiruan memerlukan sebuah *feature descriptor* atau suatu algoritma yang dapat menjelaskan ciri-ciri dari suatu gambar agar dapat dimengerti oleh komputer [4]. Salah satu *feature descriptor* yang dapat digunakan dalam pembelajaran gambar adalah *Histogram of Oriented Gradients* atau disingkat menjadi HOG. *Feature descriptor* ini menggunakan

kumpulan dari *gradients* atau perubahan intensitas warna pada gambar, beserta dengan arah perubahannya untuk menjelaskan bentuk dari gambar tersebut [5].

Namun tidak hanya pengenalan objek yang diperlukan oleh *humanoid robot*. Karena robot dianggap dapat bergerak, mengenali objek tidaklah cukup agar robot dapat bergerak menuju objek tersebut. Diperlukan juga suatu metode untuk mengetahui posisi koordinat objek pada pandangan. Hal ini dapat dilakukan dengan menggabungkan metode pengenalan objek dengan metode pencarian objek yang dikenali. Salah satu metode pencarian yang dapat digunakan adalah *Multiscale Sliding Window*. Metode ini menggunakan sebuah *window* atau kotak deteksi yang akan digerakkan secara bertahap (*sliding*) sampai keseluruhan gambar dicari [6]. Jika keseluruhan gambar telah dicari, kemudian ukuran gambar akan diubah, hal ini dilakukan agar keseluruhan kemungkinan ukuran objek dapat dicari. Posisi koordinat dari kotak deteksi yang memiliki objek kemudian dapat digunakan sebagai acuan untuk pergerakan robot sehingga robot dapat bergerak menuju objek tersebut.

Pada penelitian ini, akan dilakukan perancangan serta penerapan sistem pengenalan dan pelacakan objek pada *humanoid robot* dengan metode pembelajaran Jaringan Saraf Tiruan, *Histogram of Oriented Gradients* sebagai *feature descriptor*, dan metode pelacakan *multiscale sliding window*. Dengan metode yang digunakan tersebut, diharapkan robot dapat mendeteksi dan melacak objek dengan tingkat akurasi yang tinggi meskipun objek dan robot bergerak.

1.2. Tujuan Penelitian

Penelitian yang dilakukan memiliki tujuan:

1. Menerapkan algoritma *computer vision* menggunakan jaringan saraf tiruan untuk mendeteksi objek bola.
2. Membuat dan menerapkan sistem untuk melacak objek bola.
3. Mengetahui tingkat keberhasilan pendeteksian dari algoritma yang dibuat.

1.3. Batasan Penelitian

Penelitian ini berfokus pada pembuatan dan penerapan algoritma pendeteksian objek pada robot humanoid. Penelitian ini memiliki pembatasan masalah sebagai berikut:

1. Algoritma yang dibuat hanya sebatas untuk pendeteksian dan pelacakan gerakan objek.
2. Data yang digunakan untuk proses pembelajaran program diambil dari pengambilan foto objek yang bersangkutan serta publikasi data bola dari salah satu tim peserta RoboCup.
3. Kinematika pergerakan badan robot untuk bergerak mengikuti objek tidak dibahas pada penelitian ini.
4. Penelitian diterapkan pada *processing unit* berupa Laptop Lenovo Thinkpad T430u.
5. Kamera yang digunakan pada penelitian ini adalah sebuah kamera dari handphone Xiaomi 5c.

1.4. Metodologi Penelitian

Metodologi penelitian yang digunakan mengadopsi dari *Waterfall System Development Methodologies*, yang meliputi:

1. *Requirement Definition*: menentukan spesifikasi kebutuhan sistem dengan melakukan studi pustaka.
2. *System Design*: merancang sistem perangkat lunak yang akan dibuat.
3. *Implementation*: memprogram sistem perangkat lunak yang sudah dirancang.
4. *Testing and Evaluation*: melakukan pengujian terhadap performa sistem yang dibuat dengan skenario yang sudah ditentukan serta melakukan analisis dari hasil pengujian system.

1.5. Sistematika Penulisan

Sistematika dari penulisan buku seminar ini terbagi menjadi lima bab yang terdiri dari:

BAB 1 PENDAHULUAN: Memberikan informasi mengenai latar belakang penelitian, tujuan penelitian, manfaat penelitian, Batasan penelitian, metodologi penelitian, dan sistematika penulisan dari seminar ini.

BAB 2 LANDASAN TEORI: Menjelaskan hasil studi pustaka mengenai topik yang berkaitan dengan bahasan penelitian.

BAB 3 PERANCANGAN SISTEM: Menjelaskan hasil perancangan sistem yang dibuat berdasarkan studi pustaka.

BAB 4 PENGUJIAN DAN ANALISA: Menyajikan hasil penelitian berupa hasil implementasi algoritma pada sistem, hasil uji coba sesuai dengan skenario yang sudah dibuat, serta analisis terhadap hasil implementasi dan uji coba yang dilakukan.

BAB 5 KESIMPULAN: Memberikan kesimpulan dari penelitian yang digunakan.

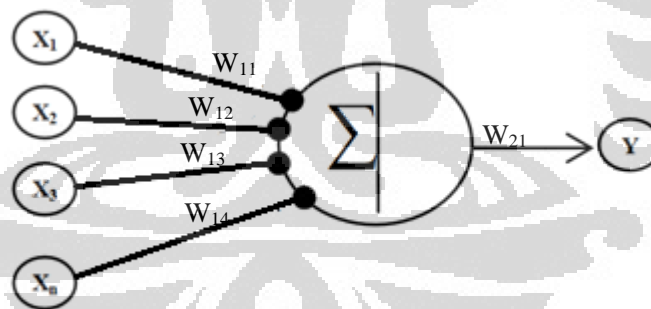
BAB 2

LANDASAN TEORI

2.1. Jaringan Saraf Tiruan

Jaringan saraf tiruan adalah sebuah sistem komputasi yang dibentuk berdasarkan cara kerja otak manusia. Jaringan saraf tiruan merupakan salah satu penerapan dari *Artificial Intelligence* atau kecerdasan buatan yang dapat digunakan untuk memecahkan berbagai masalah komputasi di kehidupan nyata. Dengan jaringan saraf tiruan, sistem dapat belajar sampai menemukan solusi yang optimal untuk suatu masalah.

Cara kerja dari jaringan saraf tiruan adalah dengan membuat jaringan neuron tiruan yang disebut dengan *node*. Masing-masing dari *node* akan mengolah suatu informasi dan meneruskan informasi tersebut ke *node* selanjutnya sampai dengan *node* terakhir. Kumpulan dari beberapa node yang sama akan membentuk sesuatu yang disebut dengan *layer* atau lapisan. Adapun jenis *layer* yang ada pada jaringan saraf tiruan ada tiga yaitu *input layer*, *hidden layer*, dan *output layer*.



Gambar 2.1 Ilustrasi *node* pada jaringan saraf tiruan

Pada Gambar 2.1 diperlihatkan ilustrasi sederhana dari tiap-tiap *node* yang ada pada sebuah jaringan saraf tiruan. Alur dari sebuah data yang akan masuk ke dalam jaringan adalah masuk melalui *node input* menjadi data X , kemudian dikalikan dengan bobot W yang terhubung pada masing-masing *node*, dan diterapkan fungsi aktivasi pada *hidden layer* lalu data dikalikan lagi dengan bobot W yang menghubungkan *layer hidden* dan *output* sehingga menghasilkan nilai

keluaran Y di *node output*. Jaringan saraf yang memiliki struktur sederhana seperti ini disebut dengan jaringan *feed forward*.

Namun untuk mendapatkan nilai keluaran Y yang diinginkan, nilai bobot W harus disesuaikan sedemikian rupa sehingga dapat menghasilkan nilai yang mendekati nilai yang diinginkan. Untuk dapat mewujudkan hal ini dapat diterapkan salah satu metode pembelajaran pada jaringan saraf yang disebut *backpropagation*. Secara sederhana, cara kerja *backpropagation* dapat dijelaskan dengan menghitung nilai *error* dari keluaran yang dihasilkan *node Y* dengan keluaran yang diinginkan, kemudian menggunakan nilai *error* tersebut untuk melakukan perhitungan perubahan bobot sampai menghasilkan nilai *error* yang paling rendah. Semakin rendah nilai *error* maka hasil keluaran dari jaringan akan semakin mendekati nilai keluaran yang diinginkan.

Adapun detail dari proses *backpropagation* mulai dari proses *feed forward* adalah sebagai berikut:

1. Melakukan proses *feed forward* pada jaringan hingga menghasilkan output.
2. Menghitung nilai *error* dengan *loss function* berupa *mean square error*.
3. Menghitung turunan parsial *loss function* atau gradient pada bobot layer keluaran - hidden.
4. Menghitung nilai perubahan bobot layer keluaran – hidden.
5. Menghitung turunan parsial *loss function* atau gradient pada layer hidden - input.
6. Menghitung nilai perubahan bobot layer hidden – input.
7. Mengulangi tahap 1 – 6 sampai terpenuhi *stopping condition* yang dapat ditentukan oleh pengguna.

Setelah *stopping condition* terpenuhi, seluruh nilai bobot dari jaringan saraf akan disimpan sehingga dapat digunakan sebagai jaringan *feed forward* untuk implementasi nyata.

2.2. *Histogram of Oriented Gradients (HOG)*

Histogram of Oriented Gradients adalah sebuah *feature descriptor* yang digunakan dalam pendeteksian objek dengan *computer vision*. Cara kerja *Histogram of Oriented Gradients* adalah dengan menghitung orientasi dari gradient yang ada pada suatu bagian gambar. Pada HOG, histogram atau distribusi dari arah gradient digunakan sebagai fitur penting yang diambil. Dengan metode ini, pendeteksian dapat dilakukan meskipun kondisi pencahayaan yang kurang baik.

Algoritma untuk mengimplementasikan HOG secara singkat dapat dijelaskan sebagai berikut:

1. Melakukan preprocessing pada gambar.
2. Menghitung gradient dari gambar.
3. Memecah gambar menjadi potongan kecil berukuran 8x8, dan untuk masing-masing potongan dihitung histogram dari gradient tersebut.
4. Melakukan normalisasi pada histogram setelah memecah gambar menjadi beberapa potongan berukuran 16x16.
5. Menghitung vector hasil.

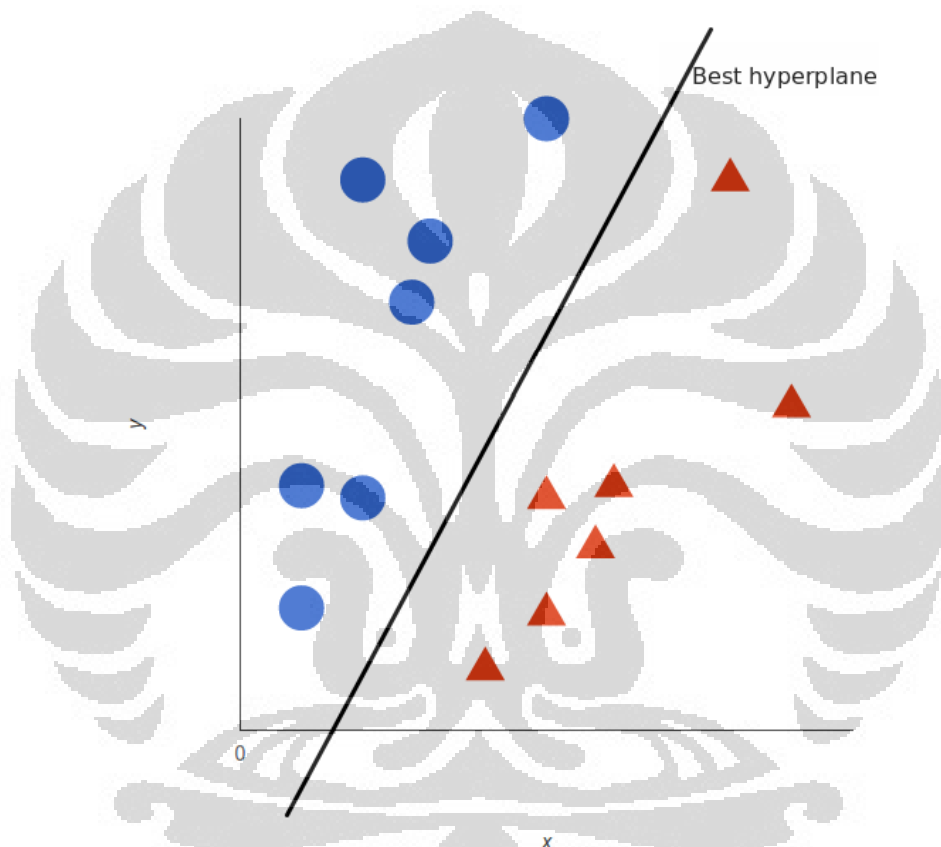
Vector yang dihasilkan dari algoritma HOG kemudian akan menjadi input untuk algoritma jaringan saraf tiruan pada tulisan ini.

2.3. *Multiscale Sliding Window*

Multiscale Sliding Window adalah metode yang digunakan dalam pencarian objek untuk mencari keseluruhan gambar dengan menggunakan sebuah kotak deteksi yang digerakkan. *Multiscale* berarti gambar akan diubah ukurannya setelah kotak deteksi telah mencari keseluruhan gambar. Perubahan ukuran ini digunakan agar pencarian dapat menemukan objek dengan berbagai ukuran.

2.4. *Support Vector Machine (SVM)*

Support Vector Machine adalah salah satu metode *machine learning* yang menggunakan *supervised learning* atau pembelajaran yang dikawal oleh pengguna [7]. SVM sering digunakan untuk menyelesaikan masalah klasifikasi dan regresi. Objektif dari algoritma SVM adalah untuk menemukan sebuah *hyperplane* pada bidang dengan n-dimensi yang dapat mengklasifikasikan perbedaan dari data masukan.



Gambar 2.2 Contoh visualisasi dari SVM

Adapun pengertian dari *hyperplane* adalah sebuah pembatas dimana masing-masing sisi dari pembatas merupakan kumpulan data dengan kelas yang berbeda. Pada bidang 2 dimensi, *hyperplane* direpresentasikan sebagai garis. Sedangkan pada bidang 3 dimensi *hyperplane* direpresentasikan sebagai bidang datar 2 dimensi.

Support vector adalah titik-titik data yang terletak dekat dengan *hyperplane* dan mempengaruhi posisi dan arah dari *hyperplane* tersebut. Maka pembelajaran

SVM adalah proses menemukan posisi dan arah dari *hyperplane* dengan menyesuaikan nilai support vectornya.

2.5. OpenCV

OpenCV merupakan sebuah framework yang berisi sekumpulan library yang dapat digunakan untuk mempermudah pengolahan citra pada komputer. Library ini mengandung lebih dari 2500 algoritma pengolahan citra yang telah dioptimisasi, termasuk algoritma *machine learning* di dalamnya [8]. Kumpulan algoritma yang dimiliki OpenCV dapat digunakan untuk menyelesaikan masalah pendeteksian wajah, pelacakan objek bergerak, dan lainnya. OpenCV dapat ditulis dengan bahasa pemrograman C++, Python, Java, dan MATLAB serta dapat digunakan dalam berbagai OS seperti Windows, Linux, Android, dan MacOS.

2.6. FANN (Fast Artificial Neural Network)

FANN merupakan sebuah *library* untuk mempermudah pembuatan jaringan saraf tiruan yang ditulis dengan bahasa C [9]. FANN memiliki kecepatan pemrosesan yang tergolong cepat, dapat mencapai 150 kali lebih cepat dibandingkan *library* lain. FANN dapat dengan mudah digunakan hanya dengan mengunduh *library*-nya dari situs resmi FANN tanpa harus melakukan instalasi apapun. FANN menggunakan metode *backpropagation* untuk pembelajaran jaringan.

Penggunaan FANN dapat dilakukan dengan mudah pada bahasa C++ dengan melakukan `#include "fann.h"` pada awal program. Untuk pembuatan jaringan saraf tiruan, FANN memerlukan beberapa parameter terlebih dahulu seperti contoh pecahan kode dibawah ini:

```
const unsigned int num_input = 2;
const unsigned int num_output = 1;
const unsigned int num_layers = 3;
const unsigned int num_neurons_hidden = 3;

struct fann *ann = fann_create_standard(num_layers, num_input,
    num_neurons_hidden, num_output);

fann_set_activation_function_hidden(ann, FANN_SIGMOID_SYMMETRIC);
fann_set_activation_function_output(ann, FANN_SIGMOID_SYMMETRIC);
```

Pada kode dapat dilihat bahwa dilakukan inisialisasi parameter berupa jumlah node input, jumlah node output, jumlah layer, dan jumlah node hidden. Kemudian jaringan dibuat dengan memanggil fungsi `fann_create_standard()` dan memasukkan parameter yang telah diinisialisasi sebelumnya. Fungsi aktivasi yang digunakan pada jaringan dapat ditentukan dengan menggunakan fungsi `fann_set_activation_function_hidden()` dan `fann_set_activation_function_output()`. Pada contoh diatas digunakan fungsi aktivasi berupa sigmoid.

Adapun untuk melakukan pembelajaran pada jaringan dapat dilihat seperti pada contoh pecahan kode dibawah ini:

```
const float desired_error = (const float) 0.001;
const unsigned int max_epochs = 500000;
const unsigned int epochs_between_reports = 1000;

fann_train_on_file(ann, "xor.data", max_epochs,
    epochs_between_reports, desired_error);

fann_save(ann, "xor_float.net");

fann_destroy(ann);
```

Parameter yang perlu disediakan untuk proses pembelajaran adalah *stopping condition* berupa nilai error yang diinginkan, nilai *epoch* maksimal, dan setiap berapa *epoch* program harus melaporkan hasil yang didapat. Setelah parameter diinisialisasi, proses pembelajaran dilakukan dengan fungsi `fann_train_on_file()`, dimana parameter input pertama adalah jaringan yang telah dibuat menggunakan fungsi `fann_create_standard()`, parameter input kedua adalah data yang akan digunakan untuk proses pembelajaran, dan parameter input selanjutnya adalah parameter yang telah diinisialisasi sebelumnya. Setelah proses pembelajaran selesai, jaringan akan disimpan dengan menggunakan fungsi `fann_save()` dengan parameter input pertama yaitu jaringan yang akan disimpan, dan parameter input kedua adalah nama dari jaringan yang akan disimpan.

Untuk melakukan proses testing pada jaringan yang telah dilakukan pembelajaran, dapat dilihat seperti pada contoh pecahan kode dibawah ini:

```
struct fann *ann = fann_create_from_file("xor_float.net");  
input[0] = -1;  
input[1] = 1;  
calc_out = fann_run(ann, input);
```

Pertama perlu dibuat jaringan saraf tiruan berdasarkan jaringan yang telah dibentuk menggunakan fungsi `fann_save()` sebelumnya. Hal ini dilakukan dengan menggunakan fungsi `fann_create_from_file()` dengan parameter input adalah nama jaringan yang telah dibuat. Kemudian proses utama dari testing adalah pada fungsi `fann_run()` dengan parameter input pertama yaitu jaringan yang akan digunakan, dan parameter input kedua adalah data yang akan digunakan sebagai input dari jaringan. Fungsi `fann_run()` kemudian akan menyimpan hasil keluaran jaringan pada variabel `calc_out`.



BAB 3

PERANCANGAN SISTEM

Pada bab ini dibahas mengenai perancangan sistem yang dibagi menjadi perancangan kebutuhan sistem, perancangan perangkat lunak, perancangan pengambilan data, dan perancangan pengujian sistem. Adapun untuk perancangan perangkat lunak akan dibagi lagi menjadi proses pembelajaran jaringan saraf tiruan, dan proses pendeteksian pada alat.

3.1. Perancangan Kebutuhan Sistem

Perancangan kebutuhan sistem merupakan tahap pertama dalam perancangan dimana didefinisikan kebutuhan-kebutuhan yang harus terpenuhi agar sistem dapat berjalan sesuai dengan tujuannya. Telah dijelaskan pada bab pendahuluan bahwa tujuan dari sistem yang akan dibuat adalah untuk mendeteksi bola yang sebagian besar berwarna putih. Telah dijelaskan juga bahwa tantangan dari masalah ini adalah banyaknya warna putih lain yang mungkin ada di sekitar, seperti pantulan cahaya dari lampu sehingga tidak dapat mengandalkan pendeteksian warna semata. Sistem juga harus mampu menentukan posisi bola pada gambar, karena informasi ini akan digunakan pada perhitungan pergerakan robot jika telah diimplementasikan kedalam robot. Berdasarkan dari hal yang telah disebutkan, maka kebutuhan yang harus terpenuhi oleh sistem adalah sebagai berikut:

1. Sistem dapat mendeteksi objek yang telah ditentukan sebelumnya yaitu berupa bola.
2. Sistem dapat membedakan objek yang harus dideteksi dengan *noise* yang ada di sekitar.
3. Sistem dapat mengetahui posisi koordinat objek yang dideteksi pada pandangan.

Untuk dapat memenuhi kebutuhan sistem poin nomor 1 dan 2 untuk mendeteksi objek dan membedakannya dengan *noise* yang ada di lingkungan, maka diperlukan suatu metode *computer vision* yang cukup *robust* agar tingkat akurasi

dari pendeteksian tinggi. Hal ini diharapkan dapat dicapai dengan melakukan implementasi dari jaringan saraf tiruan sebagai metode pembelajaran, dan *histogram of oriented gradients* sebagai metode pengambilan fitur gambar. Adapun untuk mempermudah pemrosesan gambar sehingga dapat diolah oleh algoritma *histogram of oriented gradients* akan digunakan *library* OpenCV menggunakan bahasa pemrograman C++.

Kebutuhan pada poin nomor 3 dapat dipenuhi dengan menerapkan metode *multiscale sliding window* pada sistem. Metode ini akan mencari keseluruhan gambar untuk objek yang dideteksi, jika ada objek terdeteksi maka akan memberikan hasil koordinat objek tersebut.

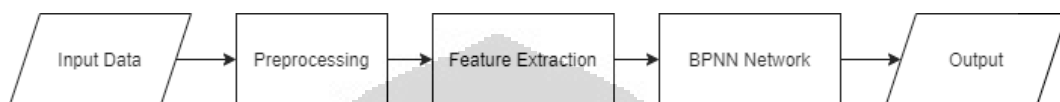
3.2. Perancangan Perangkat Lunak

Pada sub bab ini akan dibahas mengenai perancangan perangkat lunak dari sistem. Berdasarkan perancangan kebutuhan sistem, diperlukan program yang dapat melakukan pengolahan dari suatu gambar dan mendeteksi apakah ada objek bola di dalamnya. Untuk mewujudkan hal tersebut, digunakan sebuah *library* yang dapat mempermudah melakukan pemrosesan gambar yaitu OpenCV. Pembuatan sistem akan menggunakan OpenCV dengan bahasa pemrograman C++ pada OS Linux Ubuntu 16.04 LTS. Bahasa pemrograman C++ digunakan karena dapat mencapai kecepatan eksekusi yang lebih tinggi dibandingkan dua bahasa lainnya yang dapat digunakan untuk menulis OpenCV, yaitu Java dan Python. Dengan OpenCV sebagai pengatur pemrosesan gambar, akan dibuat program jaringan saraf tiruan dengan metode pembelajaran *backpropagation* dan menggunakan *histogram of oriented gradients* sebagai *feature descriptor* dari program.

Agar jaringan saraf tiruan dapat mendeteksi objek, diperlukan proses pembelajaran terlebih dahulu. Karena proses pembelajaran memiliki tahap yang berbeda dari proses pendeteksian, maka pada sub bab ini akan dibagi ke dalam dua pembahasan yaitu perancangan program pembelajaran *backpropagation*, dan perancangan program pendeteksian dan pelacakan objek.

3.2.1. Perancangan Pembelajaran Jaringan Saraf Tiruan

Proses awal dari pembelajaran adalah mempersiapkan data yang akan dipelajari. Data akan dipisah menjadi dua, yaitu data positif dan data negatif. Data positif akan berisi gambar objek yang benar sesuai sasaran. Sedangkan data negatif adalah gambar noise atau objek yang tidak relevan dengan sasaran. Perbandingan banyak data untuk proses pembelajaran dan testing adalah 70:30.



Gambar 3.1 Diagram blok proses pembelajaran BPNN

Adapun proses dari pembelajaran digambarkan pada Gambar 3.1 **Error! Reference source not found.** Pertama data gambar yang sudah disiapkan akan dimasukkan ke dalam program. Data tersebut kemudian akan diolah terlebih dahulu, yaitu dengan memastikan ukuran gambar adalah 64x64 dengan *colorspace grayscale*. Data yang sudah dilakukan pengolahan awal kemudian akan diambil fitur-fitur pentingnya. Metode pengambilan fitur yang akan digunakan dalam pembuatan program ini adalah *histogram of oriented gradients*. Metode ini dipilih karena pendeteksian tidak terlalu terpengaruh dengan perubahan geometri maupun perubahan kondisi cahaya pada gambar. Fitur-fitur penting yang telah didapatkan kemudian akan dilakukan pembelajaran pada jaringan BPNN sampai didapatkan model BPNN yang dapat mendeteksi objek dengan akurasi yang cukup tinggi. Keluaran dari proses pembelajaran adalah model BPNN dengan nilai bobot yang telah disesuaikan dengan kebutuhan data.

Pada bagian *feature extraction*, fitur yang diambil dari gambar adalah berupa distribusi atau histogram dari arah gradient atau perubahan nilai pixel. Pada pendeteksian bentuk objek, informasi mengenai bentuk dan sudut tepi dari suatu objek relatif lebih penting dibandingkan pada bagian yang datar. Adapun nilai gradient pada tepi objek relatif tinggi sehingga cocok digunakan sebagai informasi fitur yang akan diambil.

Algoritma 1. Histogram of Oriented Gradients

Include library opencv

1. Inisialisasi library

```

2. Inisialisasi parameter yang diperlukan
3. While input masuk
4.   If ukuran input tidak sesuai
5.     Error
6.   Endif
7.   VectorOutput += Hog.compute(parameter hog)
8. Endwhile
9. Save vector

```

Algoritma 3.1 Pseudocode HOG

Perhitungan HOG dapat mudah dilakukan dengan menggunakan OpenCV. Terdapat fungsi di dalam OpenCV untuk melakukan perhitungan HOG dengan parameter yang dapat diubah-ubah. Parameter utama yang diperlukan adalah ukuran input yaitu 64x64. Kemudian fungsi `compute()` [10] pada OpenCV akan melakukan perhitungan HOG dan memberikan keluaran berupa vector. Vector yang didapat kemudian akan disimpan dan digunakan untuk proses pembelajaran BPNN.

Vector yang didapatkan dari proses perhitungan HOG kemudian akan menjadi input untuk jaringan BPNN. Model BPNN yang akan dibentuk adalah *node* pada *input layer* berjumlah sama dengan vector keluaran HOG, delapan *node* pada *hidden layer*, dan satu *node* pada *output layer*.

Algoritma 2. BPNN Training

```

1. Inisialisasi parameter training
2. Inisialisasi node pada tiap layer
3. Inisialisasi bobot
4.
5. //Training
6. While !stopping condition
7.   Forward input ke hidden layer, kalikan dengan bobot
8.   Hitung dengan fungsi aktivasi pada hidden layer
9.   Forward hidden ke output layer, kalikan dengan bobot
10.  Hitung dengan fungsi aktivasi pada output layer
11.  Hitung besar error
12.  Backward output ke hidden, hitung koreksi bobot
13.  Backward hidden ke input, hitung koreksi bobot
14.  Update nilai bobot
15. Endwhile
16. Simpan nilai bobot
17.
18. //Testing

```

```

19. While input data testing
20.     Forward input ke hidden layer, kalikan dengan bobot
21.     Hitung dengan fungsi aktivasi pada hidden layer
22.     Forward hidden ke output layer, kalikan dengan bobot
23.     Hitung dengan fungsi aktivasi pada output layer
24.     Cocokkan hasil output dengan target
25.     If cocok
26.         Total_berhasil++
27.     endif
28. Endwhile
29. RecognitionRate = Total_berhasil / JumlahDataTesting
30. Simpan model BPNN

```

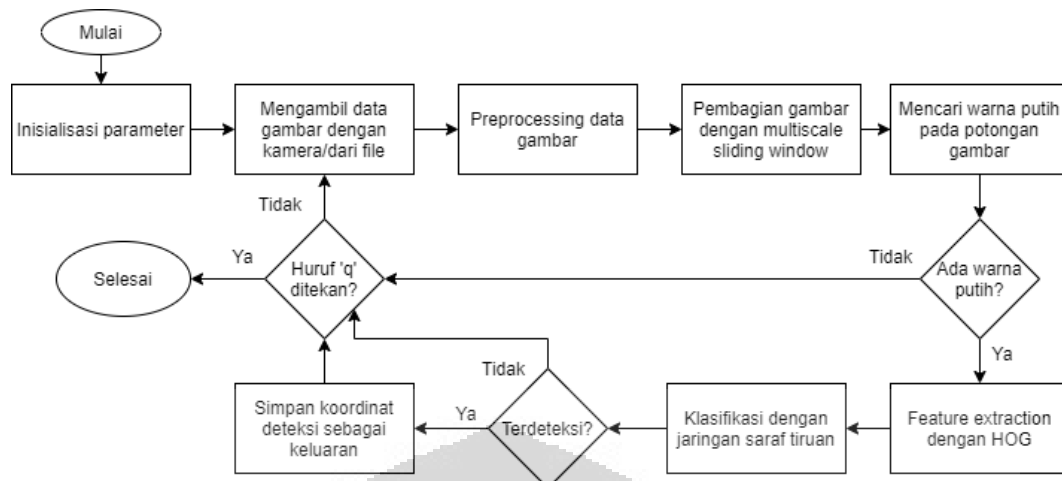
Algoritma 3.2 Pseudocode Training BPNN

Proses pembelajaran BPNN akan dibuat menggunakan *library* FANN. Inisialisasi parameter training berupa nilai alpha, miu, dan stopping condition. Kemudian vektor dari HOG akan dimasukkan ke dalam model BPNN. Inisialisasi bobot menggunakan metode random. Dilakukan *looping* untuk proses perhitungan utama dari BPNN. Proses *feed forward* sampai menghasilkan output, lalu dihitung errornya. Kemudian error digunakan untuk mendapatkan koreksi bobot dari masing-masing layer. Nilai bobot akan diupdate berdasarkan besar koreksi bobot yang didapatkan. *Looping* dilakukan sampai *stopping condition* tercapai.

Jika *stopping condition* tercapai, maka nilai bobot akan disimpan. Kemudian akan dilakukan testing dengan melakukan proses *feed forward* sampai ditemukan output. Output yang didapat akan dibandingkan dengan target, jika cocok maka akan dianggap berhasil. Nilai *recognition rate* akan dihitung dan ditampilkan untuk memberi tahu seberapa banyak pendeteksian yang sukses. Model BPNN yang telah terlatih kemudian akan disimpan untuk digunakan pada proses pendeteksian.

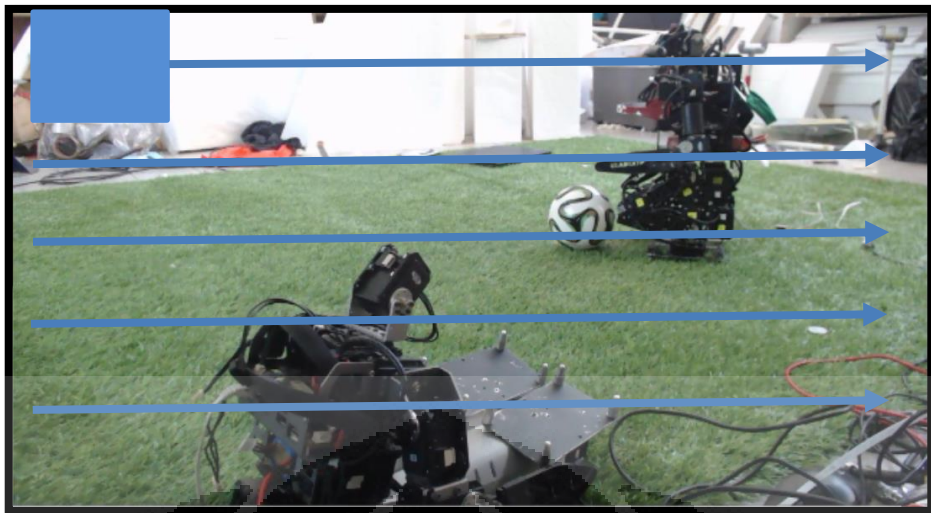
3.2.2. Perancangan Proses Pendeteksian dan Pelacakan

Setelah proses pembelajaran data telah selesai dilakukan pada model BPNN, maka selanjutnya model tersebut akan diterapkan pada sistem untuk melakukan pendeteksian dan pelacakan objek. Secara garis besar, alur dari proses pendeteksian yang akan dibuat adalah seperti pada Gambar 3.2.



Gambar 3.2 Gambar Alur Proses Pendeteksian

Proses pendeteksian dimulai dari inisialisasi parameter dan *library* yang dibutuhkan oleh program. Kemudian data gambar akan diambil dari kamera atau dari file untuk dapat diolah. Data yang diperoleh kemudian dilakukan pemrosesan awal dengan merubah gambar input menjadi *grayscale*. Selain merubah gambar menjadi *grayscale*, diterapkan juga *thresholding* warna pada gambar untuk mencari warna putih, namun hasil dari *thresholding* disimpan pada variabel yang berbeda agar tidak mengganggu proses *feature extraction*. Setelah *preprocessing* selesai, diterapkan metode *multiscale sliding window* untuk memotong gambar menjadi bagian-bagian yang lebih kecil. Metode ini akan membentuk kotak deteksi yang digerakkan, kotak deteksi yang digunakan berukuran 64x64 sesuai dengan parameter yang digunakan oleh proses *feature extraction* dengan HOG. Adapun kotak deteksi akan bergeser bertahap ke kanan sejauh 32 pixel sampai keseluruhan gambar dideteksi. Proses ini diilustrasikan secara sederhana dengan Gambar 3.3. Setelah pergeseran kotak seperti pada Gambar 3.4 selesai, kemudian gambar akan diubah ukurannya menjadi lebih kecil. Hal ini menjadikan ukuran kotak deteksi seolah lebih besar dan bisa melakukan pendeteksian jika terdapat ukuran objek yang besar. Perubahan ukuran menggunakan skala dari 1 sampai 0 dengan perubahan skala yang dapat ditetapkan oleh pengguna, pada penelitian ini digunakan perubahan skala perubahan ukuran gambar sebesar 0.2 yang berarti terdapat 5 kali perubahan ukuran gambar untuk setiap gambar yang didapat.



Gambar 3.3 Ilustrasi Pergerakan Kotak Deteksi

Kemudian untuk setiap langkah dari pergerakan kotak deteksi akan dicari warna putih terlebih dahulu. Pencarian warna putih dilakukan pada variabel hasil *thresholding* warna putih yang dilakukan pada tahap sebelumnya. Pencarian warna putih dilakukan untuk mempercepat proses pendeteksian serta mengurangi kemungkinan terjadinya *noise*. Jika terdapat warna putih pada kotak deteksi, akan dilakukan feature extraction HOG seperti pada Algoritma 3.1 untuk menentukan poin poin yang penting untuk diolah pada model jaringan saraf tiruan. Data yang masuk pada model jaringan saraf akan diterapkan pengklasifikasian dan ditentukan apakah terdapat objek yang ingin dideteksi atau tidak. Algoritma klasifikasi ini sama dengan bagian testing pada Algoritma 3.2, yaitu mencocokkan hasil output *feed forward* dengan target. Jika saat proses klasifikasi ditemukan objek sasaran, maka koordinat tempat objek tersebut ditemukan serta ukurannya akan disimpan. Kemudian akan diterapkan algoritma pelacakan untuk koordinat yang didapatkan sehingga program dapat tetap mengikuti gerak bola.

Algoritma 3. Pelacakan Objek	
1.	If objek terdeteksi > 0
2.	For each objek terdeteksi
3.	If first detection
4.	koordinatOut += koordinatDeteksi
5.	ukuranBolaOut += ukuranBolaDeteksi
6.	banyakDeteksi++
7.	else
8.	if abs(koordinatSimpan - koordinatObjek) < tooFar

```

9.         koordinatOut += koordinatDeteksi
10.        ukuranBolaOut += ukuranBolaDeteksi
11.        banyakDeteksi++
12.    End if
13. End for
14.
15.    koordinatOut /= banyakdeteksi
16.    ukuranBolaOut /= banyakdeteksi
17.
18. Else koordinatOut = -1; ukuranBolaOut = -1
19. End if
20.
21. If koordinatOut != -1
22.     For i = stackMaxSize; i >= 0; i--
23.         stackKoordinat[i] = stackKoordinat[i-1]
24.         stackUkuran[i] = stackUkuran[i-1]
25.     End for
26.
27.     If stackSize < stackMaxSize
28.         stackSize++
29.     End if
30.
31.     stackKoordinat[0] = koordinatOut
32.     stackUkuran[0] = ukuranBolaOut
33.
34. else notFound++
35. end if
36.
37. If stackSize > 0
38.     For i = 0; i < stackSize; i++
39.         koordinatOut += stackKoordinat[i]
40.         ukuranBolaOut += stackUkuran[i]
41.     end for
42.
43.     koordinatOut /= stackSize
44.     ukuranBolaOut /= stackSize
45.
46. End if
47.
48. koordinatSimpan = koordinatOut
49.
50. If notFound > limit
51.     Clear stackKoordinat
52.     Clear stackUkuran
53.     stackSize = 0
54.     notFound = 0
55. End if
56.
57. OpenCV make circle from koordinatOut and ukuranBolaOut

```

Algoritma 3.3 Algoritma Pelacakan Objek

Proses pelacakan objek secara lengkap dapat dilihat pada Algoritma 3.3. Pertama program memeriksa apakah terdapat objek yang terdeteksi oleh jaringan saraf. Karena terdapat kemungkinan bahwa ada lebih dari satu objek yang terdeteksi, maka program melakukan rata-rata terhadap koordinat dan ukuran bola yang didapatkan dari masing-masing objek yang terdeteksi pada baris 2 sampai dengan baris 16 dengan keluaran yaitu koordinatOut dan ukuranBolaOut. Sedangkan jika tidak ada objek yang terdeteksi, maka nilai koordinatOut dan ukuranBolaOut akan menjadi -1. Kemudian hasil rata-rata koordinat akan dimasukkan ke dalam stack seperti pada baris 22 sampai 32. Untuk setiap koordinat baru yang dimasukkan pada stack, maka koordinat yang paling lama berada dalam stack akan dihilangkan. Stack digunakan pada program agar nilai koordinat yang didapatkan menjadi stabil dan tidak terjadi perubahan koordinat yang mendadak. Selain itu juga berfungsi agar program dapat mengingat posisi objek meskipun pendeteksian tidak berhasil untuk beberapa saat. Koordinat dan ukuran bola yang digunakan sebagai hasil akhir adalah nilai rata-rata dari seluruh nilai yang ada di dalam stack seperti pada baris 38 sampai 44. Koordinat hasil akhir ini akan disimpan pada suatu variabel koordinatSimpan untuk digunakan pada baris ke-8. Jika koordinat deteksi yang baru didapatkan terletak terlalu jauh dari koordinatSimpan, maka koordinat tersebut akan diabaikan. Hal ini dilakukan untuk mengurangi pengaruh dari *noise* terhadap hasil koordinat. Kemudian jika program tidak dapat mendeteksi objek di dekat titik deteksi sebelumnya untuk beberapa saat, program akan mengosongkan stack seperti pada baris 50 sampai 55 sehingga program dapat melakukan pendeteksian keseluruhan seperti pada awal program.

3.3. Perancangan Pengumpulan Data

Sebelum dapat melakukan pembelajaran jaringan saraf tiruan dengan program yang telah dirancang, dibutuhkan data yang akan menjadi input dari program tersebut. Karena tujuan dari program adalah untuk mendeteksi bola, data yang digunakan adalah berbagai gambar bola dengan tampak yang berbeda. Gambar bola yang dikumpulkan juga harus berukuran 64x64 pixel sesuai dengan ukuran yang dibutuhkan oleh HOG. Masing-masing data yang didapatkan akan diberikan label sesuai dengan ciri yang dimiliki.

Data yang memuat gambar bola akan diberikan label positif yang menandakan bahwa ada bola pada gambar. Jika gambar dengan label ini digunakan sebagai masukan pada jaringan saraf, maka setelah diolah oleh jaringan diharapkan mendapatkan nilai 1 pada *node* keluaran. Hal ini menandakan bahwa jaringan saraf menganggap gambar yang menjadi masukan adalah benar berupa gambar bola.

Selain gambar bola, diperlukan juga kumpulan gambar selain bola agar jaringan dapat lebih mudah membedakan lingkungan sekitar dengan bola. Data yang tidak memuat gambar bola akan diberikan label negatif. Jika gambar dengan label ini digunakan sebagai masukan pada jaringan saraf, maka setelah diolah oleh jaringan diharapkan mendapatkan nilai 0 pada *node* keluaran. Hal ini menandakan bahwa jaringan saraf menganggap gambar yang menjadi masukan adalah bukan berupa bola.

Pada tulisan ini, akan digunakan kumpulan data bola yang telah dipublikasikan oleh salah satu tim robotik peserta RoboCup yaitu SPQR dari *University of Rome "Sapienza"* pada tahun 2016. Data tersebut dapat diunduh dari situs LabRoCoCo [11] yang memuat berbagai gambar bola yang dapat menjadi label positif dan juga gambar selain bola yang dapat menjadi label negatif. Secara keseluruhan terdapat 4660 gambar yang terdapat bola di dalamnya, dan 2141 gambar selain bola. Dari keseluruhan gambar yang ada, akan digunakan 70% untuk proses pembelajaran, dan 30% untuk proses pengujian.

Untuk menambah akurasi pendeteksian, akan ditambahkan gambar dari hasil pengujian pendeteksian dengan data yang telah diunduh. Setelah jaringan saraf berhasil dibentuk dari proses pembelajaran data tersebut, akan dilakukan pengujian dengan cara memasukan jaringan saraf pada program pendeteksian. Input dari program pendeteksian berupa kamera yang mengambil gambar secara *realtime*. Kemudian akan dilihat apakah terdapat *false detection* atau kekeliruan dalam mendeteksi input yang diberikan. Jika program mendeteksi ada bola pada bagian yang sebenarnya tidak ada bola, hal ini dinamakan *false positive*. Jika program mendeteksi tidak ada bola pada bagian yang sebenarnya ada bola, hal ini dinamakan *false negative*. Jika terdapat kedua hal tersebut, maka gambar yang memuat pendeteksian yang keliru akan disimpan dan digunakan kembali sebagai data input

pembelajaran, dimana bagian yang memuat *false positive* akan diberikan label negatif, dan bagian yang memuat *false negative* akan diberikan label positif.

3.4. Perancangan Pengujian

Pada subbab ini akan dibahas macam-macam pengujian yang dilakukan untuk mengetahui performa dari sistem yang telah dirancang. Pengujian akan dipisah untuk program pembelajaran dan program pendeteksian.

3.4.1. Perancangan Pengujian Program Pembelajaran

Untuk program pembelajaran, hal penting yang perlu diketahui mengenai performa program adalah kecepatan pembelajaran dan akurasi hasil pembelajaran. Sedangkan untuk program pendeteksian, hal penting yang perlu diketahui mengenai performa program adalah kecepatan pendeteksian dan akurasi pendeteksian.

Performa program pembelajaran mengenai kecepatan pembelajaran dan akurasi hasil pembelajaran bergantung pada parameter yang digunakan saat melakukan proses pembelajaran [12]. Parameter yang akan divariasikan untuk proses pembelajaran antara lain adalah:

1. Jumlah *node* pada *hidden layer*.
2. Laju pembelajaran.
3. Momentum pembelajaran.

Jumlah *node* yang digunakan pada *hidden layer* berpengaruh besar terhadap akurasi dan kecepatan pembelajaran. Semakin banyak *node* yang ada pada *hidden layer* maka akan semakin lambat proses pembelajaran terjadi. Tetapi jika jumlah *node* terlalu sedikit, maka jaringan saraf tidak dapat mengklasifikasikan data masukan yang diberikan dengan baik.

Pengaruh dari laju pembelajaran terhadap kecepatan dan akurasi pembelajaran adalah, jika laju pembelajaran terlalu kecil, maka proses pembelajaran akan berlangsung lama untuk mencapai akurasi yang diinginkan, dan memungkinkan terjadinya peristiwa *vanishing gradient* dimana nilai *delta* saat melakukan perubahan bobot jaringan saraf menjadi sangat kecil dan perubahan

bobot tidak terjadi setelahnya pada jaringan saraf [13]. Namun jika laju pembelajaran terlalu besar, jaringan saraf akan menjadi tidak stabil, dimana terdapat kemungkinan terjadinya peristiwa *exploding gradient* dimana nilai *delta* saat melakukan perubahan bobot menjadi terlalu besar hingga mencapai nilai maksimum yang dapat disimpan variabel bobot [14].

Nilai momentum mempengaruhi kecepatan dan akurasi pembelajaran, dimana semakin tinggi momentum yang digunakan, maka semakin besar nilai *delta* yang didapatkan dalam perubahan bobot jaringan. Nilai *delta* yang semakin besar ini menandakan jaringan saraf akan mencapai nilai yang diinginkan semakin cepat. Namun, jika nilai momentum terlalu tinggi, maka terdapat kemungkinan untuk jaringan saraf tidak mengenai nilai yang diinginkan.

Untuk seluruh parameter yang divariasikan, jumlah *epoch* yang dilakukan adalah sebanyak 200 kali dengan target error sebesar 0,001. Dengan input berupa berbagai gambar yang telah dikumpulkan pada perancangan pengambilan data. Dari keseluruhan data gambar, 70% akan menjadi input pada proses pembelajaran, dan 30% akan menjadi input pada proses testing. Adapun variasi yang dilakukan untuk masing-masing parameter adalah sebagai berikut:

1. Variasi jumlah *node* pada *hidden layer*: 4, 8, 12, 16, 24, 32. Parameter tetap: laju pembelajaran bernilai 0,5, momentum bernilai 0,5, fungsi aktivasi *sigmoid*.
2. Variasi laju pembelajaran: 0,1 – 1 dengan pertambahan 0,1 setiap perubahan variasi. Parameter tetap: *node hidden layer* berjumlah 8, momentum bernilai 0,5, fungsi aktivasi *sigmoid*.
3. Variasi momentum: 0,1 – 1 dengan pertambahan 0,1 setiap perubahan variasi. Parameter tetap: *node hidden layer* berjumlah 8, laju pembelajaran bernilai 0,5, fungsi aktivasi *sigmoid*.

3.4.2. Perancangan Pengujian Program Pendeteksian dan Pelacakan

Untuk program pendeteksian dan pelacakan, hal penting yang perlu diketahui mengenai performa jaringan saraf adalah kecepatan dan akurasi dari proses pendeteksian dan pelacakan. Pengujian akan dipisah menjadi pengujian

proses pendeteksian dan pengujian proses pelacakan. Jaringan saraf yang akan digunakan pada pengujian adalah jaringan dengan jumlah *node* pada *hidden layer* sebanyak 8 buah.

Pengujian proses pendeteksian akan dilakukan dengan data berupa gambar ruangan dengan objek bola di dalamnya. Kemudian juga akan dibandingkan dengan metode pendeteksian menggunakan model selain jaringan saraf tiruan yaitu metode pembelajaran SVM yang telah disediakan oleh OpenCV [15]. Hal ini dilakukan untuk mengetahui apakah sistem yang dibuat dapat bersaing dengan pendeteksian menggunakan metode yang sudah ada sebelumnya. Parameter yang menjadi tolak ukur performa adalah *accuracy*, *precision*, *recall*, dan *frame per second* (FPS).

Adapun pengertian dari masing-masing parameter yang akan diukur adalah sebagai berikut:

- *Accuracy* merupakan tingkat keberhasilan program untuk mendeteksi label yang diinginkan [16].
- *Precision* merupakan perbandingan banyaknya deteksi yang dianggap benar adalah label yang benar dengan keseluruhan deteksi yang dianggap benar [17].
- *Recall* merupakan perbandingan banyaknya deteksi benar dengan banyaknya objek dengan label yang benar [17].
- *Frame per second* merupakan banyaknya gambar yang dapat diproses oleh program dalam waktu satu detik.

Nilai *precision* dan *recall* dapat digunakan untuk mendapatkan nilai *F-score* yang merupakan ringkasan dari *precision* dan *recall* [18] dengan formula yaitu:

$$F = 2 \times (\textit{precision} \times \textit{recall} / (\textit{precision} + \textit{recall}))$$

F-score akan digunakan untuk mempermudah perbandingan nilai *precision* dan *recall* antara program yang dibuat dengan pembandingan SVM. Metode yang memiliki nilai *F-score* yang lebih tinggi dapat dikatakan memiliki performa yang lebih baik.

Untuk dapat mendapatkan parameter yang menjadi tolak ukur performa, diperlukan perhitungan terhadap data yang didapat. Perhitungan dilakukan dengan

menggunakan nilai True Positive (TP) atau deteksi yang benar merupakan label yang ingin dideteksi, True Negative (TN) atau deteksi yang benar merupakan objek selain objek dengan label yang ingin dideteksi, False Positive (FP) atau deteksi yang salah dimana benda lain dianggap bola, dan False Negative (FN) atau deteksi yang salah dimana bola dianggap benda lain. Adapun cara menghitung masing-masing parameter adalah sebagai berikut:

- $Accuracy = (TP + TN) / (TP + TN + FP + FN)$
- $Precision = TP / (TP + FP)$
- $Recall = TP / (TP + FN)$
- $FPS = \text{banyaknya data} / \text{waktu rata-rata}$

Sedangkan pengujian proses pelacakan akan dilakukan dengan data berupa input video dari kamera. Akan diamati apakah program dapat mengikuti bola dengan baik. Selain itu akan dihitung pula rata-rata FPS dari proses pelacakan tersebut.

BAB 4

PENGUJIAN DAN ANALISA

Pada bab ini akan dibahas mengenai pengujian dan analisa dari sistem yang sudah dibuat berdasarkan perancangan pada bab sebelumnya. Pada bagian perancangan pengujian, telah dijelaskan bagaimana sistem akan diuji. Secara garis besar, pengujian akan dibagi menjadi dua yaitu pengujian program pembelajaran jaringan saraf tiruan, dan pengujian program pendeteksian objek bola.

4.1. Pengujian Program Pembelajaran

Sesuai dengan perancangan pengujian yang telah dibuat, parameter yang akan divariasikan adalah jumlah *node* pada *hidden layer*, laju pembelajaran, dan momentum. Tolak ukur performa program berdasarkan tingkat akurasi, kecepatan pembelajaran, serta kecepatan saat melakukan testing. Adapun parameter tetap untuk semua variasi adalah jumlah *epoch* sebanyak 200 kali, dan target *error* sebesar 0,001.

4.1.1. Pengujian Variasi Jumlah *Node* Pada *Hidden Layer*

Setelah dilakukan pengujian hasil yang didapatkan adalah:

Tabel 4.1 Hasil Pengujian Variasi Jumlah *Node*

Jumlah Node	Min Error Training	Durasi Pembelajaran (detik)	MSE Testing	Durasi Testing (detik)
4	0.00587	67.94	0.0302	0.07
8	0.00330	125.08	0.0270	0.11
12	0.00284	178.37	0.0283	0.16
16	0.00311	234.37	0.0245	0.21
24	0.00306	358.38	0.0281	0.32
32	0.01047	456.07	0.0325	0.40

Dari pengujian yang hasilnya tertulis pada Tabel 4.1, dapat diperhatikan bahwa semakin banyak *node* yang ada pada *hidden layer*, semakin lambat selesainya proses pembelajaran dan testing. Hal ini terjadi karena semakin banyak *node* yang ada, semakin banyak bobot yang harus dilalui oleh data masukan sampai

menghasilkan keluaran. Dapat diperhatikan juga dengan jumlah *node* yang terlalu sedikit, nilai *error* akan menjadi besar. Hal ini dikarenakan jaringan saraf tidak memiliki bobot yang cukup untuk dapat merubah data input menjadi keluaran yang diinginkan dengan baik. Namun dari hasil pengujian terlihat bahwa jumlah *node* yang lebih besar tidak selalu menghasilkan nilai *error* yang lebih kecil, dimana hasil *error* terkecil didapatkan oleh jaringan dengan jumlah *node* 12. Hal ini dikarenakan terbatasnya epoch yang dilakukan, sehingga jaringan dengan jumlah *node* yang besar tidak mendapatkan iterasi yang cukup untuk memperkecil nilai *error*nya.

4.1.2. Pengujian Variasi Laju Pembelajaran

Setelah dilakukan pengujian hasil yang didapatkan adalah:

Tabel 4.2 Hasil Pengujian Variasi Laju Pembelajaran

Laju Pembelajaran	Min Error Training	Durasi Pembelajaran
0,1	0,00363	122,32 detik
0,2	0,00549	122,40 detik
0,3	0,00335	122,86 detik
0,4	0,00331	122,52 detik
0,5	0,00308	123,92 detik
0,6	0,00389	123,08 detik
0,7	0,00413	123,24 detik
0,8	0,00312	123,09 detik
0,9	0,00354	122,37 detik
1	0,00820	122,55 detik

Dari pengujian variasi laju pembelajaran yang hasilnya tertulis pada Tabel 4.2, dapat diperhatikan bahwa nilai *error* hanya memiliki selisih yang kecil dengan pengecualian pada laju pembelajaran 0,2 dan 1. Untuk laju pembelajaran 1, dapat diperhatikan nilai *error*nya merupakan yang paling besar dibandingkan nilai yang lain. Hal ini berarti laju pembelajaran yang digunakan sudah terlalu besar dan dapat menyebabkan *exploding gradients* jika jumlah iterasi epoch ditambahkan. Sedangkan durasi pembelajaran tidak memiliki perubahan yang signifikan dari variasi yang terjadi. Hal ini disebabkan oleh jumlah iterasi pembelajaran terbatas pada epoch yang ditentukan, jika epoch maksimum tidak ditentukan dan iterasi

dapat terjadi sampai jaringan menemui nilai minimum yang ditentukan, maka durasi pembelajaran akan bervariasi lebih signifikan.

4.1.3. Pengujian Variasi Momentum Pembelajaran

Setelah dilakukan pengujian hasilnya adalah:

Tabel 4.3 Hasil Pengujian Variasi Momentum

Momentum	Min Error Training	Durasi Pembelajaran
0,1	0,00388	123.97 detik
0,2	0,00219	122,78 detik
0,3	0,00589	122,48 detik
0,4	0,00410	123,20 detik
0,5	0,00760	121,82 detik
0,6	0,00383	122,23 detik
0,7	0,00582	121,62 detik
0,8	0,00437	122,60 detik
0,9	0,00292	122,10 detik
1	0,00268	121,91 detik

Hasil dari pengujian variasi momentum yang tertulis pada Tabel 4.3, memiliki ciri yang sama dengan pengujian variasi laju pembelajaran pada bagian durasi pembelajaran. Hal ini disebabkan kedua pengujian terikat pada hal yang sama yaitu jumlah *epoch* yang dilakukan dan jumlah *node* yang digunakan pada *hidden layer*. Sedangkan untuk nilai error yang didapatkan dari masing-masing momentum, dapat dilihat bahwa terjadi perbedaan yang cukup signifikan. Tidak terjadi pola spesifik pada variasi data yang dihasilkan. Hal ini diduga karena digunakan nilai bobot awal yang *random* pada saat pengujian.

4.2. Pengujian Program Pendeteksian Bola

Sesuai dengan perancangan pengujian yang telah dibuat, akan dilakukan pengujian program pendeteksian bola dengan data input berupa gambar yang memuat bola. Gambar yang digunakan berukuran 480x320 pixel. Karena metode *sliding window* yang digunakan memiliki kotak deteksi 64x64 pixel yang bergeser

32 pixel tiap kalinya dengan perubahan ukuran sebanyak 5 kali, maka dari satu gambar pengujian akan diperoleh data sebanyak 548 data. Jumlah gambar yang akan diproses oleh program pendeteksian adalah 30 gambar yang berlabel positif atau memuat bola. Parameter yang akan diukur pada pengujian adalah parameter yang telah disebutkan pada bagian perancangan pengujian sebelumnya. Parameter yang sama juga akan diukur untuk pendeteksian dengan metode pembelajaran SVM yang telah disediakan oleh OpenCV.

Adapun nilai yang akan didapatkan dari hasil pengujian adalah nilai *True Positive* (TP), nilai *False Positive* (FP), hasil perhitungan *accuracy*, hasil perhitungan *precision*, hasil perhitungan *recall*, dan waktu pemrosesan. Untuk *False Negative* (FN) akan mendapatkan nilai 1 jika nilai TP adalah 0 karena hanya ada satu bola pada masing-masing gambar. Sedangkan untuk nilai *True Negative* (TN) diperoleh dari total data dikurangi nilai TP, FP, dan FN karena selain deteksi yang didapat akan dianggap bahwa data adalah benar berlabel negatif atau tidak memuat bola.

Setelah dilakukan pengujian, hasil dari pendeteksian menggunakan jaringan saraf tiruan adalah:

Tabel 4.4 Hasil Pengujian Pendeteksian dengan JST

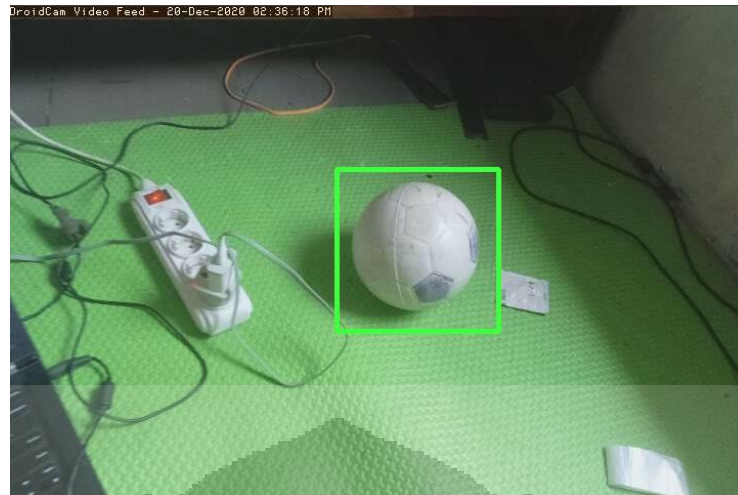
No.	True Positive	False Positive	Accuracy	Precision	Recall	Lama Proses
1	1	0	1	1	1	0,08 detik
2	1	1	0.998175	0.5	1	0,12 detik
3	1	0	1	1	1	0,13 detik
4	1	1	0.998175	0.5	1	0,10 detik
5	1	0	1	1	1	0,11 detik
6	1	5	0.990876	0.166667	1	0,11 detik
7	0	0	0.998175	1	0	0,11 detik
8	1	0	1	1	1	0,15 detik
9	0	0	0.998175	1	0	0,14 detik
10	1	0	1	1	1	0,09 detik
11	0	0	0.998175	1	0	0,12 detik
12	1	0	1	1	1	0,13 detik
13	1	0	1	1	1	0,08 detik
14	1	0	1	1	1	0,11 detik
15	1	0	1	1	1	0,15 detik

16	0	0	0.998175	1	0	0,14 detik
17	0	0	0.998175	1	0	0,10 detik
18	1	0	1	1	1	0,11 detik
19	1	0	1	1	1	0,13 detik
20	1	0	1	1	1	0,9 detik
21	0	1	0.99635	0	0	0,9 detik
22	1	0	1	1	1	0,12 detik
23	0	0	0.998175	1	0	0,10 detik
24	1	1	0.998175	0.5	1	0,09 detik
25	1	1	0.998175	0.5	1	0,11 detik
26	0	1	0.99635	0	0	0,08 detik
27	0	1	0.99635	0	0	0,11 detik
28	1	0	1	1	1	0,09 detik
29	1	0	1	1	1	0,08 detik
30	1	0	1	1	1	0,13 detik

Dari hasil yang didapatkan pada Tabel 4.4, maka didapatkan rata-rata parameter sebagai berikut:

- *Accuracy* = 0.9987
- *Precision* = 0.8055
- *Recall* = 0.7
- FPS = rata-rata lama proses / jumlah data = 6 FPS

Hasil *accuracy* yang didapatkan sebesar 0.9987 karena sebagian besar dari data berlabel negatif dengan label positif atau bola hanya 1 sehingga nilai TN yang besar menjadikan hasil *accuracy* menjadi mendekati 1. Karena adanya ketidakseimbangan jumlah data antara label negatif dan positif, maka nilai *precision* dan *recall* menjadi parameter yang lebih krusial untuk menentukan performa dari program. Dengan menerapkan formula *F-score* yang telah disebutkan pada perancangan pengujian terhadap nilai *precision* dan *recall* yang diperoleh, didapatkan hasil *F-score* sebesar 0.749.



Gambar 4.1 Contoh hasil pengujian pendeteksian dengan JST

Sedangkan hasil pengujian dengan menggunakan SVM adalah:

Tabel 4.5 Hasil Pengujian Pendeteksian dengan SVM

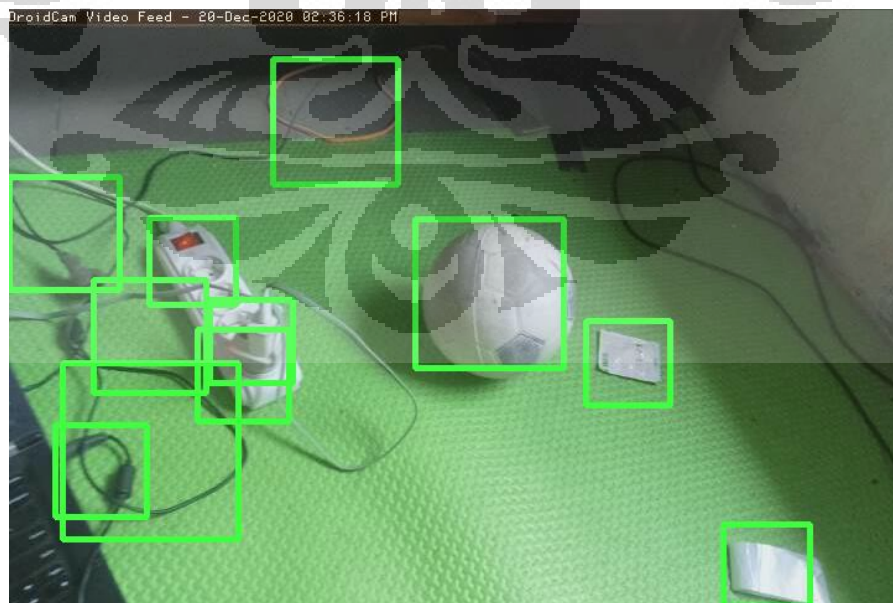
No.	True Positive	False Positive	Accuracy	Precision	Recall	Lama Proses
1	1	5	0.990875912	0.166666667	1	0,13 detik
2	1	7	0.987226277	0.125	1	0,12 detik
3	1	12	0.97810219	0.076923077	1	0,13 detik
4	0	7	0.985428051	0	0	0,12 detik
5	1	6	0.989051095	0.142857143	1	0,14 detik
6	1	6	0.989051095	0.142857143	1	0,12 detik
7	1	4	0.99270073	0.2	1	0,16 detik
8	1	7	0.987226277	0.125	1	0,12 detik
9	1	9	0.983576642	0.1	1	0,12 detik
10	1	7	0.987226277	0.125	1	0,12 detik
11	1	8	0.98540146	0.111111111	1	0,17 detik
12	1	6	0.989051095	0.142857143	1	0,12 detik
13	1	10	0.981751825	0.090909091	1	0,12 detik
14	1	7	0.987226277	0.125	1	0,13 detik
15	1	2	0.996350365	0.333333333	1	0,14 detik
16	1	2	0.996350365	0.333333333	1	0,13 detik
17	1	3	0.994525547	0.25	1	0,12 detik
18	1	6	0.989051095	0.142857143	1	0,15 detik
19	1	6	0.989051095	0.142857143	1	0,13 detik
20	1	1	0.998175182	0.5	1	0,12 detik
21	1	9	0.983576642	0.1	1	0,13 detik
22	1	2	0.996350365	0.333333333	1	0,12 detik
23	1	10	0.981751825	0.090909091	1	0,14 detik

24	1	10	0.981751825	0.090909091	1	0,12 detik
25	1	3	0.994525547	0.25	1	0,13 detik
26	1	13	0.976277372	0.071428571	1	0,12 detik
27	1	10	0.981751825	0.090909091	1	0,13 detik
28	1	3	0.994525547	0.25	1	0,15 detik
29	1	7	0.987226277	0.125	1	0,12 detik
30	1	3	0.994525547	0.25	1	0,12 detik

Dari hasil yang didapatkan pada Tabel 4.5, maka didapatkan rata-rata parameter sebagai berikut:

- $Accuracy = 0.988$
- $Precision = 0.167$
- $Recall = 0.966$
- $FPS = \text{rata-rata lama proses} / \text{jumlah data} = 7,7 \text{ FPS}$

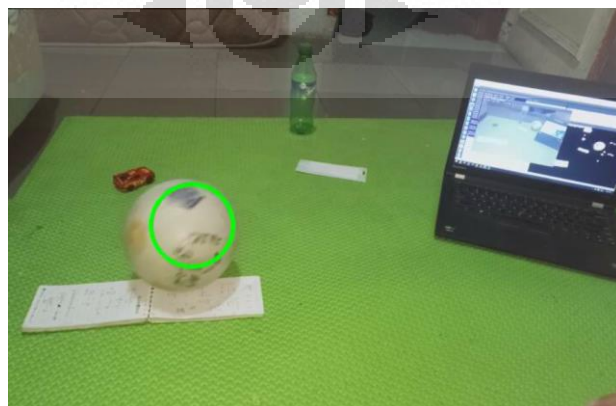
Sama seperti hasil yang didapatkan pada program yang dibuat menggunakan JST, nilai *precision* dan *recall* menjadi nilai yang lebih krusial dalam menentukan performa deteksi. Dengan menerapkan formula *F-score* yang telah disebutkan pada perancangan pengujian terhadap nilai *precision* dan *recall* yang diperoleh, didapatkan hasil *F-score* sebesar 0.285.



Gambar 4.2 Contoh hasil pengujian pendeteksian dengan SVM

Program dengan metode JST memiliki nilai *precision* yang lebih tinggi yaitu sebesar 0.8055 yang menandakan bahwa program dapat membedakan bola dengan *noise* yang ada di sekitar lebih baik dari metode SVM. Sedangkan metode SVM memiliki nilai *precision* sebesar 0.167 yang menandakan bahwa program belum dapat membedakan antara bola dengan *noise* yang ada di sekitar dengan baik, hal ini dapat dilihat pada Gambar 4.2 dimana terdapat banyak *false positive* yang terdeteksi oleh program. Program dengan metode SVM memiliki nilai *recall* yang lebih tinggi yaitu sebesar 0.966, hal ini berarti program dapat menemukan bola lebih baik untuk keseluruhan data dari metode JST. Namun rendahnya nilai *precision* pada metode SVM akan mempersulit penerapan pada skenario nyata pada robot karena terdapat kemungkinan besar bahwa program akan mengikuti objek lain selain bola. Dari kedua hasil yang didapatkan untuk masing-masing metode, dapat disimpulkan bahwa metode yang dibuat menggunakan JST memiliki nilai *F-score* yang lebih tinggi yaitu 0.749 yang berarti metode JST memiliki performa deteksi yang lebih baik. Namun metode JST sedikit lebih lambat dalam mendeteksi bola dimana didapatkan FPS sebesar 6 dibandingkan dengan metode SVM yang memiliki FPS sebesar 7,7.

Adapun hasil dari pengujian proses pelacakan adalah program dapat mengikuti bola dengan cukup baik. Rata-rata FPS yang didapatkan pada saat pengujian adalah sebesar 10 FPS. Program pelacakan dapat tetap mengikuti bola meskipun terdapat objek pada pandangan. Program juga tetap dapat mengikuti bola meskipun bentuk bola menjadi berbayang karena gerakan. Hal ini dapat dilihat pada contoh yang terdapat pada Gambar 4.3.



Gambar 4.3 Contoh proses pelacakan bola

BAB 5

KESIMPULAN

Dari perancangan dan implementasi sistem pengenalan dan pendeteksian objek bola dengan menggunakan metode jaringan saraf tiruan yang telah dilakukan dapat disimpulkan sebagai berikut:

1. Sistem pengenalan dan pendeteksian objek bola dengan menggunakan metode jaringan saraf tiruan telah berhasil dibuat.
2. Sistem mendapatkan hasil pembelajaran paling baik pada proses testing dengan menggunakan jumlah *node* sebanyak 16 pada *hidden layer* dengan nilai MSE sebesar 0,0245.
3. Kinerja sistem dalam mendeteksi bola pada gambar mendapatkan nilai *accuracy* sebesar 0.9987, nilai *precision* sebesar 0.8055, nilai *recall* sebesar 0.7, dan FPS sebesar 6.
4. Kinerja sistem pembandingan yang menggunakan SVM mendapatkan nilai *accuracy* sebesar 0.988, nilai *precision* sebesar 0.167, nilai *recall* sebesar 0.966, dan FPS sebesar 7,7.
5. Kinerja sistem yang dibuat bekerja lebih baik dibandingkan dengan sistem pembandingan berdasarkan nilai *F-score* yang didapatkan yaitu 0.749 pada sistem yang dibuat berbanding dengan 0.285 pada sistem pembandingan.
6. Sistem yang dibuat dapat melakukan pelacakan objek bola dengan kecepatan 10 FPS.

DAFTAR PUSTAKA

- [1] E. Parliament, “What is artificial intelligence and how is it used?” [Daring]. Tersedia di: <https://www.europarl.europa.eu/news/en/headlines/society/20200827STO85804/what-is-artificial-intelligence-and-how-is-it-used>. [Diakses: 29-Dec-2020].
- [2] J. GUPTA, “How AI is used for Computer Vision and Image Recognition?” [Daring]. Tersedia di: <https://www.quytech.com/blog/artificial-intelligence-computer-vision-and-image-recognition/>.
- [3] D. Puspitaningrum, *Pengantar Jaringan Saraf Tiruan*, 1st ed. Yogyakarta: C.V ANDI OFFSET, 2006.
- [4] A. I. Awad and M. Hassaballah, *Image Feature Detectors and Descriptors; Foundations and Applications*, vol. 630. 2016.
- [5] N. Dalal and B. Triggs, “Histograms of oriented gradients for human detection,” in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR’05)*, 2005, vol. 1, pp. 886–893 vol. 1.
- [6] M. Drożdż and T. Kryjak, “FPGA Implementation of Multi-scale Face Detection Using HOG Features and SVM Classifier,” *Image Process. Commun.*, vol. 21, pp. 27–44, Apr. 2017.
- [7] B. Stecanella, “An Introduction to Support Vector Machines (SVM).” [Daring]. Tersedia di: <https://monkeylearn.com/blog/introduction-to-support-vector-machines-svm/>.
- [8] “OpenCV About.” [Daring]. Tersedia di: <https://opencv.org/about/>.
- [9] S. Nissen, “FANN.” [Daring]. Tersedia di: <http://leenissen.dk/fann/wp/>. [Diakses: 22-Jan-2021].
- [10] OpenCV, “cv::HOGDescriptor Struct Reference,” 2020. [Daring]. Tersedia di: https://docs.opencv.org/master/d5/d33/structcv_1_1HOGDescriptor.html. [Diakses: 29-Dec-2020].
- [11] and D. D. B. Dario Albani, Ali Youssef, Vincenzo Suriani, Daniele Nardi, “SPQR Team NAO image dataset,” 2016. [Daring]. Tersedia di:

- <http://www.dis.uniroma1.it/~labrococo/?q=node/459>. [Diakses: 17-Dec-2020].
- [12] V. Alto, “Neural Networks: parameters, hyperparameters and optimization strategies,” 2019. [Daring]. Tersedia di: <https://towardsdatascience.com/neural-networks-parameters-hyperparameters-and-optimization-strategies-3f0842fac0a5>. [Diakses: 29-Dec-2020].
- [13] “What is the vanishing gradient problem?” [Daring]. Tersedia di: <https://deepai.org/machine-learning-glossary-and-terms/vanishing-gradient-problem>. [Diakses: 29-Dec-2020].
- [14] “What is the Exploding Gradient Problem?” [Daring]. Tersedia di: <https://deepai.org/machine-learning-glossary-and-terms/exploding-gradient-problem>. [Diakses: 29-Dec-2020].
- [15] OpenCV, “Introduction to Support Vector Machines,” 2020. [Daring]. Tersedia di: https://docs.opencv.org/3.4/d1/d73/tutorial_introduction_to_svm.html. [Diakses: 29-Dec-2020].
- [16] “Classification: Accuracy.” [Daring]. Tersedia di: <https://developers.google.com/machine-learning/crash-course/classification/accuracy>. [Diakses: 22-Jan-2021].
- [17] “Classification: Precision and Recall.” [Daring]. Tersedia di: <https://developers.google.com/machine-learning/crash-course/classification/precision-and-recall>. [Diakses: 22-Jan-2021].
- [18] T. Wood, “What is the F-score?” [Daring]. Tersedia di: <https://deepai.org/machine-learning-glossary-and-terms/f-score>. [Diakses: 22-Jan-2021].

LAMPIRAN

Data label positif untuk pengujian pendeteksian

