RANCANG BANGUN SISTEM DETEKSI OBYEK MANUSIA SECARA *REALTIME* MENGGUNAKAN JARINGAN SYARAF TIRUAN METODE *BACKPROPAGATION*

SKRIPSI

Oleh:
SATRIO YUDANTO
135090800111006

JURUSAN FISIKA
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
UNIVERSITAS BRAWIJAYA
MALANG
2018

RANCANG BANGUN SISTEM DETEKSI OBYEK MANUSIA SECARA *REALTIME* MENGGUNAKAN JARINGAN SYARAF TIRUAN METODE *BACKPROPAGATION*

SKRIPSI

Sebagai salah satu syarat untuk memperoleh gelar
Sarjana Sains dalam bidang Fisika

Oleh:
SATRIO YUDANTO
135090800111006

JURUSAN FISIKA
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
UNIVERSITAS BRAWIJAYA
MALANG
2018

(Halaman ini sengaja dikosongkan)

LEMBAR PENGESAHAN SKRIPSI
RANCANG BANGUN SISTEM DETEKSI OBYEK MANUSIA SECARA *REALTIME* MENGGUNAKAN JARINGAN SYARAF TIRUAN METODE *BACKPROPAGATION*

SKRIPSI
Oleh:
SATRIO YUDANTO
135090800111006

Setelah dipertahankan di depan Majelis Penguji pada
Tanggal .....
dinyatakan memenuhi syarat untuk memperoleh gelar Sarjana Sains dalam bidang Fisika

Pembimbing I	Pembimbing II
Dr. Eng. Agus Naba, S.Si., MT.	**Ahmad Nadhir, S.Si, MT, Ph.D**
NIP. 197208061995121001	**NIP. 197412031999031002**

Mengetahui,
Ketua Jurusan Fisika
Fakultas Matematika dan Ilmu Pengetahuan Alam
Universitas Brawijaya

Prof. Dr. rer.nat. Muhammad Nurhuda
NIP. 19640910.199002.1.001

(Halaman ini sengaja dikosongkan)

LEMBAR PERNYATAAN

Saya yang bertanda tangan di bawah ini:

Nama **: SATRIO YUDANTO**

NIM **: 135090800111006**

Jurusan **: FISIKA**

Penulis Skripsi berjudul :

RANCANG BANGUN SISTEM DETEKSI OBYEK MANUSIA SECARA *REALTIME* MENGGUNAKAN JARINGAN SYARAF TIRUAN METODE *BACKPROPAGATION*

Dengan ini menyatakan bahwa:

1. Isi dari Skripsi yang saya buat adalah benar-benar karya sendiri dan tidak menjiplak karya orang lain. Nama-nama yang termaksud di isi dan tertulis di daftar pustaka digunakan sebagai referensi pendukung dalam skripsi ini.
2. Apabila di kemudian hari ternyata Skripsi yang saya tulis terbukti hasil jiplakan, maka saya akan bersedia menanggung segala resiko yang akan saya terima.

Demikian pernyataan ini dibuat dengan segala kesadaran.

Malang, 19 Februari 2018
Yang menyatakan,

(SATRIO YUDANTO)

NIM. 135090800111006

(Halaman ini sengaja dikosongkan)

RANCANG BANGUN SISTEM DETEKSI OBYEK MANUSIA SECARA REALTIME MENGGUNAKAN JARINGAN SYARAF TIRUAN METODE BACKPROPAGATION

ABSTRAK

Perkembangan pada pengolahan citra sudah banyak dikembangkan pada berbagai perangkat. Proses pengolahan citra semakin hari semakin berkembang dan dapat digunakan untuk berbagai macam hal, pendeteksian obyek salah satunya. Deteksi obyek sudah banyak diaplikasikan dalam berbagai hal, misalnya *face detection* pada ponsel cerdas yang dapat digunakan ketika akan melakukan pemotretan. Pada penelitian ini dibuat sistem deteksi obyek untuk manusia. Dalam penelitian ini digunakan ekstraksi fitur mirip-Haar yang digabungkan dengan pengklasifikasi jaringan syaraf tiruan menggunakan metode *backpropagation* yang seringkali digunakan dalam bidang kecerdasan buatan. Pada pelatihan jaringan syaraf tiruan digunakan data nilai fitur dari sampel citra positif dan citra negatif. Jaringan syaraf tiruan yang sudah dilatih dan diuji kemudian akan diaplikasikan secara *realtime* menggunakan video yang diambil ditempat yang memuat obyek manusia. Dalam penelitian ini didapatkan nilai akurasi sistem yang berbeda-beda yang dapat dipengaruhi oleh berbagai faktor. Pada implementasinya secara *realtime*, sistem sudah dapat membedakan obyek manusia dengan cukup baik, tetapi akurasi dari sistem masih dapat ditingkatkan dengan menambahkan metode pada tahapan *pre-processing*.

Kata kunci: *OpenCV, ekstraksi fitur mirip-Haar, deteksi obyek, jaringan syaraf tiruan.*

(Halaman ini sengaja dikosongkan)

REAL TIME HUMAN DETECTION SYSTEM PROTOTYPE USING ARTIFICIAL NEURAL NETWORK BACKPROPAGATION METHOD

ABSTRACT

Developments in image processing have been widely developed on various devices. Image processing is growing rapidly every day and can be used for various things, for example object detection. Object detection has been used in many ways, face recognition on smartphone is one of the example, used to take a picture. Object detection for human detection had been done in this research. The research used Haar-like feature extraction combined with artificial neural network using backpropagation method that is commonly used in artificial intelligence. Training of artificial neural network used data features values from the sample of positif image and negatif image. Artifical neural network that has been trained was applied in realtime video taken at a place that contains human as an objects. In this research, the value of system accuracy can be influence by many factors. In realtime implementation, system has been able to detect human fairly well, but the accuracy of the system still can improved by adding pre-processing method called histogram normalization.

Keywords: *OpenCV, Haar-like feature extraction, object detection, artificial neural network.*

(Halaman ini sengaja dikosongkan)

KATA PENGANTAR

Puji syukur penulis panjatkan kehadirat Allah SWT, karena berkat Rahmat dan Karunia-Nya penulis dapat menyelesaikan penulisan laporan skripsi dengan judul **" RANCANG BANGUN SISTEM DETEKSI OBYEK MANUSIA SECARA REALTIME MENGGUNAKAN JARINGAN SYARAF TIRUAN METODE BACKPROPAGATION".** Shalawat beserta salam semoga senantiasa terlimpah curahkan kepada Nabi Muhammad SAW, keluarga dan beseta sahabatnya.

Penulisan skripsi ini dilakukan dalam rangka memenuhi salah satu syarat untuk memperoleh gelar Sarjana dalam bidang Sains Jurusan Fisika, Fakultas Matematika dan Ilmu Pengetahuan Alam, Universitas Brawijaya. Dalam penyusunan dan penulisan skripsi ini tidak terlepas dari bantuan, bimbingan serta dukungan dari berbagai pihak. Oleh karena itu dalam kesempatan ini penulis menyampaikan terimakasih kepada:

1. Hary Kristanto danYuningsih selaku orangtua, dan adik-adik tecinta yang tidak pernah lelah memberikan dukungan moril, material, segenap doa serta curahan kasih sayang tiada terkira kepada penulis.
2. Keluarga besar penulis yang tidak bisa penulis sebutkan satu persuatu yang telah memberikan dukungan dan segenap doa untuk penulis.
3. Bapak Dr. Eng. Agus Naba S.Si, MT dan bapak Ahmad Nadhir S.Si, MT selaku dosen pembimbing yang selalu bijaksana memberikan nasihat, bimbingan dan waktunya.
4. Seluruh Dosen Jurusan Fisika serta Staff dan Karyawan jurusan Fisika yang telah memberikan ilmu yang sangat bermanfaat dan membantu dalam proses perkuliahan.
5. Untuk calon pendamping hidup. Pejuang cinta dan pembawa harapan yang penuh dengan keanggunan. Yang selalu penulis tautkan dalam doa. Dengan izin Allah akan saling dipertemukan serta disatukan dengan ikatan suci pernikahan. Semoga tetap diberi kesabaran dalam menunggu.

6. Keluarga BEM FMIPA UB masa kepengurusan periode 2014, 2015 dan 2016 terimakasih atas kebersamaan nya.
7. Keluarga PSDM BEM FMIPA MAGNETIK 2016 (Rossy YS, Satria Wira Bagaskara, Dianisari Sofia Ranti, Cessa Mithaco, Selena Bunga Deshinta, Lyshe Martya Herlistriana, Galuh Rahmaniah, Fadhail Faiz Effendi, Syamaidzar, Mochammad Zainal, Muhammad Faisal, Sakhiyyah Afifah, Widiarni Ginta Sasmita, Mochammad Irfanudin). Anak-anak, adik-adik tersayang. Yang selalu ada di hati penulis. Yang keunikan fisik dan mental nya selalu akan penulis kenang sebagai sesuatu yang tidak bermanfaat.
8. Keluarga PSDM BEM FMIPA REVOLUSI 2017 (Dianisari Sofia Ranti, Syamaidzar, Lyshe Martya Herlistriana, Nabila Nurvelia, Rafika Choirunnisa, Farida Rachmawati, Sakhiyyah Afifah, Widiarni Ginta Sasmita, Adam LC, Feryan Adi Anggana, Muhammad Zulkifli, Mochammad Irfanudin) yang sudah rela penulis ganggu kehidupannya selama di kepengurusan BEM FMIPA.
9. Divisi acara BOKER PROBINMABA FMIPA 2015 (Risa, Laila, Sari, Yola, Eyi, Salim, Rossy dan Satria) yang bersedia waktunya digunakan untuk hal-hal yang tidak berguna selama beberapa bulan dan seterusnya ketika bersama dengan penulis.
10. Keluarga instrumentasi (IC dan IMFI) Universitas Brawijaya yang selalu menyediakan tempat untuk berbagi cerita
11. Anggia Noor Ramadhani, Muhammad Fikri Salim, Naufal Muhammad Hirzi, Dianisari Sofia Ranti, Qurrota A'yun Masyhur, Lina Fitriyana, Effrihan, Alfian Afan Ghafar, Shinta Irabyatul Rahman, Febby Nurdia Ningsih dan yang lainnya yang tidak bisa disebutkan satu persatu sebagai sahabat, kakak, saudara dan sebutan dekat dan hangat lainnya. Yang dengan sukarela kehidupannya diganggu.

Seluruh pihak yang terkait yang tidak bisa penulis sebutkan satu persatu, terimakasih atas segala batuan dan dukungannya.

Penulis menyadari bahwa penulisan ini masih terdapat kekurangan baik dalam penyusunan, bahasa dan penyajian penjelasannya. Oleh karena itu penulis mengharapkan saran dan kritik dari pembaca sehingga dapat memberikan perubahan ke arah yang lebih baik. Penulis berharap dengan adanya laporan ini dapat memberikan pengetahuan dan manfaat kepada pembaca, terutama dalam pengembangan ilmu pengetahuan.

Malang, 19 Februari 2018

Penulis

(Halaman ini sengaja dikosongkan)

DAFTAR ISI

DAFTAR GAMBAR

(Halaman ini sengaja dikosongkan)

DAFTAR TABEL

(Halaman ini sengaja dikosongkan)

DAFTAR LAMPIRAN

(Halaman ini sengaja dikosongkan)

BAB I
PENDAHULUAN

1.1 Latar Belakang

Hingga kini, perkembangan pada pengolahan citra sudah banyak diterapkan pada berbagai perangkat, mulai dari komputer hingga perangkat mikrokontroler. Proses pengolahan citra semakin hari semakin berkembang, hingga kini pengolahan citra juga dapat dilakukan untuk menganalisis video. Pemilihan metode yang tepat yang digunakan untuk menganalisis video sangatlah penting karena kompleksitas dari algoritma yang digunakan akan sangat mempengaruhi kecepatan dari respon analisis (Pradana dkk., 2016).

Deteksi obyek merupakan bagian dari kehidupan sehari-hari manusia. Manusia sering mendeteksi banyak obyek, seperti misalnya sesama manusia, bangunan, kendaraan dan lain-lain. Hingga kini masih menjadi misteri bagaimana manusia dapat mendeteksi obyek disekitarnya secara akurat dengan tanpa membutuhkan usaha yang besar. Dari misteri tersebut telah menantang para psikolog maupun ahli fisiologi untuk menemukan penjelasan yang komprehensif bagaimana hal tersebut dapat terjadi selama lebih dari satu abad. Mendeteksi obyek secara otomatis memerlukan usaha yang besar. Selama lebih dari 30 tahun, penelitian pada *computer vision* menghasilkan progres yang sedikit. Tantangan pada deteksi obyek secara otomatis ada pada jumlah variasi yang muncul. Sebuah detektor obyek haruslah dapat mengatasi tantangan keduanya, yaitu variasi yang ada pada suatu obyek maupun keragaman citra visual yang ada. Salah satu contohnya adalah mobil, yang memiliki variasi dalam segi ukuran, warna, bentuk bahkan hal detilnya seperti dari ban, lampu maupun *grille*. Sehingga algoritma deteksi mobil juga harus dapat membedakan antara mobil dengan berbagai pola yang ada di sekitarnya (Schneiderman dan Kanade, 2004).

Tujuan dari adanya deteksi suatu obyek adalah agar dapat mendeteksi semua obyek yang diketahui seperti contohnya yaitu mobil atau wajah pada suatu citra. Sistem deteksi obyek akan menghasilkan suatu model dari suatu obyek yang dideteksi yang sebelumnya telah dilakukan pelatihan. Metode pada deteksi obyek secara umum dibagi menjadi dua kategori, yaitu deteksi obyek

generative dan deteksi obyek *discriminative*. Pada deteksi dengan kategori *generative*, model parameter dapat diperkirakan cukup dengan menggunakan data pelatihan dan keputusan. Untuk deteksi kategori *discriminative*, dapat membuat klasifikasi berdasarkan gambar (atau sub-gambar) yang mengandung obyek yang akan dideteksi atau yang tidak mengandung obyek yang akan dideteksi. Parameter dari suatu klasifikasi obyek digunakan untuk meminimalisir kesalahan yang terjadi ketika dilakukan pelatihan data yang berguna untuk menghindari *overvitting* (Amit dan Felzenszwalb, 2013).

Dari uraian tersebut, maka pada penelitian ini dimaksudkan untuk mendeteksi manusia menggunakan jaringan syaraf tiruan dengan metode *backpropagation* yang bertempat di Malang dan sekitarnya. Penulis menggunakan pengklasifikasi jaringan syaraf tiruan yang sudah banyak digunakan untuk berbagai aplikasi analisis citra.

1.2 Rumusan Masalah

Rumusan masalah pada penelitian ini meliputi:

1. Bagaimana melakukan pelacakan manusia dalam citra digital dengan jaringan syaraf tiruan?
2. Bagaimana arsitektur jaringan syaraf tiruan yang digunakan dalam penelitian?
3. Bagaimana kemampuan program yang digunakan untuk mengidentifikasi obyek manusia?

1.3 Batasan Masalah

Batasan dari penelitian yang dilakukan yaitu:

1. Akuisisi video yang digunakan untuk analisis kemampuan dari program dilakukan di Kota Malang dan Batu
2. Pada proses pra-pemrosesan tidak dilakukan normalisasi histogram

3. Pola yang dikenalkan pada jaringan syaraf tiruan hanya tampak depan dari obyek manusia
4. Obyek akan mulai terdeteksi pada jarak 2-3 meter dari kamera

1.4 Tujuan

Tujuan dari penelitian ini yaitu:

1. Melakukan pelacakan manusia menggunakan jaringan syaraf tiruan.
2. Merancang arsitektur jaringan syaraf tiruan yang digunakan dalam penelitian.
3. Mengetahui kemampuan dari program yang dibuat untuk identifikasi obyek manusia.

1.5 Manfaat

Dari penelitian yang dilakukan diharapkan dapat memberikan manfaat penjelasan tentang perancangan jaringan syaraf tiruan yang meliputi arsitektur, algoritma training, model, algoritma pemrograman untuk deteksi manusia dengan menggunakan metode *backpropagation*. Pada implementasinya, penelitian ini diharapkan dapat bermanfaat dan dikembangkan lebih jauh untuk diterapkan dalam pembatasan jumlah manusia pada suatu tempat.

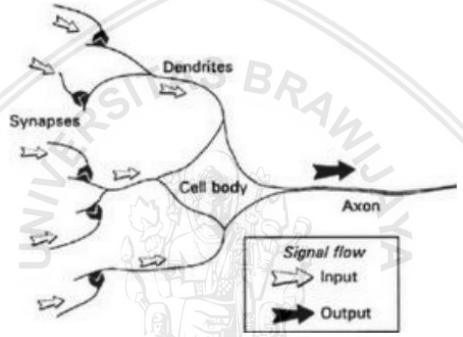
(Halaman ini sengaja dikosongkan)

BAB II
TINJAUAN PUSTAKA

2.1 Jaringan Syaraf Tiruan

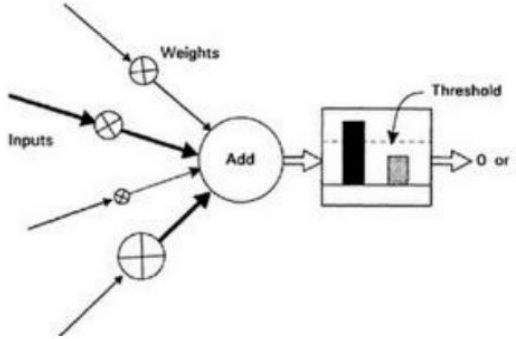
2.1.1 Permodelan

Neural network merupakan suatu perakitan sederhana dari suatu elemen pemrosesan yang terdiri dari unit, yang memiliki fungsi berdasarkan neuron manusia. kemampuan pemrosesan dari jaringan akan disimpan pada bobot, yang didapat sebelumnya dari proses pelatihan.



Gambar 2.1. Pemodelan Jaringan Syaraf Tiruan

Otak manusia diperkirakan memiliki kurang lebih 100 juta jaringan saraf. Gambar 2.1 menunjukkan pemodelan otak manusia, dimana satu neuron dengan neuron lain dapat berkomunikasi menggunakan sinyal elektrik. Koneksi antar neuron memiliki perantara yang disebut dengan sinapsis, yang bertempat pada cabang sel yang bisa disebut dengan dendrit. Masing-masing neuron menerima ribuan koneksi dari neuron lain dan tak jarang bahkan bisa menerima sinyal dalam jumlah yang sangat besar pada satu neuron nya, yang pada akhirnya akan diterima pada sel tubuh. Nantinya neuron akan terintegrasi satu dengan yang lainnya dan akan berkomunikasi.



Gambar 2.2. Model Neuron

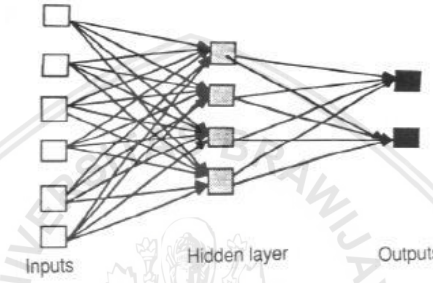
Neuron biologis dapat dimisalkan sebagai unit pada jaringan syaraf tiruan. Kemudian sinapsis dimodelkan dengan bobot sehingga nantinya input akan dijumlahkan dengan bobot sebelumnya sebelum dikirim menuju sel tubuh seperti yang di ilustrasikan gambar 2.2. Bobot sinyal kemudian akan dijumlahkan menggunakan penjumlah aritmatika sederhana untuk mengirimkan sinyal aktivasi, yang dapat didefinisikan sebagai *Threshold Logic Unit* (TLU). Jika sinyal ativasi melebihi *threshold*, maka unit akan menghasilkan nilai output yang tinggi atau didefinisikan dengan "1", begitu juga sebaliknya (Gurney, 2004).

Karakteristik utama dari *Neural Network* adalah bahwa *Artificial Neural Network* memiliki kemampuan untuk mempelajari kompleksitas hubungan antara input dan output nonlinear, dilanjutkan menggunakan prosedur pelatihan yang beurutan dan kemudian menyesuaikan untuk data yang tersedia. Proses pelatihan pada *neural network* melibatkan pembaruan arsitektur jaringan dan koneksi antar bobot sehingga jaringan yang terbuat nanti dapat bekerja secara efisien untuk klasifikasi hal-hal yang spesifik. Jaringan syaraf tiruan menyediakan deretan algoritma nonlinear yang dapat digunakan untuk ekstraksi fitur (menggunakan hidden layer) dan klasifikasi. Sebagai tambahan, ekstraksi fitur dan klasifikasi algoritma juga dapat dipetakan pada arsitektur dari jaringan syaraf tiruan agar dapat bekerja lebih efisien. Sebuah Jaringan syaraf tiruan merupakan suatu pengolahan informasi yang terinspirasi dari jaringan sistem saraf

biologis, seperti pemrosesan informasi yang ada di otak (Bayu, dkk. 2010).

2.1.2 Arsitektur

Menurut Christos (1996) secara umum Artificial Neural Network dibagi menjadi tiga bagian *layer* (lapisan) yang ditunjukkan seperti pada gambar 2.3: yaitu yang terdiri dari *input layer* (lapisan masukan) yang tersambung dengan *hidden layer* (lapisan tersembunyi). Kemudian *hidden layer* (lapisan tersembunyi) yang terhubung dengan *output layer* (lapisan keluaran).



Gambar 2.3. Contoh Jaringan Sederhana

1) Layer input bertanggung jawab terhadap informasi yang akan dikirimkan ke jaringan. Pada layer ini, informasi yang dialirkan masih mentah yang berarti tidak ada suatu proses pengolahan informasi.

2) Sementara aktivitas yang terjadi pada hidden layer ditentukan oleh input yang masuk serta dari bobot yang terletak diantara layer input dan hidden layer.

3) Nilai pada layer output ditentukan dari aktivitas yang terjadi antara hidden unit dengan bobot yang berada di antara hidden layer dengan output layer.

Jaringan yang sederhana tersebut dapat dikatakan menarik, dikarenakan hidden unit memiliki kemampuan untuk merepresentasikan input yang masuk secara bebas.

2.1.3 Metode *Backpropagation*

Salah satu masalah yang sering dijumpai ketika mendesain suatu jaringan syaraf tiruan adalah menghitung jumlah bobot yang tepat yang digunakan untuk aktivitas neuron. Bobot tersebut dapat di dapatkan dari proses pelatihan yang digunakan pada jaringan syaraf tiruan. Pada proses pelatihan, algoritma digunakan untuk menghitung bobot jaringan, sehingga kesalahan kuadrat antara output yang terhitung dan yang teramati dari proses pelatihan dapat di minimalisir (Skorin-Kapov dan Tang, 2001)

Metode backpropagation pada sekarang telah banyak digunakan untuk jaringan syaraf tiruan. Seiring berjalannya waktu metode backpropagation menjadi suatu alat yang sangat baik yang dapat digunakan untuk keperluan pengenalan pola, pemodelan dinamis, analisis sensitivitas, kontrol suatu sistem dan masih banyak lagi. Metode backpropagation juga dapat digunakan pada jaringan syaraf tiruan untuk keperluan di bidang seperti model *econometric*, struktur logika fuzzy dan lain-lain. Pada intinya, metode backpropagation merupakan metode sederhana yang efisien dan metode yang tepat untuk menghitung semua turunan pada satu kuantitas target (contohnya yaitu seperti klasifikasi pola error) (Werbos, 1990).

Fungsi utama dari pelatihan yang dikontrol dapat di temukan pada pengerjaan pengenalan pola. Sebagai contohnya adalah rancang bangun jaringan syaraf tiruan yang digunakan untuk mengenali kode ZIP (Werbos, 1990).

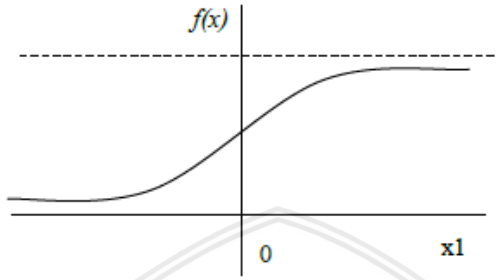
Metode backpropagation, terdapat beberapa syarat yang harus dipenuhi agar fungsi aktivasi backpropagation dapat digunakan. Yaitu: bersifat kontinu, dapat terdiferensial dengan mudah serta fungsi yang digunakan merupakan fungsi yang tidak turun. Fungsi yang sering digunakan dan memenuhi ketiga syarat tersebut salah satunya adalah fungsi sigmoid biner. Fungsi sigmoid biner memiliki rentang (0,1), dengan persamaan

$$f(x) = \frac{1}{1 + e^{-x}} \qquad (2.1)$$

Dan apabila diturunkan menjadi

$$f'^{(x)} = f(x)\left(1 - f(x)\right) \qquad (2.2)$$

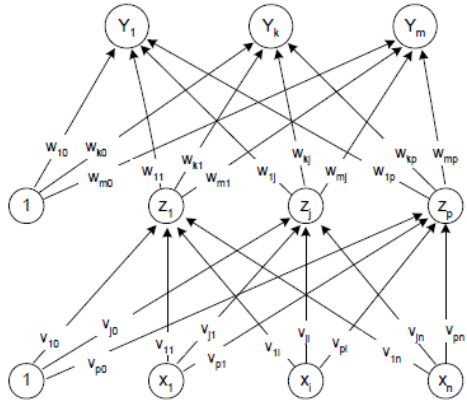
Gambar 2.4 merupakan grafik fungsi dari sigmoid biner



Gambar 2.4. Grafik Fungsi Sigmoid Biner

Fungsi dari sigmoid biner memiliki nilai maksimum 1. Sehingga apabila pola dengan target lebih dari 1, pola masukan dan keluaran haruslah di transformasikan terlebih dahulu agar memiliki range yang sama dengan fungsi sigmoid biner (Lesnussa dkk., 2015).

Metode backpropagation memiliki kemampuan untuk melatih jaringan agar mendapatkan keseimbangan kemampuan jaringan syaraf tiruan untuk mengenali pola yang digunakan pada latihan sehingga dapat memeberikan respon terhadap pola masukan yang serupa dengan pola yang dipakai selama proses *training*. Backpropagation memiliki unit yang ada dalam *hidden layer*. Gambar 2.5 merupakan contoh dari arsitektur backpropagation dengan n buah masukan dan sebuah *hidden layer* sejumlah *p* unit serta unit keluaran sejumlah m buah dimana masukan dan unit dari *hidden layer* akan ditambakan dengan sebuah bias. Gambar 2.5, V_{ji} adalah bobot garis yang menghubungkan unit masukan dari X_i ke unit *hidden layer* Z_j sementara V_{j0} adalah bobot garis yang menghubungkan bias masukan ke hidden layer Z_j. W_{kj} menunjukkan bobot dari *hidden layer* yang ada pada Z_j ke keluaran Y_k (Siang, 2005).



Gambar 2.5. Arsitektur Backpropagation

Menurut Siang (2005) terdapat tiga fase yang harus dilakukan dalam pelatian dengan metode backpropagation. Fase yang pertama yaitu fase maju. Fase maju pola dari masukan akan dihitung maju dimulai dari layar masukan sampai ke layar keluaran dengan menggunakan fungsi aktivasi yang telah ditentukan. Fase berikutnya merupakan fase mundur, dimana selisih antara keluaran dari jaringan dan target merupakan kesalahan. Kesalahan yang didapat tersebut akan di propagasi mundur, dari garis-garis yang berhubungan dengan *output layer*. Dan fase yang terakhir yaitu bobot akan dimodifikasi ulang yang berguna agar kesalahan yang terjadi dapat diturunkan pada proses pelatihan selanjutnya dan dapat mendekati nilai dari target yang diharapkan.

Fase I : Propagasi Maju

Propagasi maju sinyal masukan (X_i) akan dipropagasi ke *hidden layer* dengan fungsi aktivasi yang telah ditentukan. Kemudian keluaran dari *hidden layer* (Z_j) selanjutnya akan dipropagasi ke *hidden layer* diatasnya dengan menggunakan fungsi aktivasi yang telah ditentukan. Begitu seterusnya hingga didapatkan nilai pada *output* (Y_k) dari jaringan tersebut. Selanjutnya *output* (Y_k) tersebut akan dibandingkan dengan nilai target (T_k) yang telah ditentukan. Seperti yang sudah disampaikan Siang (2005) sebelumnya bahwa selisih dari T_k -Y_k merupakan kesalahan. Apabila nilai kesalahan tersebut bernilai lebih

kecil dibandingkan dengan batas toleransi yang diberikan, maka proses pelatihan akan dihentikan. Tetapi apabila nilai kesalahan tersebut lebih besar dibandingkan dengan toleransi yang diberikan, maka bobot pada garis yang ada dalam jaringan akan dimodifikasi untuk mengurangi kesalahan dan dapat mendekati target yang telah ditentukan.

Fase II : Propagasi Mundur

Setelah didapatkan nilai kesalahan dari selisih T_k - Y_k, selanjutnya dihitung faktor S_k (k= 1, 2, ..., m) yang digunakan untuk mendistribusikan kesalahan yang ada di Y_k ke semua layer yang berhubungan langsung dengan Y_k. S_k juga digunakan untuk mengubah bobot garis yang berhubungan dengan *output*. Dengan menggunakan cara yang sama, faktor S_j dihitung pada setiap unit di *hidden layer* yang digunakan sebagai dasar merubah bobot pada semua garis yang berasal dari unit dibawahnya. Cara tersebut digunakan hingga semua faktor *S* di setiap unit tersembunyi yang berhubungan langsung dengan unit masukan dihitung.

Fase III : Perubahan Bobot

Yang terakhir yaitu fase perubahan bobot. Setelah faktor *S* telah terhitung, maka bobot pada semua garis akan dimodifikasi secara bersamaan. Perubahan bobot pada garis didasarkan atas faktor *S* neuron di layer diatasnya. Sebagai contoh, perubahan bobot garis yang menuju *output layer* didasarkan pada S_k yang ada di unit keluaran.

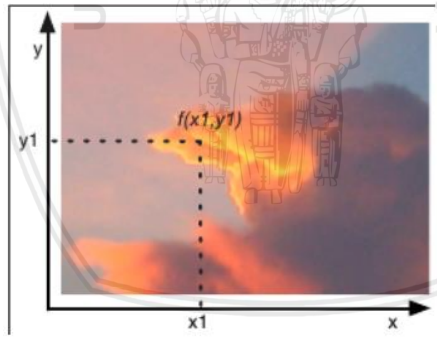
Ketiga fase tersebut akan diulang-ulang hingga kondisi penghentian terpenuhi. Kondisi penghentian yang biasanya digunakan adalah kesalahan. Yaitu proses pelatihan akan dihentikan apabila nilai kesalahan lebih kecil dari nilai toleransi yang diijinkan.

2.2 Citra Digital

Pengolahan citra atau yang bisa dikenal juga dengan pemrosesan citra, dapat digunakan komputer digital agar citra yang dimanipulasi dapat menghasilkan kualitas citra yang lebih baik. Juga dapat berfungsi agar citra tersebut dapat dikenali dengan mudah oleh

manusia ataupun mesin. *Computer vision* dapat menghasilkan suatu informasi dari citra digital yang digunakan untuk mengenali pola, sehingga informasi tersebut dapat diolah lebih jauh. Agar komputer dapat melakukan pengenalan pola, maka citra digital yang di dapat perlu diperbaiki dikenal dengan *image restoration* yang berfungsi agar komputer dapat mengenali citra tersebut. Citra digital memiliki keunikan, seperti dari ukuran citra dan format lainnya. Maka dari itu, citra digital diharuskan memiliki format agar dapat mewakili suatu obyek tertentu. Format yang biasanya digunakan yaitu seperti *grayscale* dan warna (Muhtadan dan Harsono, 2008).

Citra merupakan suatu representasi dari suatu obyek yang tertangkap. Pada kehidupan sehari-hari citra dapat dikelompokkan menjadi dua jenis, yaitu citra tampak dan citra tidak tampak. Citra tampak dapat dicontohkan dengan gambar atau lukisan sedangkan citra tidak tampak dapat dicontohkan dengan data gambar yang ada dalam file atau bisa disebut dengan citra digital yang dicontohkan seperti pada gambar 2.6, yang dalam hal ini merupakan sesuatu yang dapat diolah dengan komputer. Pencitraan dapat diartikan sebagai suatu kegiatan yang berguna untuk mengubah citra fisik non digital menjadi digital.



Gambar 2.6. Citra digital

Citra digital dapat difungsikan sebagai variabel *f(x,y)* di mana *x* dan *y* merupakan suatu koordinat dan nilai dari *f(x,y)* dapat diartikan sebagai intensitas citra. Teknologi untuk menciptakan serta menampilkan warna pada citra digital adalah kombinasi dari tiga warna dasar yaitu *Red, Green, Blue* atau warna RGB (Gazali dkk., 2012).

2.3 Bahasa Pemrograman C++

Bahasa pemrograman atau biasa yang dikenal dengan bahasa komputer merupakan suatu teknik instruksi yang digunakan untuk memerintahkan komputer. Bahasa pemrograman dapat digunakan untuk membangun suatu aplikasi yang didasarkan pada kebutuhan pada berbagai bidang, contohnya seperti bidang pendidikan, sosial budaya dan masih banyak lagi. Saat ini, pada komputer terdapat banyak bahasa pemrograman yang dapat digunakan, salah satu nya adalah bahasa *C++*. Bahasa *C++* merupakan bahasa pemrograman yang dikembangkan dari bahasa sebelumnnya yaitu bahasa *C* (Dewi, 2010).

Pada konteks pemrograman, terdapat beberapa bahasa pemrograman yang dapat digunakan, yaitu seperti bahasa *Pascal*, *C* dan *C++*. Secara umum, bahasa pemrograman dapat dibagi menjadi dua macam, yaitu *high-level language* dan *low-level language*. Bahasa tingkat tinggi pada bahasa pemrograman berorientasi pada bahasa manusia yang dibuat agar manusia dapat dengan mudah memahami program yang dibuat. Biasanya menggunakan bahasa inggris. Contohnya adalah *AND* untuk menyatakan dan. Bahasa *C* dan Bahasa *C++* merupakan contoh dari *high-level language*. Sedangkan *low-level language* merupakan bahasa pemrograman yang berorientasi pada mesin, yang artinya bahasa ini menggunakan kode-kode tertentu. Contohnya adalah dengan penggunaan kode *biner* yang hanya mengenal angka 0 dan 1 (Heriyanto, 2005).

2.4 OpenCV

OpenCV atau dapat dikenal dengan *Open Computer Vision* merupakan pustaka gratis yang disediakan dan dikembangkan oleh Intel Corporation yang dikhususkan untuk melakukan suatu *image processing*. Tujuan dari diciptakannya OpenCV ini adalah agar komputer dapat memiliki kemampuan yang hampir sama dengan manusia dalam aspek pengolahan visual. OpenCV mempunyai *Application Programming Interface* yang digunakan untuk *high-level language* ataupun *low-level language*. Dari *Application Programming Interface* tersebut, terdapat fungsi yang dapat digunakan untuk gambar maupun video, seperti untuk *saving* dan *loading* (Mulyawan dkk., 2011).

OpenCV pertama kali dibuat oleh Gary Bradski pada tahun 1999 pada perusahaan Intel yang di gunakan untuk mempercepat penelitian pada *computer vision* di dunia. Sedangkan untuk Intel sendiri, OpenCV digunakan untuk membuat komputer yang memiliki kemampuan yang lebih tangguh. Vadim Pisarevsky memutuskan untuk bergabung dengan Gary untuk mengelola tim Intel OpenCV yang ada di Rusia. Pustaka *open-source computer-vision* milik intel memiliki pengaruh untuk mempermudah ketika memprogram *computer-vision*. Dalam pustaka tersebut terdapat kemampuan yang mumpuni, seperti deteksi wajah, pelacakan wajah, pengenalan wajah dan masih banyak lagi metode kecerdasan buatan yang siap untuk digunakan. Selain itu, OpenCV juga menyediakan algoritma dasar dari *computer-vision* (Emami dan Suciu, 2012).

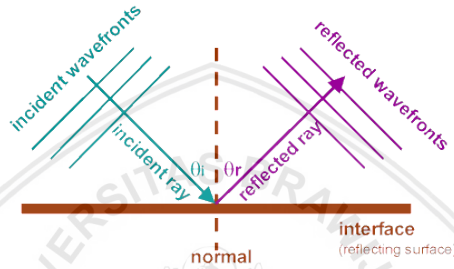
Pustaka yang ada pada OpenCV memiliki lebih dari 2500 algoritma, yang didalam nya terdapat perkembangan mutakhir *computer vision* dan pelatihan algoritma mesin. Pustaka OpenCV telah digunakan pada perusahaan, badan penelitian dan bahkan oleh badan pemerintahan. OpenCV dapat digunakan dengan bahasa C++, C, Python dan Java (Anonim, 2017).

OpenCV sebagai jaringan multi-platform memiliki beberapa keuntungan, yaitu diantaranya OpenCV dapat digunakan pada Windows dan Linux, dan bahkan Mac OS X. OpenCV memiliki banyak kemampuan yang luar biasa. Pemahaman yang baik tentang bagaimana cara kerja OpenCV diperlukan agar dapat dihasilkan hasil yang baik ketika menggunakan OpenCV (Emami dan Suciu, 2012).

2.5 Cahaya

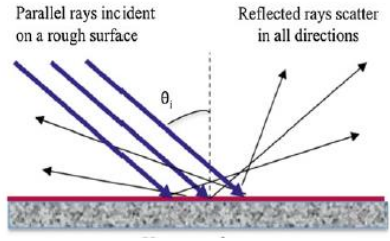
Cahaya dapat diartikan sebagai suatu spektrum elektromagnetik yang dapat ditangkap oleh mata. Cahaya yang dapat dirasakan ataupun dilihat oleh mata dapat dikatakan sebagai cahaya tampak. Sementara dengan penglihatan maka manusia dapat merasakan ada nya cahaya. Bagaimana penglihatan dapat merasakan cahaya memiliki kesamaan dengan bagaimana kamera dapat bekerja. Secara umum, cahaya terbentuk dari milyaran foton yang dapat dihasilkan dari sumber cahaya ataupun pantulan dari suatu obyek (Anderson Sihombing, 2008).

Jumlah cahaya yang terpantul ke berbagai arah oleh suatu obyek tertentu bergantung dengan baik atau tidak nya tempat permukaan cahaya tersebut dipantulkan. Pemantulan yang terjadi pada permukaan yang halus seperti cermin dapat dikenal dengan *specular reflection* seperti yang terlihat pada gambar 2.7. Ketidaksempurnaan permukaan memiliki nilai lebih kecil dibandingkan dengan panjang gelombang cahaya yang datang. Berdasarkan hukum Snell, cahaya yang datang terpantul pada sudut yang sama ketika dipantulkan (Keiser, 2016).



Gambar 2.7. Pemantulan teratur (*specular reflection*)

Sementara *diffuse reflection* atau pemantulan yang tersebar dapat disebabkan oleh beberapa hal, yaitu diantaranya permukaan pantulan yang kasar, tidak rata, dan berpori. Tipe pemantulan *diffuse reflection*, cahaya yang terpantul akan menyebar ke banyak arah, seperti yang terlihat pada gambar 2.8. Pada kenyataannya di dunia ini tidak semua permukaan memiliki bentuk yang halus, sehingga banyak menyebabkan cahaya terpantul secara menyebar ke berbagai arah (Keiser, 2016).



Gambar 2.8. Pemantulan menyebar (*diffuse reflection*)

Sebagai contohnya, perbedaan dari kedua jenis pantulan tersebut dapat menjelaskan bagaimana manusia dapat dengan sulit mengemudi pada malam hari ketika cuaca sedang hujan. Apabila keadaan jalan tersebut basah, permukaan halus dari air tersebut dapat memantulkan secara teratur dan cahaya yang datang dari lampu mobil akan ke arah menjauhi mobil. Kebalikan dari sebelumnya, ketika kondisi jalan kering, maka permukaan kasar menyebabkan cahaya terpantul ke berbagai arah dan dapat menyebabkan cahaya yang dikeluarkan dari lampu mobil kembali ke arah pengemudi, sehingga menyebabkan pengemudi dapat melihat jalan dengan lebih jelas (Emeritus, 2004).

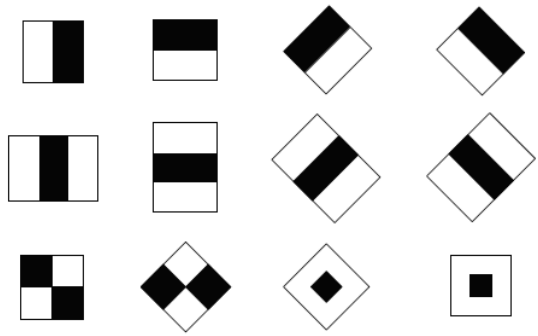
Cahaya yang datang dan cahaya yang terpantul akan membentuk sudut θ dan θ'. Kedua hubungan tersebut dapat dikenal dengan Hukum Pemantulan. Pada teori dan percobaannya menunjukkan bahwa sudut dari cahaya yang dipantulkan sama dengan cahaya yang datang yang ditunjukkan pada persamaan di bawah ini. (Emeritus, 2004):

$$\theta_1' = \theta_1$$

2.6 Ekstraksi Fitur Mirip-Haar

Fitur mirip-Haar merupakan fitur berbentuk persegi panjang yang dapat digunakan untuk menunjukkan karakteristik yang khusus pada sebuah gambar. Ide utama dibalik pengembangan fitur mirip Haar adalaha untuk mengenali suatu obyek atau fitur yang didasarkan atas jumlah fitur sederhana, bukan jumlah pixel secara langsung. Fitur mirip-Haar terdiri dari dua atau tiga persegi panjang berwarna hitam dan putih yang digunakan untuk menghitung perbedaan jumlah intensitas pixel pada suatu citra. Gambar 2.9 menunjukkan beberapa fitur mirip-Haar. Persamaan 2.3 menunjukkan cara perhitungan untuk menghitung nilai fitur haar dengan menggunakan selisih perbedaan jumlah pixel (Maheswari, 2015).

$$f(x) = \sum F\ black\ (jumlah\ pixel) - \sum F\ white\ (jumlah\ pixel) \quad (2.3)$$



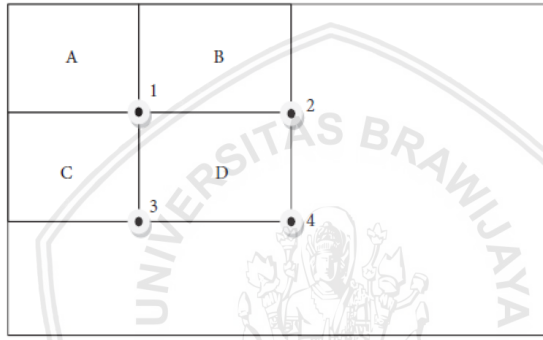
Gambar 2.9. Beberapa fitur mirip-Haar

Pendeteksian wajah juga merupakan bagian dari *face recognition*. Viola dan Jones mengembangkan metode untuk mengenali suatu pola dengan ekstraksi fitur mirip haar (*Haar-like feature*). Tiga hal penting yang perlu diperhatikan dalam pengenalan pola tersebut yaitu adalah penggunaan citra integral yang berfungsi untuk mempercepat penghitungan fitur, kemudian algoritma pembelajaran berbasis AdaBoost, dan yang terakhir menggunakan klasifikasi bertingkat (Firdausy, 2013).

Fitur mirip-Haar juga dapat diaplikasikan untuk klasifikasi obyek generik. Fitur mirip-Haar merupakan salah satu metode yang familiar digunakan untuk deteksi wajah, dimana sistem akan menentukan apakah obyek tersebut dapat dikategorikan sebagai wajah. Cukup dengan mengetahui bahwa suatu obyek dapat dikategorikan sebagai wajah maka juga akan sangat berguna untuk segmentasi gambar. Secara teknis, fitur mirip-Haar mengacu pada bagaimana memotong-motong suatu gambar agar dapat mengidentifikasi pola utama pada gambar tersebut. Informasi tersebut biasanya akan disimpan pada haar-cascade, yang biasanya file tersebut akan berformat XML. Metode ini tentunya akan tetap membutuhkan pekerjaan yang cukup banyak untuk melatih sistem pengklasifikasi hingga akhirnya dapat menghasilkan file cascade. Metode penghitungan untuk fitur mirip-Haar akan dapat lebih cepat jika digunakan dengan citra integral (Mustafa dkk., 2014).

2.7 Citra Integral

Metode penghitungan dengan fitur mirip-Haar dapat dipercepat dengan penggunaan citra integral atau penjumlahan area tabel. Karena hal tersebut juga maka klasifikasi *Haar-cascade* dan *train-cascade* dapat dikomputasikan dengan sangat cepat. Fitur citra dua dimensi dapat dikomputasikan dengan cepat menggunakan suatu representasi yang dikenal dengan citra integral. Sebuah citra integral yang ditunjukkan dengan *ii* (*x*,*y*), pada lokasi (*x*,*y*) merupakan suatu penjumlahan dari pixel yang ada di bagian atas dan kiri dari (*x*,*y*). Jumlah dari citra integral pada posisi (*x*,*y*) merupakan penjumlahan dari semua pixel yang ada di kiri atas.



Gambar 2.10. Kotak penjumlahan

Hal tersebut didasarkan pada persamaan:

$$ii\,(x,y) = \sum_{x' \le x, y' \le y} i\,(x', y') \qquad (2.4)$$

dimana *ii* (*x*,*y*) merupakan citra integral dan *i* (*x*,*y*) adalah citra asli yang didasarkan pada persamaan berikut:

$$s\,(x, y) = s\,(x\,, y\text{-}1) + i\,(x\,, y), \qquad (2.5)$$

$$ii\,(x, y) = ii\,(x - 1, y) + s\,(x\,, y). \qquad (2.6)$$

Gambar 2.10 mengilustrasikan penggunaan metode penjumlahan area tabel. Sebagai contoh dengan hanya menggunakan empat titik

referensi, penjumlahan pixel yang ada pada persegi D dapat dihitung dengan cara berikut; pada lokasi 1 merupakan jumlah dari pixel di persegi A, pada lokasi 2 merupakan penjumlahan dari persegi A+B, pada lokasi 3 merupakan jumlah dari persegi A+C, dan pada lokasi 4 merupakan jumlah dari persegi A+B+C+D. Sehingga jumlah dari persegi D akan menjadi 4+1-(2+3) (Mustafa dkk., 2014).

(Halaman ini sengaja dikosongkan)

BAB III
METODE PENELITIAN

3.1 Waktu dan Tempat Penelitian

Penelitian ini dilaksanakan di Laboratorium Komputasi dan Pemodelan Jurusan Fisika Fakultas MIPA Universitas Brawijaya dan Laboratorium Instrumentasi Jurusan Fisika Fakultas MIPA Universitas Brawijaya. Tempat pengambilan data dilaksanakan di lingkungan Universitas Brawijaya Malang, Batu Secret Zoo Jawa Timur Park II, dan Alun – alun kota Malang. Penelitian ini dilakukan selama bulan Desember 2016 – Desember 2017.

3.2 Persiapan Perangkat Keras dan Perangkat Lunak

Perangkat keras yang digunakan pada penelitian ini adalah *personal computer* dengan spesifikasi sebagai berikut:

Tabel 3.1. Spesifikasi *Personal Computer* yang digunakan

Komponen	Keterangan
CPU	Intel® Core™ i5-7200U @2.5Ghz (4 CPUs), ~2.7Ghz
GPU	Intel® HD Graphics 620 & NVIDIA GeForce 920MX
RAM	8192MB DDR4
Kamera USB	SJCam SJ4000 *Action Camera*

Sedangkan *software* yang digunakan selama penelitian tertera pada tabel 3.2.

Tabel 3.2. *software* yang digunakan selama penelitian

Komponen	Keterangan
OS	Windows 10 Home Single Language 64-bit
Compiler Bahasa C++	Microsoft Visual Studio 2015
Kamera USB	SJCam SJ4000 *Action Camera*
Pustaka OpenCV	Versi 3.1.0
Resolusi Video	640x480 pixel

3.3 Pengumpulan Sampel Positif dan Sampel Negatif

a. Sampel Positif

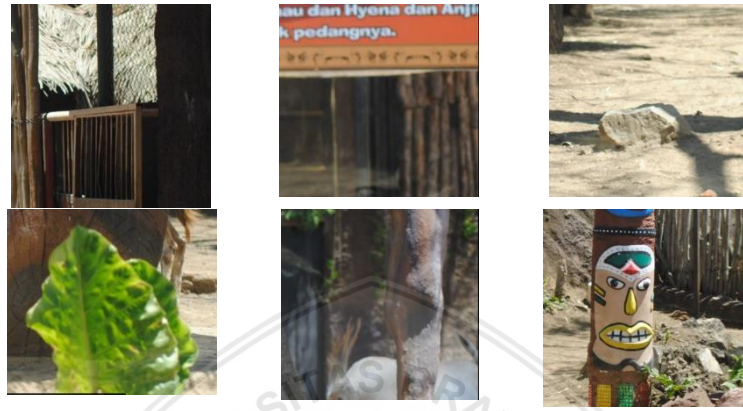
Sampel positif adalah citra yang didalamnya terdapat pola manusia yang sudah dilatih pada jaringan syaraf tiruan. Pada penelitian ini digunakan 90 sampel citra positif yang didapatkan dari google.com maupun di ambil dengan kamera DSLR secara langsung. Sampel positif yang dilatih merupakan tampak depan dari manusia dengan resolusi yang digunakan sebesar 80x60 pixel. Gambar 3.1 menunjukkan beberapa contoh citra positif yang digunakan untuk pelatihan pada jaringan syaraf tiruan.

Gambar 3.1. Contoh sampel positif yang digunakan.

b. Sampel Negatif

Sampel negatif merupakan citra yang memuat selain citra positif manusia. Sampel negatif diambil disekitar lokasi citra positif berada yang bertujuan agar jaringan syaraf tiruan dapat membedakan citra yang dilatih dengan baik. Penelitian menggunakan 60 sampel negatif yang diambil secara langsung di lapang dengan kamera DSLR. Sampel negatif yaitu diantaranya seperti citra jalan setapak,

pohon, dinding kandang, dan lain-lain dengan resolusi 80x60 pixel. Gambar 3.2 menunjukkan beberapa contoh citra positif yang digunakan untuk pelatihan pada jaringan syaraf tiruan



Gambar 3.2. Contoh sampel negatif yang digunakan.

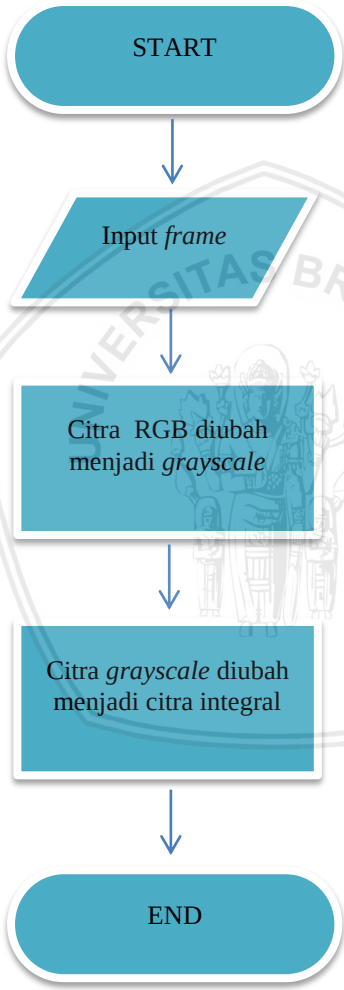
3.4 Pra-Pemrosesan Data Sampel

Penelitian menggunakan ekstraksi fitur-Haar. Pemrosesan awal citra RGB akan dikonversi menjadi citra *grayscale*. Hal ini diperuntukkan agar citra nantinya dapat diubah menjadi citra integral. Kondisi pencahayaan sangat mempengaruhi obyek yang akan dideteksi, dikarenakan pada penelitian kali ini, normalisasi histogram pada citra tidak dilakukan. Ketidakseimbangan tingkat keabuan dapat mempengaruhi terganggunya obyek yang akan dideteksi. Sehingga peletakkan kamera dan waktu pengambilan citra perlu diperhatikan agar obyek dapat dideteksi secara sempurna.

(a) (b)

Gambar 3.3. Tahap pra-pemrosesan: a. Citra RGB, b. Citra *grayscale*

Gambar 3.3 menunjukkan tahap pra-pemrosesan yang dilakukan pada penelitian. Citra di konversi dari citra RGB menjadi citra *grayscale* (keabu-abuan). Citra yang sudah diubah menjadi citra *grayscale* kemudian diubah kembali menjadi citra integral. Perubahan citra menjadi citra integral dibutuhkan agar proses komputasi dapat lebih mudah dan cepat. Adapun diagram alir tahap pra-pemrosesan ditunjukkan pada gambar 3.4



Gambar 3.4. Diagram alir tahap pra-pemrosesan

3.5 Ekstraksi Fitur Mirip-Haar

Ekstraksi fitur pada penelitian ini akan dilakukan dengan meletakkan fitur pada citra yang akan dijadikan citra pelatihan. Ekstraksi fitur-Haar akan menghitung intensitas pixel dari pola gelap terang fitur yang diletakkan tersebut. Adapun fitur pola gelap terang pada penelitian ini yaitu contohnya pola gelap terang antara mata dengan dahi, pola gelap terang antara warna rambut dengan latar belakang, pola gelap terang antara leher dengan baju yang digunakan, dan pola gelap terang antara sepatu yang digunakan dengan latar belakangnya. Sebanyak lima fitur digunakan untuk mencirikan citra manusia yang dimaksud. Lima fitur tersebut dianggap cukup untuk mencirikan manusia sebagai obyek yang akan dideteksi. Gambar 3.5. Merupakan gambar dari peletakkan fitur Haar yang digunakan pada citra.

Gambar 3.5. Peletakkan fitur-Haar yang digunakan pada peneltian

3.6 Jaringan Syaraf Tiruan

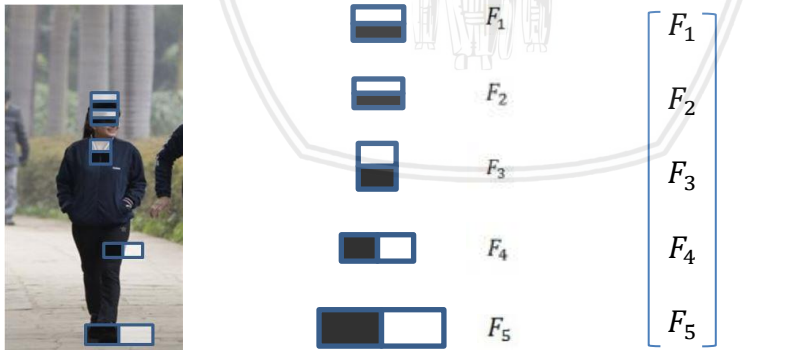
Penelitian menggunakan desain jaringan syaraf tiruan meliputi arsitektur jaringan syaraf tiruan, algoritma pelatihan yang digunakan selama pelatihan dan fungsi aktivasi dari masing-masing neuron yang ada pada jaringan syaraf tiruan.

a) Model Jaringan Syaraf Tiruan

Model dari jaringan syaraf tiruan yang digunakan pada penelitian ini yaitu jaringan syaraf tiruan dengan metode *feed-forward backpropagation (backpropagation neural network)*. *Backpropagation neural network* terdiri dari *input layer, hidden layer dan output layer*. Pembuatan hidden layer akan dibuat seminimal mungkin untuk meminimalisir kompleksitas jaringan dengan tetap memperhatikan kekuatan akurasi program.

b) *Input Layer*

Jaringan syaraf tiruan yang digunakan pada penelitian ini akan memiliki lima masukan. Kelima masukan tersebut berasal dari lima fitur yang digunakan selama pelatihan untuk pendeteksian obyek yang diteliti. Kelima fitur mirip-Haar yang digunakan tersebut akan dibentuk menjadi sebuah matriks vektor yang digunakan sebagai input jaringan syaraf tiruan. Matriks yang digunakan dapat dilihat pada gambar 3.6.



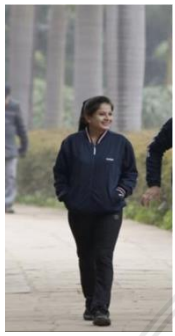
(a) Citra + fitur yang diletakkan

(b) Fitur yang digunakan akan dibentuk menjadi matriks

(c) Matriks yang digunakan sebagai *input*

Gambar 3.6. Lima fitur yang digunakan akan membentuk matriks

Penelitian menggunakan 150 sampel yang terdiri dari 90 sampel positif dan 60 sampel negatif. Sehingga pada masukan jaringan syaraf tiruan akan terbentuk matriks dengan ukuran 5x150. Selain 150 sampel yang digunakan sebagai masukan, akan dibentuk juga matriks target yang digunakan sebagai keluaran dari jaringan syaraf tiruan tersebut. Matriks target tersebut akan berukuran 1x150. Kedua hal tersebut dapat dilihat seperti pada gambar 3.7.



$$\begin{bmatrix} F_{1,1} & F_{1,2} & \cdots & F_{1,150} \\ F_{2,1} & F_{2,2} & \cdots & F_{2,150} \\ F_{3,1} & F_{3,2} & \cdots & F_{3,150} \\ F_{4,1} & F_{4,2} & \cdots & F_{4,150} \\ F_{5,1} & F_{5,2} & \cdots & F_{5,150} \end{bmatrix} \qquad \begin{bmatrix} T_{1,1} & T_{1,2} & \cdots & T_{1,150} \end{bmatrix}$$

(a) 150 citra sampel

(b) Matriks 5x150 sebagai *input*

(c) Matriks 1x150 sebagai target keluaran jaringan syaraf tiruan

Gambar 3.7. Matriks pada lapisan masukan dan keluaran untuk pelatihan jaringan syaraf tiruan

c) *Hidden Layer*

Tidak ada metode khusus yang digunakan untuk menentukan fungsi aktivasi, banyaknya neuron yang digunakan, jumlah hidden layer dan lain-lain pada jaringan syaraf tiruan. Jumlah neuron dan banyaknya jumlah hidden layer ditentukan berdasarkan metode hit and trial sampai didapatkan jumlah neuron dan hidden layer yang akurat dengan tidak mengesampingkan keefektifan program.

d) *Output Layer*

Keluaran yang digunakan pada penelitian hanya menggunakan satu keluaran. Yaitu bernilai 0 dan 1. Apabila keluaran citra hasil pelatihan akan bernilai mendekati 0, maka citra tersebut masuk ke dalam kategori citra negatif. Sedangkan jika citra hasil pelatihan

bernilai mendekati 1, maka citra tersebut masuk ke dalam citra positif.

3.7 Pelatihan Jaringan Syaraf Tiruan

Pelatihan jaringan syaraf tiruan dilakukan menggunakan *neural network toolbox* (nntool) yang ada pada *software* MATLAB R2015a. Setelah proses pelatihan jaringan syaraf tiruan telah dilakukan, maka nilai bias dan nilai bobot akan didapatkan. Selama proses pelatihan juga akan ditentukan jumlah neuron dari *hidden layer* seperti yang sudah disebutkan pada desain jaringan syaraf tiruan. Hasil yang didapat pada proses pelatihan akan digunakan untuk proses deteksi obyek manusia secara *real time*.

3.8 Pengujian Jaringan Syaraf Tiruan

Jaringan syaraf tiruan yang telah dibuat dan dilatih dengan menggunakan data latih akan kembali diuji tingkat keakuratan nya dengan menggunakan citra uji. Citra uji merupakan citra yang tidak pernah dikenalkan ke jaringan syaraf tiruan. Hal ini berguna untuk melihat sejauh mana jaringan syaraf tiruan yang telah dibuat dapat mengenal secara baik antara obyek negatif dan obyek positif.

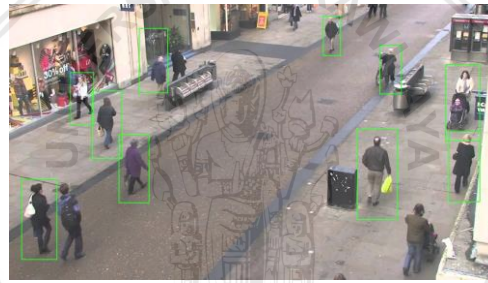
3.9 Aplikasi Secara *Real Time*

Jaringan syaraf tiruan yang sudah dilatih dan diuji tingkat akurasi nya menggunakan citra latih dan citra uji kemudian diimplementasikan secara *realtime*. Program yang dibuat untuk deteksi obyek pada penelitian menggunakan bahasa pemrograman C++ yang ditambahkan dengan pustaka OpenCV versi 3.1.0. Aplikasi secara *real time* mengikuti implementasi yang dilakukan oleh Pratama (2016). Adapun alur dari implementasi jaringan syaraf tiruan adalah sebagai berikut:

1. Pengenalan bobot dan bias yang didapatkan dari pelatihan jaringan syaraf tiruan.
2. Pengambilan *frame* dari kamera untuk *real time*
3. Pemisahan antara *foreground* dengan *background*
4. Pra-pemrosesan dilakukan pada frame yang telah diambil. Pra-pemrosesan yang dimaksud seperti yang sudah dijelaskan pada

3.2.3. Yaitu citra yang diambil akan diubah menjadi citra *grayscale* dan citra integral tanpa ada normalisasi histogram.

5. *Scanning* akan mulai dilakukan dengan menggeser *region* dari pixel ke pixel. *Region* tersebut berfungsi untuk mencari obyek manusia pada *frame* yang telah diambil. Ukuran *region* berubah sesuai dengan ukuran yang telah ditentukan.
6. *Region* akan diubah ke ukuran 80x60 pixel.
7. Pada tiap *region* kemudian dilakukan ekstraksi fitur mirip-Haar.
8. Nilai-nilai fitur-Haar yang didapat kemudian akan dimasukkan sebagai *input* untuk jaringan syaraf tiruan yang selanjutnya akan dipilih *region* yang memuat obyek manusia.
9. Setelah jaringan syaraf tiruan berhasil mengidentifikasi obyek manusia, selanjutnya terdapat *overlay* yang ditunjukkan pada citra asli dengan dibuat nya kotak pembatas pada obyek yang terdeteksi seperti pada gambar 3.8.



Gambar 3.8. Kotak pembatas pada obyek yang terdeteksi

10. Citra dengan hasil *overlay* selanjutnya ditampilkan.
11. Setelah selesai kembali ke langkah 2.

3.10 Metode Analisis

Analisis yang dilakukan pada penelitian ini menitikberatkan pada lokasi pengambilan video pada saat implementasi secara *realtime*. Pada implementasi secara *realtime*, video diambil di beberapa tempat di lingkungan kampus Universitas Brawijaya, yaitu gerbang Fakultas Peternakan, *lobby* MIPA Center dan tempat pejalan kaki antara gedung Jurusan Biologi dan Jurusan Fisika. Kondisi pencahayaan juga menjadi salah satu analisis pada penelitian ini, yaitu kondisi pencahayaan diluar ruang dan kondisi pencahayaan di

dalam ruang. Kondisi pencahayaan memiliki pengaruh yang besar untuk sistem mendeteksi antara obyek positif, negatif, dan positif salah. Analisis dari akurasi program mengikuti analisis yang dilakukan oleh Pratama (2016) yaitu meliputi jumlah terdeteksi obyek positif, negatif, dan positif salah. Serta kekuatan akurasi dalam besarnya persentase dari sistem yang telah dibuat dihitung dengan menjumlahkan obyek yang terdeteksi saat pengambilan video secara *realtime*.

Untuk menghitung akurasi pendeteksian dari sistem yang telah dibuat akan dilakukan dengan cara menghitung jumlah obyek positif, negatif, dan positif salah terhadap dua menit pengambilan data pada saat implementasi secara *realtime*. Akurasi program (*A*) secara keseluruhan dapat dihitung dengan persamaan berikut:

$$A = \frac{Obyek\ positif}{Obyek\ positif + obyek\ negatif + obyek\ positif\ salah}\ x\ 100\% \qquad (3.1)$$

Dimana selanjutnya obyek positif, obyek negatif, dan obyek positif salah masing-masing dapat disebut dengan *Op*, *On* dan *Ops*.

Untuk menghitung perbandingan antara obyek positif (*Op*) dengan obyek positif salah (*Ops*) yang selanjutnya disebut R_{riil} dapat dituliskan dengan persamaan:

$$R_{riil} = \frac{Op}{Op+Ops} \qquad (3.2)$$

Sedangkan untuk menghitung perbandingan antara obyek positif (*Op*) dengan obyek negatif (*On*), yang selanjutnya disebut dengan N_{riil} dapat dituliskan dengan persamaan:

$$N_{riil} = \frac{Op}{Op+On} \qquad (3.3)$$

Tabel hasil dari implementasi secara *realtime* dapat dilihat seperti pada contoh tabel 4.3

Tabel 3.3. Contoh tabel hasil implementasi secara *realtime*

Op	*On*	*Ops*	R_{riil}	N_{riil}	*A*
50	15	5	0.834	0.654	71,43%

BAB IV
HASIL DAN PEMBAHASAN

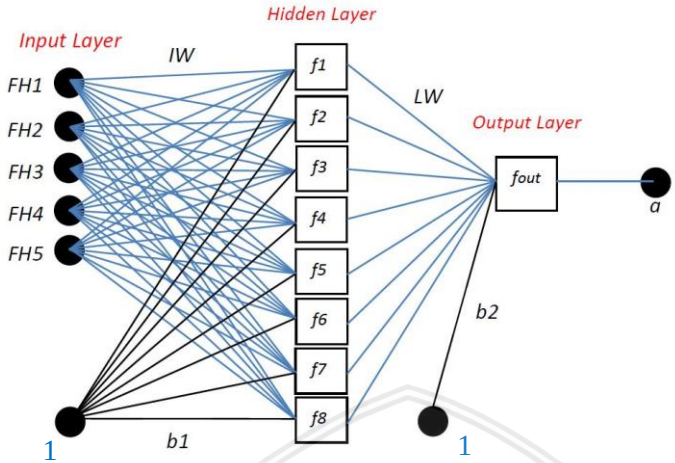
4.1 Jaringan Syaraf Tiruan

Klasifikasi obyek pada penelitian menggunakan jaringan syaraf tiruan yang telah dibuat kemudian dilatihkan menggunakan citra sampel dan diuji dengan citra uji. Jaringan syaraf tiruan yang dibuat akan menghasilkan keluaran 1 dan 0. Jika keluaran mendekati nilai 1, maka obyek akan diklasifikasikan sebagai citra positif. Sedangkan apabila keluaran dari jaringan syaraf tiruan mendekati nilai 0, maka obyek akan diklasifikasikan sebagai citra negatif. Nilai 0 dan 1 merupakan nilai yang telah ditentukan sebagai nilai target pada pelatihan jaringan syaraf tiruan.

4.1.1. Arsitektur Jaringan Syaraf Tiruan

Dari banyaknya jaringan syaraf tiruan yang telah dibuat, maka didapatkan satu buah jaringan syaraf tiruan yang dianggap mampu mengenalkan obyek manusia secara baik. Jaringan syaraf tiruan tersebut terdiri dari satu lapisan masukan (*input layer*), satu masukan tersembunyi (*hidden layer*), dan satu lapisan keluaran (*output layer*). Dimana terdapat sebanyak delapan neuron untuk lapisan tersembunyi (*hidden layer*) dan satu neuron untuk lapisan keluaran (*output layer*). Fungsi transfer yang digunakan pada *hidden layer* yaitu fungsi *log-sigmoid* (*logsig*).

Sedangkan pada *output layer* digunakan fungsi transfer *linear* (*purelin*). Permodelan dari jaringan syaraf tiruan yang dibuat dapat dilihat pada gambar 4.1. IW dan b1 masing-masing merupakan bobot dan bias yang menghubungkan antara *input layer* dengan *hidden layer*. Sedangkan LW dan b2 masing-masing merupakan bobot dan bias yang menghubungkan antara *hidden layer* dengan *output layer*. Nilai dari bobot dan bias terlampir.



Gambar 4.1. Arsitektur jaringan syaraf tiruan

4.1.2. Pengujian Jaringan Syaraf Tiruan

A. Pengujian dengan Citra Sampel

Citra sampel merupakan citra yang digunakan selama pelatihan jaringan syaraf tiruan. Citra sampel tersebut kemudian digunakan sebagai *input* untuk jaringan syaraf tiruan. Adapun keluaran yang didapat dari jaringan syaraf tiruan yang dibandingkan dengan target diperlihatkan pada tabel 4.1 dan 4.2.

Tabel 4.1. Hasil pengujian dengan citra sampel postif

Citra			
Target	1	1	1
Keluaran	0.8512	0.9821	0.7252

Tabel 4.2. Hasil pengujian dengan citra sampel negatif

Citra			
Target	0	0	0
Keluaran	0.3941	0.4038	0.2377

Setelah dilakukan pelatihan jaringan syaraf tiruan menggunakan 150 sampel postif dan negatif. Maka didapatkan akurasi sebesar 84% bahwa sistem dapat mengenali obyek manusia. Sehingga dari pelatihan menggunakan citra sampel tersebut dapat disimpulkan sistem dapat mengenali obyek manusia dengan cukup baik. Pada penelitian, penggunaan sampel positif memiliki jumlah yang lebih banyak agar pada saat implementasi secara *realtime* sistem dapat mengenal obyek positif lebih banyak dibandingkan dengan obyek negatif.

B. Pengujian dengan Citra Uji

Selanjutnya citra uji merupakan citra yang tidak pernah dikenalkan kepada jaringan syaraf tiruan. Pengujian dengan citra uji digunakan untuk melihat tingkat akurasi dari jaringan syaraf tiruan yang telah dibuat. Adapun hasil dari pengujian dengan citra uji diperlihatkan pada tabel 4.3 dan 4.4.

Tabel 4.3. Hasil pengujian dengan citra uji postif

Citra			
Target	1	1	1
Keluaran	0.5307	0.8245	0.7408

Tabel 4.4. Hasil pengujian dengan citra uji negatif

Citra			
Target	0	0	0
Keluaran	0.2114	0.2506	0.2751

Setelah dilakukan pengujian jaringan syaraf tiruan dengan citra uji sebanyak 190 citra yang terbagi menjadi 100 citra positif dan 90 citra negatif, didapatkan akurasi jaringan syaraf tiruan sebesar 86% dapat mengenali obyek manusia. Dari hasil keluaran tersebut, dapat disimpulkan jaringan syaraf tiruan dapat cukup baik membedakan citra positif dan citra negatif.

4.2 Klasifikasi Obyek

Pada aplikasi secara *real time*, kotak pembatas akan muncul sesuai dengan kemampuan prediksi dari jaringan syaraf tiruan. Pada penelitian, klasifikasi obyek dibagi menjadi tiga jenis. Yang pertama yaitu apabila kotak pembatas melingkupi obyek manusia, maka obyek tersebut adalah obyek positif. Yang kedua apabila tidak ada kotak pembatas yang melingkupi obyek manusia, maka obyek tersebut dapat diklasifikasikan sebagai obyek negatif. Yang ketiga apabila kotak pembatas melingkupi obyek non-manusia, maka obyek tersebut diklasifikasikan sebagai obyek positif salah.

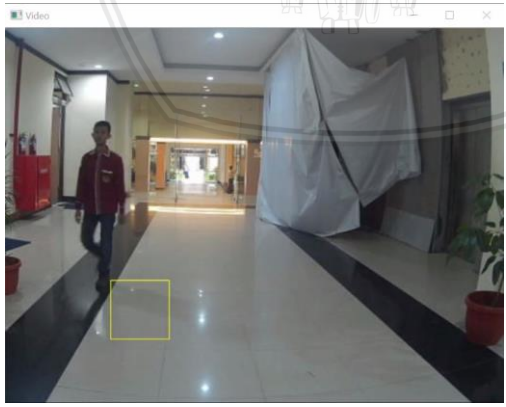
Ketiga jenis klasifikasi tersebut sesuai dengan pelatihan yang telah dilakukan oleh jaringan syaraf tiruan. Ketiga jenis klasifikasi tersebut masing-masing diperlihatkan pada gambar 4.2, gambar 4.3, dan gambar 4.4.

Gambar 4.2. Contoh obyek positif

Gambar 4.3. Contoh obyek negatif

Gambar 4.4. Contoh obyek positif salah

Penelitian yang dilakukan tidak menggunakan normalisasi histogram. Hal tersebut sangat memiliki dampak pada akurasi pendeteksian obyek secara *realtime* dengan adanya bayangan yang terdeteksi. Normalisasi histogram memiliki fungsi agar tingkat keabuan pada citra dapat tersebar rata setelah citra di konversi menjadi citra *grayscale*. Dengan adanya bayangan pada manusia, sistem memiliki kemungkinan untuk mendeteksi bayangan tersebut seperti yang dicontohkan pada gambar 4.5.



Gambar 4.5. Bayangan yang terdeteksi oleh sistem

4.3 Implementasi secara *realtime*

Implementasi pengambilan video secara *realtime* dilakukan dilingkungan kampus Universitas Brawijaya Malang. Tempat-tempat tersebut yaitu diantaranya gerbang masuk/keluar Fapet, lobby MIPA Center, dan tempat pejalan kaki di antara Gedung Jurusan Fisika dan Gedung Jurusan Biologi. Tempat-tempat tersebut dipilih karena memuat obyek manusia yang berlalu-lalang juga serta memiliki kompleksitas citra dari yang dilatihkan sebelumnya.

4.3.1 Lokasi pertama (Gerbang Fapet)

Lokasi implementasi secara *realtime* pertama, kamera diletakkan di Gerbang Fapet untuk mendeteksi obyek manusia. Kamera diletakkan tepat di persimpangan jalan diantara gerbang keluar/masuk dengan parkiran Fakultas Peternakan, sehingga pada saat pengambilan secara *realtime* dapat memuat obyek manusia tampak depan maupun tampak belakang. Waktu pengambilan di lokasi ini yaitu pada tanggal 11 Desember 2017 pukul 08.47 WIB. Gambar 4.6. Menunjukkan beberapa cuplikan pada saat implementasi nya secara *realtime*.

Gambar 4.6. Beberapa cuplikan pada lokasi pertama

Setelah dilakukan pengambilan data secara *realtime* selama tiga menit, maka dihasilkan sebagaimana ditunjukkan pada tabel 4.5. Jumlah obyek positif (*Op*), obyek negatif (*On*) dan obyek positif salah (*Ops*) dihitung secara manual.

Tabel 4.5. Data hasil pengambilan *realtime* di Gerbang Fapet

Op	*On*	*Ops*	R_{riil}	N_{riil}	*A*
24	7	2	0.923	0.774	72,72%

Hasil dari implementasi menunjukkan bahwa selama dua menit kesalahan terdapat pada sistem yang mendeteksi obyek negatif (*On*). Tetapi dari hal tersebut dapat disimpulkan bahwa sistem sudah cukup baik dalam memprediksi antara obyek manusia dan obyek bukan manusia. Tidak terdeteksi nya obyek manusia yang seharus nya terdeteksi disebabkan karena pada saat manusia lewat, obyek tidak tepat pada tampak depan atau belakang melainkan tampak sedikit menyamping, sehingga berpengaruh pada kemampuan proyeksi sistem. Sedangkan akurasi pendeteksian secara keseluruhan memiliki nilai sebesar 72,72%.

4.3.2 Lokasi kedua (perempatan Fisika – Biologi - FIB)

Lokasi implementasi secara *realtime* di lokasi kedua, kamera diletakkan di tempat pejalan kaki antara Gedung Jurusan Fisika, Gedung Jurusan Biologi, dan jalan menuju FIB. Sehingga sistem dapat mendeteksi obyek manusia pada tampak bagian depan dan tampak belakang. Waktu pengambilan di lokasi kedua yaitu pada tanggal 11 Desember 2017 pukul 11.44 WIB. Gambar 4.7. Menunjukkan beberapa cuplikan video di lokasi kedua.

Tabel 4.6. Data hasil pengambilan secara *realtime* antara perempatan Fisika-Biologi-FIB

Op	*On*	*Ops*	R_{riil}	N_{riil}	*A*
14	1	6	0.7	0.933	66,67%

Setelah dilakukan pengambilan secara *realtime* selama dua menit, maka didapatkan data yang diperlihatkan pada tabel 4.6. Jumlah dari obyek positif (*Op*), obyek negatif (*On*) dan obyek positif salah (*Ops*) dihitung secara manual.

Gambar 4.7. Beberapa cuplikan pada lokasi kedua

Hasil implementasi secara *realtime* menunjukkan sistem masih memiliki kekurangan dalam hal pendeteksian jika banyak obyek manusia yang tertangkap kamera yang ditunjukkan dengan meningkatnya obyek positif salah (*Ops*). Hal tersebut disebabkan oleh komputasi yang tidak terlalu baik, sehingga sistem terkadang berantakan dalam meletakkan lokasi kotak pembatas Secara keseluruhan, akurasi dari sistem yang telah dibuat pada lokasi kedua menurun menjadi 66,67%.

4.3.3 Lokasi ketiga (Lobby MIPA Center)

Lokasi implementasi secara *realtime* di lokasi ketiga, kamera diletakkan di depan lift lobby MIPA Center. Peletakkan kamera di tempat tersebut agar sistem dapat mendeteksi obyek manusia tampak depan maupun tampak belakang. Waktu pengambilan di lokasi ketiga yaitu pada tanggal 11 Desember 2017 pukul 11.32 WIB. Gambar 4.8.

Menunjukkan beberapa cuplikan pada saat pengambilan secara *realtime*.

Gambar 4.8. Beberapa cuplikan pada lokasi ketiga

Setelah dilakukan pengambilan video secara *realtime* selama dua menit, didapatkan data yang diperlihatkan pada tabel 4.7. Pada hasil pengambilan secara *realtime* sistem memilki nilai akurasi pendeteksian secara keseluruhan yaitu sebesar 63,63%. Terdapat peningkatan pada pendeteksian obyek negatif (*On*). Jumlah obyek negatif yang meningkat disebabkan oleh nilai dari fitur-Haar yang tidak terlalu signifikan, hal tersebut dikarenakan dengan obyek manusia yang menggunakan pakaian gelap, sehingga sistem tidak dapat mendeteksi obyek tersebut.

Tabel 4.7. Data pengambilan secara *realtime* di lobby MIPA Center

Op	*On*	*Ops*	R_{riil}	N_{riil}	*A*
21	6	6	0.78	0.78	63,63%

4.3.4 Perbandingan antar lokasi pengambilan

Setelah dilakukan pengambilan video secara *realtime* di tiga lokasi yang berbeda, kemudian ketiga lokasi tersebut dibandingkan untuk mengetahui kekuatan dan kelemahan akurasi di masing-masing

lokasi. Tabel 4.8. Menunjukkan hasil berdasarkan nilai R_{riil} dan nilai N_{riil} saat pengambilan secara *realtime* di ketiga lokasi

Tabel 4.8. Perbandingan dari ketiga lokasi

Ekstraksi Fitur Mirip-Haar dengan Jaringan Syaraf Tiruan Metode *Backpropagation*			
Lokasi ke-	R_{riil}	N_{riil}	Akurasi Keseluruhan (*A*)
1	0.923	0.774	72.72%
2	0.7	0.933	66.67%
3	0.78	0.78	63.63%

Dari hasil implementasi secara *realtime* yang ditunjukkan pada tabel 4.8., nilai akurasi paling tinggi ditunjukkan pada lokasi pertama (Gerbang Fapet) dan nilai akurasi paling rendah ditunjukkan di lokasi ketiga (Lobby MIPA Center). Nilai akurasi secara keseluruhan dapat dipengaruhi oleh banyak atau tidaknya obyek positif, obyek positif salah, dan obyek negatif yang terdeteksi.

(Halaman ini sengaja dikosongkan)

BAB V
PENUTUP

5.1 Kesimpulan

Jaringan syaraf tiruan yang dibuat pada penelitian terdiri dari 8 neuron pada *hidden layer* menggunakan fungsi *log-sigmoid* (*logsig*) dan 1 neuron pada *output layer* menggunakan *linear transfer function* (*purelin*), dimana 5 nilai fitur haar digunakan sebagai *input* untuk jaringan syaraf tiruan.

Pada implementasi secara *realtime*, sistem mendapatkan nilai R_{riil} pada lokasi pertama, kedua, dan ketiga masing-masing sebesar 0.923, 0.7, dan 0.78. Sedangkan nilai N_{riil} yang didapatkan pada lokasi pertama, kedua, dan ketiga masing-masing sebesar 0.774, 0.933, dan 0.78. Sedangkan akurasi sistem secara keseluruhan yang didapat yaitu sebesar 72.72% pada lokasi pertama, 66.67% pada lokasi kedua, dan 63.63% pada lokasi ketiga. Kekurangan dari sistem yang telah dibuat yaitu masih kurangnya kecepatan komputasi dalam mendeteksi suatu obyek yang secara tidak langsung berpengaruh pada jumlah obyek positif (*Op*), obyek negatif (*On*), dan obyek positif salah (*Ops*). Pakaian gelap yang digunakan manusia juga menjadi kendala untuk sistem dalam memproyeksikan obyek.

5.2 Saran

Pada penelitian ini akurasi sistem secara keseluruhan masih memiliki kemungkinan untuk ditingkatkan dengan menggunakan *pre-processing* normalisasi histogram. Proses normalisasi histogram bekerja dengan cara meratakan tingkat keabuan pada citra digital setelah citra dikonversi menjadi *grayscale*. Dengan cara tersebut, sistem memiliki kemungkinan dapat mendeteksi obyek positif lebih banyak dan mengurangi jumlah terdeteksinya obyek positif salah. Penambahan metode selain ekstraksi fitur mirip-Haar juga dapat digunakan untuk menyempurnakan tingkat akurasi menjadi lebih baik lagi.

Pada saat implementasi secara *realtime*, penggunaan *single-board computer* (SBC) bisa menjadi salah satu pertimbangan pada pengembangan nya. SBC dapat digunakan untuk memudahkan penggunaan sehingga tidak perlu lagi membawa PC saat

implementasi nya secara *realtime*. Pengembangan sistem kearah yang lebih aplikatif dapat difokuskan untuk penelitian selanjutnya. Sistem yang telah dibuat dapat digunakan untuk menghitung jumlah manusia yang masuk ke suatu tempat misalnya kebun binatang, bandara, dan tempat lainnya.

DAFTAR PUSTAKA

Amit, Y., dan Felzenszwalb, P. (2013): Object Detection, *Encyclopedia of computer vision*.

Anderson Sihombing, F. (2008): *Studi pemanfaatan pencahayaan alami pada beberapa rancangan ruang kelas perguruan tinggi di medan*, Universitas Sumatera Utara Medan.

Christos S., dan Dimitros S, 1996. *Neural Networks* . Diambil dari: https://www.doc.ic.ac.uk/~nd/surprise_96/journal/vol4/cs11/report.html#Network layers (3 Agustus 2017).

Dewi, L. J. E. (2010): Media Pembelajaran Bahasa Pemrograman C++, *UNDIKSHA*, *Vol. 7*, *No*, 63–72.

Emami, S., dan Suciu, V. P. (2012): Facial Recognition using OpenCV, *Journal of Mobile, Embedded and Distributed Systems*, *4*(1), 38–43, diambil dari http://www.jmeds.eu/index.php/jmeds/article/view/57.

Emeritus, R. A. S. (2004): *Physics for Scientist and Engineers 6th Edition*, New York: Thomson Brooks/Cole.

Firdausy, K. (2013): Penjejak Pose Wajah Otomatis pada Sistem Pengenalan Wajah, *Vol. 35*, *N*, 33–41.

Gazali, W., Soeparno, H., dan Ohliati, J. (2012): Penerapan Metode Konvolusi Dalam Pengolahan Citra Digital, *School of Computer Science Binus University*, *Vol 12*, 103–113.

Heriyanto, Abdul Kadir. 2006. *Algoritma Pemrograman Menggunakan C++*. Yogyakarta: Andi.

Keiser, G. (2016): *Biophotonics*, Singapore: Springer Science + Business Media, https://doi.org/10.1007/978-981-10-0945-7.

Lesnussa, Y. A., Latuconsina, S., dan Persulessy, E. R. (2015): Aplikasi Jaringan Saraf Tiruan Backpropagation untuk Memprediksi Prestasi Siswa SMA (Studi kasus : Prediksi Prestasi Siswa SMAN 4 Ambon), *Jurnal Matematika Integratif*,

11(2), 149–160.

Maheswari, S. U. M. A. (2015): The Weight Adjustment of Haar-Like Features in Viola Jones Classifier Using Principal Component Analysis Method for Hand Posture Recognition, *International Journal of Advances in Electronics and Computer Science*, *Volume-2*(12), 78–82.

Muhtadan, dan Harsono, D. (2008): Pengembangan Aplikasi Untuk Perbaikan Citra Digital Film Radiografi, *Seminar Nasional IV SDM Teknologi Nuklir*, 467–478, Yogyakarta: Sekolah Tinggi Teknologi Nuklir - BATAN.

Mulyawan, H., Samsono, M. Z. H., dan Setiawardhana (2011): Identifikasi dan Tracking Obyek Berbasis Image Processing Secara Real Time, 1–5, diambil dari http://repo.pens.ac.id/1324/1/Paper_TA_MBAH.pdf.

Mustafa, R., Min, Y., dan Zhu, D. (2014): Obscenity detection using haar-like features and gentle Adaboost classifier, *The Scientific World Journal*, *2014*, https://doi.org/10.1155/2014/753860.

Pratama, B., 2016. *Rancang Bangun Sistem Pendeteksian Real-Time Obyek Mobil Berbasis Raspberry Pi 3 Dengan Metode Ekstraksi Fitur Mirip-Haar Dan Jaringan Syaraf Tiruan*. Jurusan Fisika. Malang: Universitas Brawijaya.

Pradana, A., Wicaksono, A., Rusdinar, A., Suratman, F. Y., Elektro, F. T., Telkom, U., Elektro, F. T., Telkom, U., Elektro, F. T., dan Telkom, U. (2016): *Perancangan Pelacak Obyek Menggunakan Pengolahan Citra Secara Real Time Dengan Metode Sum Area Table*, Universitas Telkom.

Schneiderman, H., dan Kanade, T. (2004): Object detection using the statistics of parts, *Int J Comput Vision*, *56*(3), 151–177.

Siang, J. J. (2005): *Jaringan Syaraf Tiruan dan Pemrogramannya menggunakan Matlab*, Yogyakarta: Penerbit Andi.

Skorin-Kapov, J., dan Tang, K. (2001): Training Artificial Neural Networks: Backpropagation via Nonlinear Optimization, *Journal of Computing and Information Technology*, *9*(1), 1–14, https://doi.org/10.2498/cit.2001.01.01.

Werbos, P. J. (1990): Backpropagation throught time: What it does and how to do it, *Proceedings of the IEEE*, **78**, 1550–1560.

(Halaman ini sengaja dikosongkan)

LAMPIRAN A
NILAI BOBOT DAN BIAS JARINGAN

IW =

-0.0417	0.6694	-3.0461	-2.3255	-1.5139
1.0478	-2.0829	0.0155	-3.3826	1.6382
-0.8790	-1.6057	1.2918	-3.1030	1.7570
-2.8188	2.4869	2.8907	-2.7834	0.4966
-4.3848	1.2831	0.5323	-1.9494	-1.5811
1.1742	-2.1281	-2.8586	-1.0458	-1.8813
-1.2338	-3.7377	0.7028	-0.6557	-1.2216
-0.4844	-3.2863	-1.4316	-0.8670	2.0019

b1 =

4.4374
-3.1480
1.8833
0.3552
0.4560
1.3619
-3.3948
-4.3348

LW =

0.5395	1.9174	0.1564	-1.6411	2.0886	-1.4614	-1.5729	-1.6212

b2 = 0.3710

(Halaman ini sengaja dikosongkan)

LAMPIRAN B

KODE PROGRAM

B.1 Kode Program Ekstraksi Fitur Haar

```
sample = 90;
for i = 1:sample
s1 = int2str(57);
s2 = '.jpg';
s = strcat(s1, s2);
image = imread(s);
INT = integralImage(image);
A(1, i) = (abs((INT(15, 51) + INT(13, 38) - INT(13, 51) -
INT(15, 38)) - (INT(17, 51) + INT(15, 38) - INT(15, 51) -
INT(17, 38))));
A(2, i) = (abs((INT(18, 52) + INT(17, 31) - INT(17, 52) -
INT(18, 37)) - (INT(19, 52) + INT(18, 37) - INT(18, 52) -
INT(19, 37))));
A(3, i) = (abs((INT(26, 47) + INT(23, 39) - INT(23, 47) -
INT(26, 39)) - (INT(29, 47) + INT(26, 39) - INT(26, 47) -
INT(29, 39))));
A(4, i) = (abs((INT(46, 62) + INT(42, 53) - INT(42, 62) -
INT(46, 53)) - (INT(46, 53) + INT(42, 41) - INT(42, 53) -
INT(46, 41))));
A(5, i) = (abs((INT(59, 59) + INT(56, 51) - INT(56, 59) -
INT(59, 51)) - (INT(59, 51) + INT(56, 37) - INT(56, 51) -
INT(59, 37))));
end
```

B.2 Kode Program Utama

```
#include <opencv2/core/core.hpp>
#include <opencv2/imgproc/imgproc.hpp>
#include <opencv2/highgui/highgui.hpp>
#include <opencv2/imgcodecs.hpp>
#include <opencv2/objdetect/objdetect.hpp>
#include <opencv2/videoio.hpp>
#include <opencv2/video.hpp>
#include <opencv2/video/background_segm.hpp>
#include <iostream>
#include <cmath>
#include <ctime>
#include <vector>

using namespace cv;
using namespace std;

int main(int argc, const char** argv)
{
        VideoCapture cap(1);//capture video dari kamera

        if (!cap.isOpened()) //tutup program jika video tidak ada
        {
                cout << "Tidak bisa membuka file video" << endl;
                return -1;
        }

        namedWindow("Gray", CV_WINDOW_NORMAL | CV_WINDOW_KEEPRATIO);
        namedWindow("Foreground", CV_WINDOW_NORMAL |
CV_WINDOW_KEEPRATIO);
        namedWindow("Integral", CV_WINDOW_NORMAL |
CV_WINDOW_KEEPRATIO);
        namedWindow("Video", CV_WINDOW_AUTOSIZE);

        Mat frame, frame_big, fgmask_bw, fgmask;
        int history, nonzero;
        double varThreshold;
        bool detectShadows;
        Ptr<BackgroundSubtractorMOG2> hapusbg;
        hapusbg = createBackgroundSubtractorMOG2(history = 600,
varThreshold = 88, detectShadows = false);

        while (1)
        {

int start_s = clock();
                Mat gray_frame, eq, intimg;
                bool bSuccess = cap.read(frame);
                //frame_big = frame(Rect(0, 0, 640, 480));
                frame = frame(Rect(160, 120, 480, 360));
```

```
                resize(frame, frame_big, Size(640, 480), 0, 0,
INTER_CUBIC);
                hapusbg->apply(frame_big, fgmask_bw);
                fgmask = Mat::zeros(frame_big.rows, frame_big.cols,
frame_big.type());
                frame_big.copyTo(fgmask, fgmask_bw);

                if (!bSuccess) //jika gagal maka stop
                {
                        cout << "Tidak dapat membaca frame" << endl;
                        break;
                }

cvtColor(fgmask, gray_frame, CV_BGR2GRAY);
                equalizeHist(gray_frame, gray_frame);
                integral(gray_frame, intimg);//rubah ke citra
integral
                int kolom = frame_big.cols - 150;
                int baris = frame_big.rows;
                int x, y, lebar, tinggi;
                Mat cropped;
                int x1, y1, f1, f2, f3, f4, f5;
                int n = 0;
                int jumlah = 0;
                double out, a1, a2, a3, a4, a5, a6, a7, a8;
                double IW[8][5] = { { -0.041657003709401,
0.669379985827259,      -3.046148253958870,     -2.325547630790341,
        -1.513888189708991 },
                { 1.047751390635722,    -2.082890479843006,
        0.015469167455263,      -3.382568810301986,
        1.638153199497080 },
                { -0.878956377019082,   -1.605739254162023,
        1.291845220632303,      -3.102999304222826,
        1.757036317742411 },
                { -2.818778694350848,   2.486903591530209,
        2.890748258895996,      -2.783361423123070,
        0.496608807262726 },
                { -4.384791628326817,   1.283067120885951,
        0.532250619768678,      -1.949395362672776,     -
1.581126949812953 },
                { 1.174178435188848,    -2.128122275545430,     -
2.858584342942192,      -1.045789490831312,     -1.881340971555092 },
                { -1.233823193170321,   -3.737695544339271,
        0.702832766401487,      -0.655719186413463,     -
1.221570999740149 },
                { -0.484426211748617,   -3.286320189565068,     -
1.431572643705840,      -0.867005774168767,     2.001876071087874 },
                };
                double b1[8] = { 4.437388037077420, -
3.148012565422142, 1.883257684279961, 0.355176045827713,
```

```
0.455958671303398, 1.361925251097279, -3.394782055339483, -
4.334771231745932 };
                double LW[8] = { 0.539510028377341,
        1.917434314226752,      0.156366964153064, -
1.641103589399699,      2.088604422985829,      -1.461395300360407,
        -1.572865860098547,     -1.621249056595643 };
                double b2 = 0.371016470160627;
                vector<Rect> boundRect0(100);

for (int i = 120; i < 340;)
//scanning 80x60
                {
                        for (int j = 0; j < kolom;)
                        {
                                lebar = 80;
                                tinggi = 60;
                                x = 0 + j;
                                y = 0 + i;
                                cropped = gray_frame(Rect(x, y,
lebar, tinggi));
                                nonzero = countNonZero(cropped);
                                if (nonzero>2000)
                                {
                                        cropped = intimg(Rect(x, y,
lebar, tinggi));
                                        f1 =
abs(((cropped.at<int>(13, 49)) + (cropped.at<int>(11, 36)) -
(cropped.at<int>(11, 49)) - (cropped.at<int>(13, 36))) -
((cropped.at<int>(15, 49)) + (cropped.at<int>(13, 36)) -
(cropped.at<int>(13, 49)) - (cropped.at<int>(15, 36))));
                                        f2 =
abs(((cropped.at<int>(16, 50)) + (cropped.at<int>(15, 35)) -
(cropped.at<int>(15, 50)) - (cropped.at<int>(16, 35))) -
((cropped.at<int>(17, 50)) + (cropped.at<int>(16, 35)) -
(cropped.at<int>(16, 50)) - (cropped.at<int>(17, 35))));
                                        f3 =
abs(((cropped.at<int>(24, 45)) + (cropped.at<int>(21, 37)) -
(cropped.at<int>(21, 45)) - (cropped.at<int>(24, 37))) -
((cropped.at<int>(27, 45)) + (cropped.at<int>(24, 37)) -
(cropped.at<int>(24, 45)) - (cropped.at<int>(27, 37))));
                                        f4 =
abs(((cropped.at<int>(44, 60)) + (cropped.at<int>(40, 51)) -
(cropped.at<int>(40, 60)) - (cropped.at<int>(44, 51))) -
((cropped.at<int>(44, 51)) + (cropped.at<int>(40, 39)) -
(cropped.at<int>(40, 51)) - (cropped.at<int>(44, 39))));
                                        f5 =
abs(((cropped.at<int>(57, 57)) + (cropped.at<int>(54, 49)) -
(cropped.at<int>(54, 57)) - (cropped.at<int>(57, 49))) -
((cropped.at<int>(57, 49)) + (cropped.at<int>(54, 35)) -
(cropped.at<int>(54, 49)) - (cropped.at<int>(57, 35))));
```

```
                                        a1 = 1 / (1 + exp(-1 *
(IW[0][0] * f1 + IW[0][1] * f2 + IW[0][2] * f3 + IW[0][3] * f4 +
IW[0][4] * f5 + b1[0])));
                                        a2 = 1 / (1 + exp(-1 *
(IW[1][0] * f1 + IW[1][1] * f2 + IW[1][2] * f3 + IW[1][3] * f4 +
IW[1][4] * f5 + b1[1])));
                                        a3 = 1 / (1 + exp(-1 *
(IW[2][0] * f1 + IW[2][1] * f2 + IW[2][2] * f3 + IW[2][3] * f4 +
IW[2][4] * f5 + b1[2])));
                                        a4 = 1 / (1 + exp(-1 *
(IW[3][0] * f1 + IW[3][1] * f2 + IW[3][2] * f3 + IW[3][3] * f4 +
IW[3][4] * f5 + b1[3])));
                                        a5 = 1 / (1 + exp(-1 *
(IW[4][0] * f1 + IW[4][1] * f2 + IW[4][2] * f3 + IW[4][3] * f4 +
IW[4][4] * f5 + b1[4])));
                                        a6 = 1 / (1 + exp(-1 *
(IW[5][0] * f1 + IW[5][1] * f2 + IW[5][2] * f3 + IW[5][3] * f4 +
IW[5][4] * f5 + b1[5])));
                                        a7 = 1 / (1 + exp(-1 *
(IW[6][0] * f1 + IW[6][1] * f2 + IW[6][2] * f3 + IW[6][3] * f4 +
IW[6][4] * f5 + b1[6])));
                                        a8 = 1 / (1 + exp(-1 *
(IW[7][0] * f1 + IW[7][1] * f2 + IW[7][2] * f3 + IW[7][3] * f4 +
IW[7][4] * f5 + b1[7])));

                                        out = LW[0] * a1 + LW[1] * a2
+ LW[2] * a3 + LW[3] * a4 + LW[4] * a5 + LW[5] * a6 + LW[6] * a7 +
LW[7] * a8 + b2;
                                        if (out > 1.5)
                                        {
                                                x1 = x;
                                                y1 = y;
                                                boundRect0[n] =
Rect(x1, y1, 60, 60);
                                                j = j + 40;
                                                n = n + 1;
                                        }
                                        else
                                        {
                                                j = j + 10;
                                        }
                                }
                                else
                                {
                                        j = j + 10;
                                }
                        }
                        i = i + 5;
                }

kolom = frame_big.cols - 121;
                baris = frame_big.rows - 101;
```

```
                for (int i = 250; i < 340;)
//scanning 120x90
                {
                        for (int j = 10; j < kolom;)
                        {
                                lebar = 120;
                                tinggi = 90;
                                x = 0 + j;
                                y = 0 + i;
                                cropped = gray_frame(Rect(x, y,
lebar, tinggi));
                                nonzero = countNonZero(cropped);
                                if (nonzero>2000)
                                {
                                        cropped = intimg(Rect(x, y,
lebar, tinggi));
                                        resize(cropped, cropped,
Size(80, 60), 0, 0, INTER_NEAREST);
                                        f1 =
abs(((cropped.at<int>(13, 49)) + (cropped.at<int>(11, 36)) -
(cropped.at<int>(11, 49)) - (cropped.at<int>(13, 36))) -
((cropped.at<int>(15, 49)) + (cropped.at<int>(13, 36)) -
(cropped.at<int>(13, 49)) - (cropped.at<int>(15, 36))));
                                        f2 =
abs(((cropped.at<int>(16, 50)) + (cropped.at<int>(15, 35)) -
(cropped.at<int>(15, 50)) - (cropped.at<int>(16, 35))) -
((cropped.at<int>(17, 50)) + (cropped.at<int>(16, 35)) -
(cropped.at<int>(16, 50)) - (cropped.at<int>(17, 35))));
                                        f3 =
abs(((cropped.at<int>(24, 45)) + (cropped.at<int>(21, 37)) -
(cropped.at<int>(21, 45)) - (cropped.at<int>(24, 37))) -
((cropped.at<int>(27, 45)) + (cropped.at<int>(24, 37)) -
(cropped.at<int>(24, 45)) - (cropped.at<int>(27, 37))));
                                        f4 =
abs(((cropped.at<int>(44, 60)) + (cropped.at<int>(40, 51)) -
(cropped.at<int>(40, 60)) - (cropped.at<int>(44, 51))) -
((cropped.at<int>(44, 51)) + (cropped.at<int>(40, 39)) -
(cropped.at<int>(40, 51)) - (cropped.at<int>(44, 39))));
                                        f5 =
abs(((cropped.at<int>(57, 57)) + (cropped.at<int>(54, 49)) -
(cropped.at<int>(54, 57)) - (cropped.at<int>(57, 49))) -
((cropped.at<int>(57, 49)) + (cropped.at<int>(54, 35)) -
(cropped.at<int>(54, 49)) - (cropped.at<int>(57, 35))));
                                        a1 = 1 / (1 + exp(-1 *
(IW[0][0] * f1 + IW[0][1] * f2 + IW[0][2] * f3 + IW[0][3] * f4 +
IW[0][4] * f5 + b1[0])));
                                        a2 = 1 / (1 + exp(-1 *
(IW[1][0] * f1 + IW[1][1] * f2 + IW[1][2] * f3 + IW[1][3] * f4 +
IW[1][4] * f5 + b1[1])));
                                        a3 = 1 / (1 + exp(-1 *
(IW[2][0] * f1 + IW[2][1] * f2 + IW[2][2] * f3 + IW[2][3] * f4 +
IW[2][4] * f5 + b1[2])));
```

```
                                        a4 = 1 / (1 + exp(-1 *
(IW[3][0] * f1 + IW[3][1] * f2 + IW[3][2] * f3 + IW[3][3] * f4 +
IW[3][4] * f5 + b1[3])));
                                        a5 = 1 / (1 + exp(-1 *
(IW[4][0] * f1 + IW[4][1] * f2 + IW[4][2] * f3 + IW[4][3] * f4 +
IW[4][4] * f5 + b1[4])));
                                        a6 = 1 / (1 + exp(-1 *
(IW[5][0] * f1 + IW[5][1] * f2 + IW[5][2] * f3 + IW[5][3] * f4 +
IW[5][4] * f5 + b1[5])));
                                        a7 = 1 / (1 + exp(-1 *
(IW[6][0] * f1 + IW[6][1] * f2 + IW[6][2] * f3 + IW[6][3] * f4 +
IW[6][4] * f5 + b1[6])));
                                        a8 = 1 / (1 + exp(-1 *
(IW[7][0] * f1 + IW[7][1] * f2 + IW[7][2] * f3 + IW[7][3] * f4 +
IW[7][4] * f5 + b1[7])));
                                        out = LW[0] * a1 + LW[1] * a2
+ LW[2] * a3 + LW[3] * a4 + LW[4] * a5 + LW[5] * a6 + LW[6] * a7 +
LW[7] * a8 + b2;
                                        if (out > 1.5)
                                        {
                                                x1 = x;
                                                y1 = y;
                                                boundRect0[n] =
Rect(x1, y1, 90, 90);
                                                j = j + 45;
                                                n = n + 1;
                                        }
                                        else
                                        {
                                                j = j + 10;
                                        }
                                }
                                else
                                {
                                        j = j + 10;
                                }
                        }
                        i = i + 10;
                }


kolom = frame_big.cols - 161;
                baris = frame_big.rows - 121;
                for (int i = 300; i < baris;)
//scanning 160x120
                {
                        for (int j = 10; j < kolom;)
                        {
                                lebar = 160;
                                tinggi = 120;
```

```
                                x = 0 + j;
                                y = 0 + i;
                                cropped = gray_frame(Rect(x, y,
lebar, tinggi));
                                nonzero = countNonZero(cropped);
                                if (nonzero>8000)
                                {
                                        cropped = intimg(Rect(x, y,
lebar, tinggi));
                                        resize(cropped, cropped,
Size(80, 60), 0, 0, INTER_NEAREST);
                                        f1 =
abs(((cropped.at<int>(13, 49)) + (cropped.at<int>(11, 36)) -
(cropped.at<int>(11, 49)) - (cropped.at<int>(13, 36))) -
((cropped.at<int>(15, 49)) + (cropped.at<int>(13, 36)) -
(cropped.at<int>(13, 49)) - (cropped.at<int>(15, 36))));
                                        f2 =
abs(((cropped.at<int>(16, 50)) + (cropped.at<int>(15, 35)) -
(cropped.at<int>(15, 50)) - (cropped.at<int>(16, 35))) -
((cropped.at<int>(17, 50)) + (cropped.at<int>(16, 35)) -
(cropped.at<int>(16, 50)) - (cropped.at<int>(17, 35))));
                                        f3 =
abs(((cropped.at<int>(24, 45)) + (cropped.at<int>(21, 37)) -
(cropped.at<int>(21, 45)) - (cropped.at<int>(24, 37))) -
((cropped.at<int>(27, 45)) + (cropped.at<int>(24, 37)) -
(cropped.at<int>(24, 45)) - (cropped.at<int>(27, 37))));
                                        f4 =
abs(((cropped.at<int>(44, 60)) + (cropped.at<int>(40, 51)) -
(cropped.at<int>(40, 60)) - (cropped.at<int>(44, 51))) -
((cropped.at<int>(44, 51)) + (cropped.at<int>(40, 39)) -
(cropped.at<int>(40, 51)) - (cropped.at<int>(44, 39))));
                                        f5 =
abs(((cropped.at<int>(57, 57)) + (cropped.at<int>(54, 49)) -
(cropped.at<int>(54, 57)) - (cropped.at<int>(57, 49))) -
((cropped.at<int>(57, 49)) + (cropped.at<int>(54, 35)) -
(cropped.at<int>(54, 49)) - (cropped.at<int>(57, 35))));
                                        a1 = 1 / (1 + exp(-1 *
(IW[0][0] * f1 + IW[0][1] * f2 + IW[0][2] * f3 + IW[0][3] * f4 +
IW[0][4] * f5 + b1[0])));
                                        a2 = 1 / (1 + exp(-1 *
(IW[1][0] * f1 + IW[1][1] * f2 + IW[1][2] * f3 + IW[1][3] * f4 +
IW[1][4] * f5 + b1[1])));
                                        a3 = 1 / (1 + exp(-1 *
(IW[2][0] * f1 + IW[2][1] * f2 + IW[2][2] * f3 + IW[2][3] * f4 +
IW[2][4] * f5 + b1[2])));
                                        a4 = 1 / (1 + exp(-1 *
(IW[3][0] * f1 + IW[3][1] * f2 + IW[3][2] * f3 + IW[3][3] * f4 +
IW[3][4] * f5 + b1[3])));
                                        a5 = 1 / (1 + exp(-1 *
(IW[4][0] * f1 + IW[4][1] * f2 + IW[4][2] * f3 + IW[4][3] * f4 +
IW[4][4] * f5 + b1[4])));
```

```
                                        a6 = 1 / (1 + exp(-1 *
(IW[5][0] * f1 + IW[5][1] * f2 + IW[5][2] * f3 + IW[5][3] * f4 +
IW[5][4] * f5 + b1[5])));
                                        a7 = 1 / (1 + exp(-1 *
(IW[6][0] * f1 + IW[6][1] * f2 + IW[6][2] * f3 + IW[6][3] * f4 +
IW[6][4] * f5 + b1[6])));
                                        a8 = 1 / (1 + exp(-1 *
(IW[7][0] * f1 + IW[7][1] * f2 + IW[7][2] * f3 + IW[7][3] * f4 +
IW[7][4] * f5 + b1[7])));
                                        out = LW[0] * a1 + LW[1] * a2
+ LW[2] * a3 + LW[3] * a4 + LW[4] * a5 + LW[5] * a6 + LW[6] * a7 +
LW[7] * a8 + b2;
                                        if (out > 1.5)
                                        {
                                                x1 = x;
                                                y1 = y;
                                                boundRect0[n] =
Rect(x1, y1, 120, 120);
                                                j = j + 60;
                                                n = n + 1;
                                        }
                                        else
                                        {
                                                j = j + 10;
                                        }
                                }
                                else
                                {
                                        j = j + 10;
                                }
                        }
                        i = i + 10;
                }


groupRectangles(boundRect0, 1, 0.7);
                for (int h = 0; h < boundRect0.size(); h++)
                {
                        rectangle(frame_big, boundRect0[h].tl(),
boundRect0[h].br(), Scalar(0, 255, 255));
                        jumlah = jumlah + 1;
                }

                int stop_s = clock();
                cout << "Waktu: " << (stop_s - start_s) /
double(CLOCKS_PER_SEC) * 1000 << " ms" << endl;
                cout << "FPS:" << double(1000 / ((stop_s - start_s) /
double(CLOCKS_PER_SEC) * 1000)) << endl;
                cout << "Jumlah Terdeteksi:" << jumlah - 1 << endl;
                cout << "----------------------------" << endl;
```

```
                imshow("Video", frame_big);
                imshow("Gray", gray_frame);
                imshow("Foreground", fgmask);
                imshow("Integral", intimg);

                if (waitKey(30) == 27) //wait for 'esc' key press for
30 ms. If 'esc' key is pressed, break loop
                {
                        break;
                }
        }
        return 0;
}
```