BAB IV
HASIL DAN PEMBAHASAN

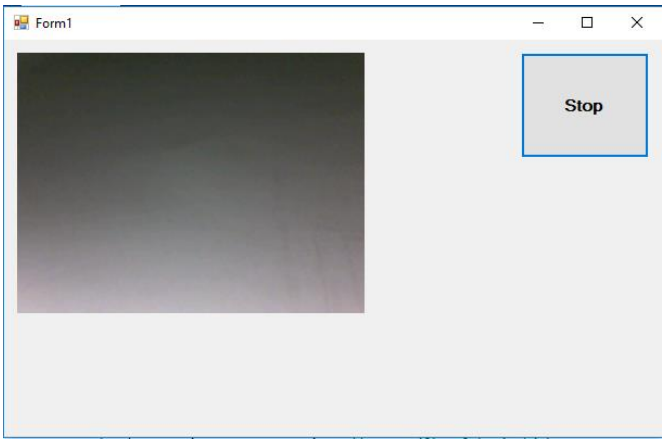
4.1 Pembahasan

4.1.1 Microsoft Visual Studio

Microsoft Visual Studio merupakan sebuah perangkat lunak lengkap (*suite*) yang dapat digunakan untuk melakukan pengembangan aplikasi, baik itu aplikasi bisnis, aplikasi personal, ataupun komponen aplikasinya, dalam bentuk aplikasi console, aplikasi *Windows*, ataupun aplikasi *Web*. Visual Studio mencakup kompiler, SDK, *Integrated Development Environment (IDE)*, dan dokumentasi (umumnya berupa MSDN *Library*). Kompiler yang dimasukkan ke dalam paket Visual Studio antara lain Visual C++, Visual C#, Visual Basic, Visual Basic .NET, Visual InterDev,Visual J++, Visual J#, Visual FoxPro, dan Visual *SourceSafe*. Saat ini Microsoft Visual Studio terbaru adalah versi 2015, versi ini hanya bisa berjalan di windows 2010.

5 Antar Muka

GUI (*Graphical User Interface*) merupakan antar muka pengguna suatu program berbasis grafis, yakni perintah-perintah tidak diketik melalui *keyboard*, berikut adalah beberapa tampilan antar pengguna untuk berinteraksi dengan Aplikasi Deteksi Objek untuk modul cerdas pada Robot NAO menggunakan algoritma *Feature Descriptor Histogram of Oriented Gradient*



Gambar 4.1 Tampilan Aplikasi Deteksi Objek

Gambar diatas adalah tampilan aplikasi deteksi objek menggunakan algoritma *Histogram Of Oriented Gradient*, algoritma ini mempunyai beberapa tahapan dalam proses pendeteksian, yaitu : Akuisisi Citra, Normalisasi Warna, *Gradient Compute*, *Spatial Orientation Binning, Normalization Block, Detector Windows,* Klasifikasi *Linear Support Vector Machine.*

4.2.3.1 Akuisisi Citra

Akuisisi citra adalah tahap awal untuk mendapatkan citra digital. Tujuan akuisisi citra adalah untuk menentukan data yang diperlukan dan memilih metode perekaman citra digital. Tahap ini dimulai dari objek yang akan diambil gambarnya, persiapan alat-alat, sampai pada pencitraan. Pencitraan adalah kegiatan transformasi dari citra tampak (foto, gambar, lukisan, patung,pemandangan dan lain-lain) menjadi citra digital. Beberapa alat yang dapat digunakan untuk pencitraan adalah :

a) Video kamera
b) Kamera digital
c) Kamera konvensional dan *converter analog to digital*
d) Scanner
e) Photo sinar-X / infra merah

1.1 Data Akuisisi Citra Positif

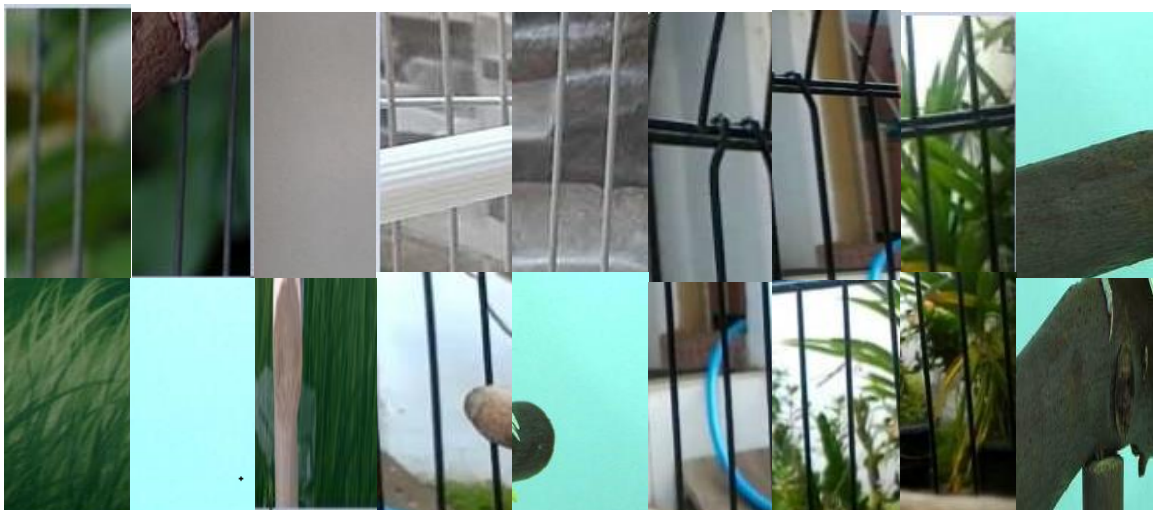
Akuisisi citra positif merupakan data training objek yang akan di deteksi, berikut adalah data training positif :

Gambar 4.5 Data Training Akuisisi Citra Positif

Gambar diatas adalah data *training* objek positif yang akan dilakukan proses pendeteksian menggunakan *algoritma Histogram of Oriented Gradient* yang akan diterapkan pada aplikasi deteksi objek untuk pendeteksian jenis-jenis burung *Lovebird*.

1.2 Data Training Akuisisi Citra Negatif

Akuisisi citra negatif merupakan data training objek yang tidak di deteksi, berikut adalah data training negatif :



Gambar 4.6 Data Training Akuisisi Citra Negatif

Gambar diatas adalah data training objek positif yang akan dilakukan proses pendeteksian menggunakan algoritma *Feature Descriptor Histogram of Oriented Gradient* yang akan diterapkan pada aplikasi deteksi objek untuk pendeteksian jenis-jenis burung *Lovebird.*

4.2.3.2 Normalisasi Warna

Pada tahapan ini adalah proses merubah gambar asli menjadi gambar grayscale .Proses ini dapat dilakukan dengan cara mengambil semua piksel pada gambar kemudian warna tiap piksel akan diambil informasi mengenai 3 warna dasar yaitu merah, biru dan hijau (melalui fungsi warna RGB), ketiga warna dasar ini akan dijumlahkan kemudian dibagi tiga sehingga didapat nilai rata-rata. Nilai rata-rata inilah yang akan dipakai untuk memberikan warna pada piksel gambar sehingga warna menjadi *grayscale*, tiga warna dasar dari sebuah piksel akan diatur menjadi nilai rata-rata (melalui fungsi RGB), untuk mendapatkan nilai *grayscale* dapat ditemukan dengan ketentuan sebagai berikut :

$$\square_0(\square,\square) = \frac{\square_{\square}^{\square}(\square,\square) + \square_{\square}^{\square}(\square,\square) + \square_{\square}^{\square}(\square,\square)}{3}$$

2.1 Proses merubah gambar asli menjadi gambar *grayscale*

Gambar 4.7 Gambar Asli Beberapa Objek Yang Akan Di Lakukan Pendeteksian

Gambar 4.8 Hasil Gambar Grayscale Beberapa Objek Yang Akan Di Lakukan Pendeteksian

4.2.33 Menghitung Nilai *Gradient (Gradient Compute)*

Menghitung Nilai Gradient *(Gradient Compute)* merupakan proses setelah konversi citra. Gradien merupakan hasil pengukuran perubahan dalam sebuah fungsi intensitas, dan sebuah citra dapat dipandang sebagai kumpulan beberapa fungsi intensitas kontinyu dari citra. Proses ini digunakan untuk mendapatkan garis tepi pada objek dalam citra. *Gradien* dari suatu gambar dapat diperoleh dengan penyaringan dengan filter 2 dimensi yaitu filter vertikal dan horisontal. Yang pertama dilakukan adalah gambar dikonversi dalam bentuk grayscale untuk menghindari keharusan untuk mempertimbangkan kontribusi intensitas yang berbeda untuk setiap bidang warna (RGB). Metode yang biasa digunakan adalah 1-D *centered*, dengan matriks seperti berikut:

$$[-1,0,1]$$

1) Menggunakan rumus derivatif parsial untuk fungsi *imag f(x,y)* :

a) Untuk sumbu *x* : $\frac{\partial f}{\partial x} = \frac{f(x+h) - f(x-h)}{2h}$

b) Untuk sumbu *y* : $\frac{\partial f}{\partial y} = \frac{f(y+h) - f(y-h)}{2h}$

2) Akan diperoleh nilai x dan y yang digunakan untuk menghitung *gradient* :

a) Magnitude (besar *gradient*) : $\square = \sqrt{\square^2 + \square^2}$

b) Orientasi gradient (dalam sudut) : $\square = \square\square\square$

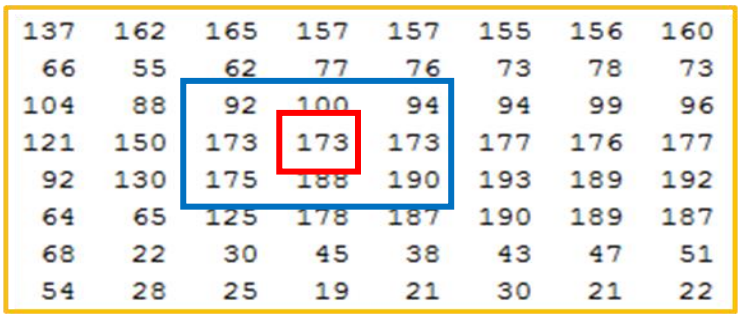
$\square\square\square\ \square\square\square\ \left(\frac{\square}{\square}\right)$

137	162	165	157	157	155	156	160
66	55	62	77	76	73	78	73
104	88	92	100	94	94	99	96
121	150	173	173	173	177	176	177
92	130	175	188	190	193	189	192
64	65	125	178	187	190	189	187
68	22	30	45	38	43	47	51
54	28	25	19	21	30	21	22

Gambar 4.9 Nilai Sel Dalam Gambar

Gambar diatas adalah tampak nilai sel dalam gambar, setiap sel terdiri dari 8X8 px.

3) Proses perhitungan *gradient* menggunakan konvolusi metode 1-D *centered*



137	162	165	157	157	155	156	160
66	55	62	77	76	73	78	73
104	88	92	100	94	94	99	96
121	150	173	173	173	177	176	177
92	130	175	188	190	193	189	192
64	65	125	178	187	190	189	187
68	22	30	45	38	43	47	51
54	28	25	19	21	30	21	22

Filter mask = -1 0 1 untuk Dx

(-1 0 1)T untuk Dy

Dx = -150 + 173 = 23

Dy = -92 + 175 = 83

Magnitude → $|\square\square| = \sqrt{\square\square^2 + \square\square^2 = 86{,}12}$

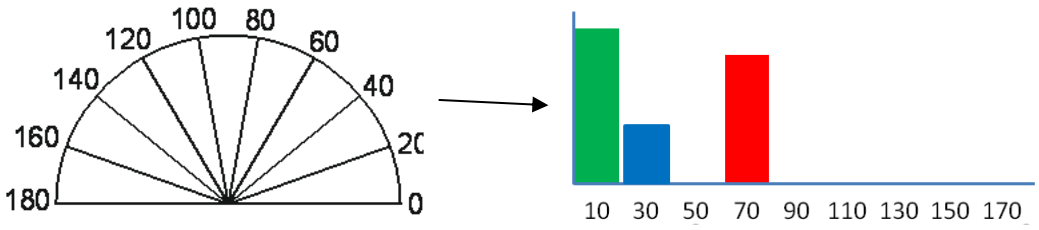
Orientation → $\square = \square\square\square\square\square\square\square\ \square\square\square\square\square\square\square \left(\frac{\square\square}{\square\square}\right) \approx 15^0$

4.2.3.4 Menentukan Bin Orientasi *(Spatial Orientation Binning)*

Pada proses ini menghasilkan pengkodean yang sensitif terhadap konten gambar lokal, namun tetap tahan terhadap perubahan kecil dalam pose dan penampilan. Gambar dibagi menjadi beberapa daerah dengan ruang kecil yang disebut sel. Untuk setiap sel akan diakumulasikan *histogram* 1-D lokal atau

orientasi tepi pada semua pixel dalam sel. Kombinasi *cell-level histogram* 1-D membentuk representasi *histogram* orientasi dasar. Setiap *histogram* orientasi membagi berbagai sudut gradient menjadi angka tetap dalam *bins* yang telah ditentukan. Besarnya gradient dari *pixel* dalam sel digunakan untuk vote ke dalam *histogram* orientasi. Misalnya akan dibangun *histogram* yang didistribusikan melalui 0°-180° dengan sejumlah channel sama dengan 9, maka untuk pemberian vote dalam *histogram* adalah sebagai berikut:

a) Semua gradient dengan besar sudut [0°-20°] memberikan vote untuk channel 1.
b) Semua *gradient* dengan besar sudut [20°-40°] memberikan vote untuk *channel* 2.
c) Dan seterusnya.



(a) (b)

Gambar 4.11 (a) Sudut Gradient, (b) Grafik *Histogram* Pada Sel

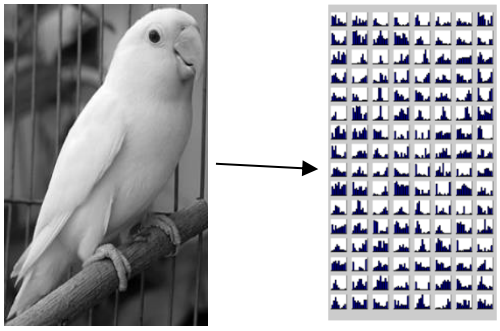
Gambar (a) adalah sudut gradient arah pencahayaan yang terdapat pada suatu citra, gambar (b) adalah grafik *Histogram* pada setiap sel dalam citra, setiap *histogram* orientasi membagi berbagai sudut *gradient* menjadi angka tetap dalam *bins* yang telah ditentukan. Besarnya gradient dari *pixel* dalam sel digunakan untuk vote ke dalam *histogram* orientasi**.** berikut adalah proses perhitungan menentukan orientasi *binnig (Spatial Orientation Binning)* pada citra kursi :

Orientation = 70 ^{o}
Orientation = 70 ^{o}
Orientation = 15 ^{o}

Magnitude = 35
Magnitude = 20
Magnitude = 86,1

$$\frac{30-15}{20} * 86 = 64{,}5 \text{ } 10^{\square}$$

$$\frac{15-10}{20} * 86 = 21{,}5 \text{ } 30^{\square}$$



(a) (b)

(c)

Gambar 4.12 (a) Citra *Grayscale*, (b) Nilai *Histogram*, (c) *Magnitude*, (d) Visualisasi Fitur *Histogram of Oriented Gradient*

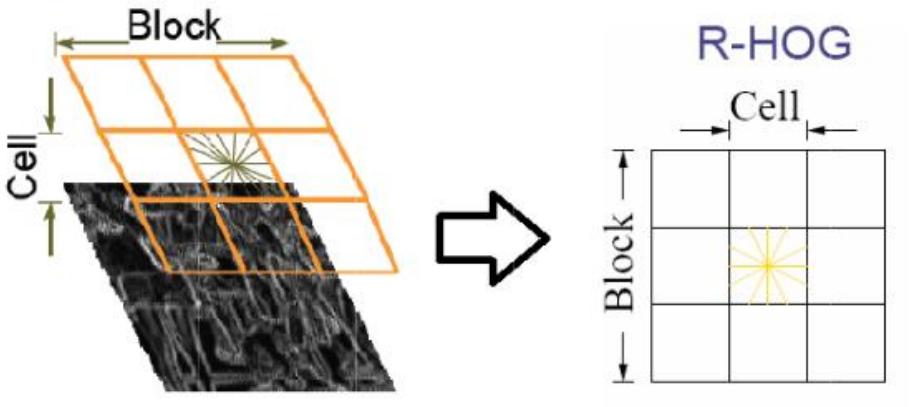
4.2.3.5 Normalisasi Blok

Pada saat proses *Compute Gradient* di peroleh nilai gradien yang berbeda-beda. Sehingga perlu pengelompokkan tiap cells dalam cells perlu dikelompokkan lagi menjadi kelompok yang lebih besar yang disebut block. Sedangkan setelah pengelompokkan menjadi blok, blok ini biasanya terjadi overlap atau tumpang tindih. Dalam normalisasi blok ini menggunakan geometri blok persegi R-HOG. Proses ini untuk merupakan proses akhir dari metode HOG yang menghasilkan fitur. Proses ini dilakukan saat proses *detector windows* seperti pada proses menghitung *bin* orientasi *bin*. Ukuran *detector windows* yang digunakan adalah 64x128 yang terdiri dari 8x8 piksel. Menghitung normalisasi blok yaitu mengambil kelompok sel dan menormalisasi respon kontras secara keseluruhan. Hal ini dilakukan dengan mengakumulasikan ukuran *histogram* dari grup sel yang disebut blok. Hasilnya akan digunakan untuk menormalisasi setiap sel dalam blok , berikut adalah penghitungan *histogram* dalam blok :

1) $\mathbf{L1} - norm: v \rightarrow v \,/\, (\|v\|_1 + \varepsilon);$
2) $\mathbf{L1} - sqrt: v \rightarrow \sqrt{v \,/\, (\|v\|_1 + \varepsilon)};$ yaitu L1 -norm yang diikuti akar kuadrat, jumlah untuk memperlakukan vector descriptor sebagai distribusi peluang.
3) $\mathbf{L2} - norm: v \rightarrow v \,/\, \sqrt{\|v\|_2^2 + \varepsilon^2};$
4) $\mathbf{L2} - Hys$, L2 -norm diikuti oleh clipping (dibatasi nilai maksimum sampaiu) dan renormalizing dimana merupakan *vector* yang tidak dinormalisasi. Standar ke-k dan sebuah konstanta kecil.

1) Descriptor Windows

Pada proses ini adalah mengumpulkan *descriptor* dari semua blok yang merupakan *overlapping grid* yang mencangkup *detection window* ke alam *feature vector* gabungan untuk digunakan dalam classifier, yang digunakan sebagai *descriptor* parameter *series* adalah R-HOG. Deskriptor blok R-HOG menggunakan sel berbentuk *grid* persegi atau persegi panjang yang *overlap*. R-HOG menghitung *grid* (yang menefinisikan angka sel paa tiap blok) dari *pixel* sel yang masing-masing mengandung *bins*, dimana tahapan ini adalah parameter.



Gambar 4.13 R-HOG Sel

Jadi apabila dengan asumsi memiliki :

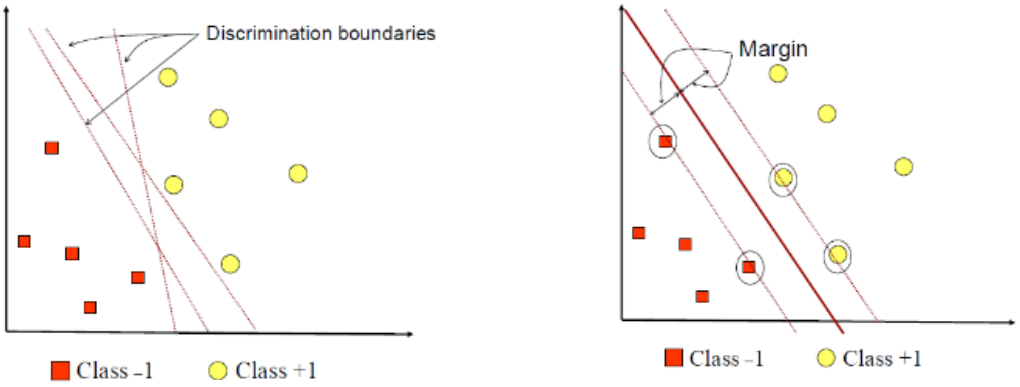
1) Ukuran Jendela
2) Sel-sel piksel (tital R-HOG sel)
3) Blok sel tanpa overlap (sebanyak 7 blok)

Maka akan diperoleh descriptor akhir ukuran : (7 X 15) X (2 X 2) X 9 = 3780

4.2.3.6 *Linear Support Vector Machine* (SVM) *Classifier*

Support Vector Machine (SVM) merupakan metode *learning machine* yang bekerja atas pronsip *Strucrural Risk Minimization* (SRM) dengan tujuan menemukan hyperplane terbaik yang memisahkan dua buah class (kategori) pada input space. Prinsip dasar *Support Vector Machine* adalah *linear classifier*, dan sedang dikembangkan agar dapat bekerja pada non-linear. Untuk mengecek apakah dalam *window* tersebut terdapat objek yang akan dideteksi atau tidak, digunakan

SVM *Classifier* untuk memisahkan human dan non-human atau dalam penelitian ini adalah pendeteksian objek. Pada SVM *Classifier* dan algoritma klasifikasi yang berusaha memisahkan sebuah *hyperplan* optimal (Cristianini & Shawe-Taylor, 2000).



Gambar 4.14 SVM dalam upayanya mencari hyperlane terbaik untuk memisahkan class -1 dan +1

Gambar diatas menunjukkan ada beberapa pattern yang merupakan anggota dari dua buah class +1 dan -1. Problem klasifikasi dapat diterjemahkan dengan usaha menemukan garis (hyperlane) yang memisahkan kedua class tersebut. *Feature* HOG juga dipakai sebagai input dari *learning* SVM. *Feature* HOG dirubah kedalam *feature vector* dengan ukuran 4608x1. Ukuran *feature vector* dihasilkan dari perkalian dari ukuran block (2x2 *cell*) , jumlah bin 9, dan banyaknya blok yang terbentuk dari *image* dengan ukuran 128x64. *Feature vector* inilah yang dipakai sebagai input untuk proses *learning* SVM.

4.2.3.7 Hasil Keakurasian Deteksi Objek Pada Aplikasi Deteksi Objek Menggunakan Algoritma *Feature Descriptor Histogram of Oriented Gradient*

Dalam proses *learning* pada SVM dilakukan 2 tahap, yaitu *training* data dan tes model. Pada *training data*, *feature* HOG dari *image* positif diberi label +1 yang menandakan bahwa *image* yang dilatih merupakan *feature image* positif dan *feature*

image negatif diberi label -1 yang menandakan *image* yang dilatih merupakan *feature image* negatif. *Training* SVM ini menghasilkan sebuah model yang akan dites dengan menggunakan *classify* SVM, berikut adalah proses *learning* dalam *Support Vector Machine :*

a) Data Set

Data set merupakan komponen yang dibutuhkan untuk data *learning* dengan menggunakan SVM. Data *learning* yang dibutuhkan berupa *training* data positif yaitu data yang berisi obyek mobil, *training* data negatif yang berisi obyek bukan mobil (*background*), data tes positif, data tes negatif, serta *image* yang akan dideteksi. *Image* yang dipakai semuanya berukuran 64x128, kecuali *image* yang akan dideteksi, memiliki ukuran 680x480, lebih besar dari *image training* dan *image* tes. Berikut adalah data *image* yang akan di deteksi :

1. 4 data citra positif untuk kursi
2. 4 data citra positif untuk monitor
3. 4 citra data positif untuk ember
4. 4 citra data positif untuk meja

Untuk *image* negatif didapatkan dengan *crop image* acak dengan ukuran 64x128. Banyaknya *image* negatif yang digunakan adalah sebagai berikut :

1. 36 *training* citra negatif
2. 36 *testing* citra negatif

Secara keseluruhan komposisi *image data set* ditunjukkan pada table berikut

Tabel 4.2 Data *Image Test* Dalam Pengujian Data *Training*

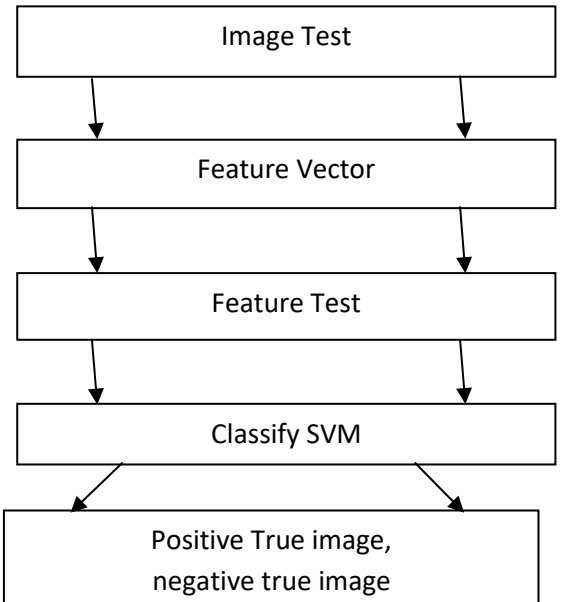
Image Positif		Image Negatif	
Training Image	Test Image	Training Image	Test Image
24	24	24	24

b) *Testing Model*

Testing image merupakan pengujian model hasil learning untuk diketahui tingkat akurasi model dalam mengenali *image* tes. *Image* tes memiliki besar piksel yang sama dengan data *training* , namun memiliki variasi *image* yang berbeda. Dari gambar 8, sebelum melakukan *testing* model terlebih dilakukan ekstraksi feature HOG dari *image* yang dites. Selanjutnya feature tersebut dirubah kedalam bentuk *feature* vector yang ukurannya sama dengan *feature vector* data *training* yaitu 4608x1. *Feature vector* dari *image* positif digabung dengan *feature vector image* negatif yang menghasilkan sebuah *feature* tes. *Feature vector* dari *image* tes ini diklasifikasikan menggunakan *classify* SVM dengan model *training* sebagai acuannya. Dari tes ini diketahui banyaknya *image* yang berhasil dideteksi sesuai dengan *class* yang ditentukan sebelumnya (*image* positif dideteksi sebagai *image* positif, *image* negatif dideteksi sebagai *image* negatif) dan juga diketahui banyaknya *image* yang tidak berhasil dideteksi sesuai dengan *class* yang ditentukan.

c) Deteksi Objek

Proses deteksi dilakukan dengan melakukan filtering pada image yang dideteksi dengan model (SVM Model) sebagai filter, yaitu dengan cara menggerakkan model dari pojok kiri atas dari image yang dideteksi sampai pojok kanan bawah. Apabila dalam proses fiter terdapat feature HOG yang bobotnya *(weight)* diatas nilai threshold yang ditentukan maka *feature* tersebut dideteksi sebagai image positif dengan memberi kotak pada bagian tersebut. Namun apabila bobot *feature* yang dideteksi dibawah nilai *threshold* yang ditentukan maka *feature* tersebut danggap sebagai *image* negatif.



Gambar 4.15 *Testing Image*

Pada pengujian ini, jumlah *training* data yang digunakan dibuat berbeda-beda. Hasil *training* disimpan dalam bentuk model kemudian dites dengan menggunakan SVM *classify*. Pengujian dilakukan dengan komposisi data *training* dengan jumlah tes *image* positif sebanyak 16 tes *image* positif dan 36 tes *image* negatif. Pengujian dilakukan sebanyak 2 kali dengan data *training* awal sebanyak 16 *image* positif dan 36 *image* negatif. Untuk pengujian pertama, menghasilkan data sebagai berikut.

Pengujian ke-1

Data *image training* :

1. *Image* positif 6 *image*
2. *Image* negatif 18 *image*

Hasil Tes :

1. Tes *image* positif : 16 correct, 3 *incorrect*, 19 total
2. Tes *Image* negatif : 30 correct, 8 *incorrect*, 38 total
3. Tes *Image* positif dan negative : 46 *correct*, 11 *incorrect*, 57 total

Pengujian ke-2

Data *image training* :

1. *Image* positif 6 *image*
2. *Image* negatif 18 *image*

Hasil Tes :

1. Tes *image* positif : 6 *correct*, 3 *incorrect*, 19 total
2. Tes *Image* negatif : 18 *correct*, 3 *incorrect*, 19 total
3. Tes *Image* positif dan *negatif* : 24 *correct*, 11 *incorrect*, 57 total

Pengujian dilanjutkan sampai 2 kali pengujian. Dari hasil pengujian tersebut didapatkan tabel 2 seperti dibawah ini.

Tabel 4.3 Hasil Pengujian Data *Training*

Training Positif	Training Negaitif	Hasil Tes Positif	Hasil Tes Negaitif	Hasil Tes Positif dan Negatif
6	18	95,39%	99,60%	98,23%
6	18	95,39%	99,60%	98,23%

Hasil tes positif menunjukkan peningkatan apabila jumlah data training ditambah. Presentase terakhir menunjukkan bahwa tingkat keberhasilan tes dengan classify SVM sebesar 99.10% dengan data training positif sebanyak 6 image dan data training negative sebanyak 18 image. Namun pada saat tes ke-2, dengan jumlah training data 6 image positif dan 18 image negative mendapatkan hasil yang sama, dikarenakan model yang dihasilkan dari training ke-2 mampu mendeteksi image dengan komposisi 30 correct, 3 incorrect, 57 total. Sedangkan hasil tes negatif menunjukkan tingkat keberhasilan sebesar 99,60% dari tes image sebanyak 36 image. Secara linear, hasil tes menunjukkan peningkatan dengan ditambahnya jumlah training data.