BAB 3
ANALISIS DAN PERANCANGAN SISTEM

3.1 Analisis Masalah

Analisis sistem merupakan salah satu cara yang digunakan dalam mengidentifikasi dan mengevaluasi permasalahan, hambatan, dan kebutuhan yang terdapat dalam suatu sistem. Hasil dari analisis sistem ini diperlukan sebagai dasar untuk tahap perancangan sistem yang akan dibangun.Dalam pengklasifikasian dan pencocokan citra manusia sangatlah penting untuk menghasilkan suatu sistem pendeteksian dan penghitungan manusia dibutuhkan tingkat akurasi yang baik.Metode yang akan digunakan pada penelitian ini adalah metode Histogram of *Oriented Gradient (HOG) – Local Binary Pattern (LBP)* untuk ekstraksi fitur dan metode *Support Vector Machine* (*SVM)* untuk klasifikasinya.

Sebelumnya telah dilakukan penelitian yang membuktikan bahwa metode HOG – LBP dapat mengekstraksi tekstur dengan baik. *Local binarry pattern* menghasilkan ekstraksi ciri berupa angka desimal berdasarkan hasil masukan citra grayscale kemudian dilakukan perbandingan antara nilai tetangga atau disebut thresholding dan hasil biner yang didapat akan dijumlahkan dengan bobot. Begitu juga dengan metode HOG, metode HOG akan menghitung hasil *gradient magnitude* dari input citra, dan citra tersebut akan dibagi menjadi 3 level yang berbeda lalu ditentukan histogram dari masing-masing level. Dan didapatkan *feature vector* akhir dari semua level. Adapun Tahap-tahap analisis antara lain akan membahas mengenai:

1. Masalah dalam proses identifikasi citra manusia pada video.
2. Data dan Metode yang diperlukan untuk pembangunan perangkat lunak pendeteksian dan penghitungan jumlah manusia.
3. Analisis Kebutuhan Non Fungsional perangkat lunak yang akan dibangun dalam penelitian.
4. Analisis Kebutuhan Fungsional perangkat lunak.

Dari yang telah dijelaskan dan dijabarkan, maka dalam penelitian ini akan melakukan analisis dengan mengkombinasikan metode ekstraksi ciri Local Binary Pattern (LBP) – Histogram of Oriented Gradient (HOG) dan metode klasifikasi Support Vector Machine (SVM) dalam pendeteksian dan penghitungan manusia pada video digital.

3.1.1 Analisis Kebutuhan Data

Sistem yang dibangun adalah pendeteksian dan penghitungan manusia dengan menggunakan metode HOG-LBP pada video digital. Pada dasarnya sistem terbagi dalam 2 tahapan, yaitu training dan pengujian.

Pada tahap training membutuhkan data masukkan yang berasal dari INRIA *person dataset* yang berupa kumpulan 2416 citra manusia dan 1.724 citra negatif yaitu citra yang tidak terdapat manusia (*background*). Citra dari dataset tersebut nantinya akan diolah menggunakan metode HOG-LBP untuk ektraksi ciri. Selanjutnya akan diolah menggunakan metode *support vector machine* (SVM) untuk menghasilkan fitur vector manusia pada tahapan learning. Contoh dataset training dapat dilihat pada Gambar 3.1.



(a)



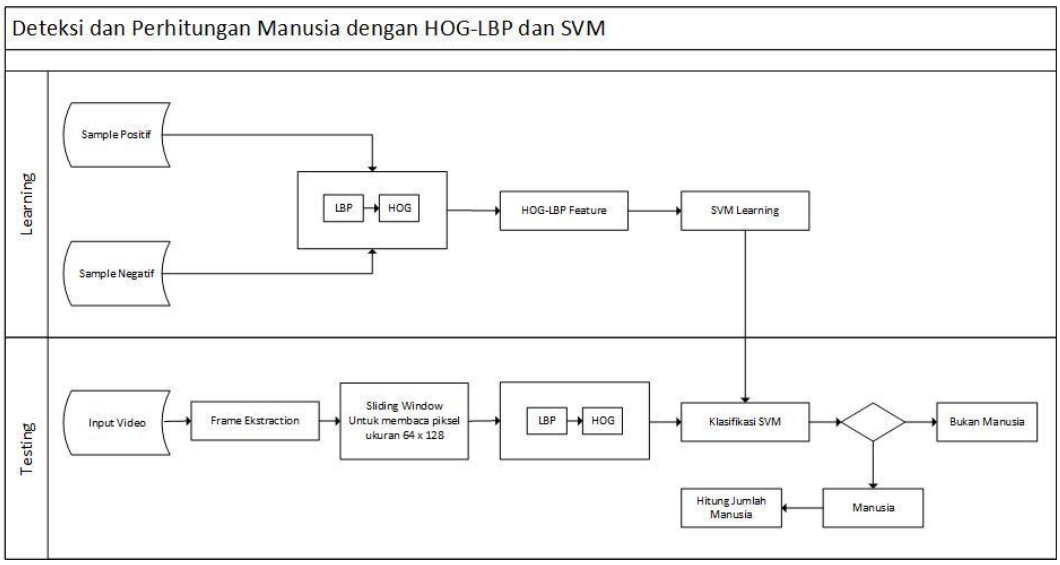
(b)

Gambar 3.1 Dataset Training Sample Positif (a) Negatif (b)

3.1.2 Analisis Sistem

Sistem yang dibangun pada penelitian ini menggunakan kombinasi HOG-LBP untuk fitur ekstraksi. Metode tersebut digunakan untuk mengekstraksi ciri manusia pada video digital. Sedangkan Metode untuk klasifikasi yaitu support vector machine yang mana untuk mendapatkan data latih lalu pada data uji diklasifikasikan oleh metode SVM tersebut.

Sistem yang dibangun terdiri dari 2 buah tahap. Tahap pertama yaitu tahap training atau disebut tahap pelatihan, tahap kedua yaitu tahap pengujian yang mana pada tahap ini data uji akan diproses dan dilakukan pencocokan dengan data latih. Gambaran deskripsi umum sistem dapat dilihat pada Gambar 3.2.



Gambar 3.2 Gambaran Umum Sistem

Penjelasan dari tahapan-tahapan diatas adalah sebagai berikut :

1. Gambar Training Positif

   Merupakan data gambar *training* positif yang disimpan dalam media penyimpanan di komputer, yaitu sejumlah 2.416 citra positif yang terdapat sosok manusia.

2. Gambar training Negatif

   Merupakan data gambar *training* negatif, yaitu 1.218 citra negatif yang merupakan gambar pemandangan yang tidak terdapat sosok manusia.

3. Proses LBP

Merupakan tahapan pengolahan citra input dengan menggunakan metode HOG-LBP untuk menghasilkan *feature vector* dari gambar yang akan diolah oleh SVM.

4. Proses SVM

   Dalam tahap ini, *feature vector* yang dihasilkan oleh proses metode HOG akan di *learning* dengan SVM. SVM bertugas untuk menemukan *hyperlane* terbaik sebagai pemisah antara *class*+1 dan *class*-1. *Class*+1 adalah kelas hasil pengolahan data *training* positif, sedangkan *class*-1 adalah kelas hasil pengolahan data *training* negatif.

5. *Frame Ekstraction*

   Merupakan data feature vector yang nantinya akan dilakukan kernelisasi dengan feature vector data uji.

6. Input Video

   Merupakan data masukan berupa video untuk dilakukan pendeteksian dan penghitungan manusia.

7. *Sliding Windows*

   Merupakan tahapan *scanning* pada jendela video untuk membaca piksel dengan ukuran 64 x 128.

8. HOG – LBP Feature

   Gabungan nilai feature HOG-LBP setelah dilakukan perhitungan masing-masing metode.

9. Klasifikasi SVM

   SVM model yang didapat dalam tahap *learning* akan digunakan dalam tahap pengujian dalam proses klasisfikasi SVM. Dalam tahap ini, *feature vector* hasil pengolahan *detector window* citra uji dengan metode HOG akan diklasifikasi apakah termasuk dalam *Class*-1 (non manusia) atau *Class*+1 (manusia).

10. Manusia dan Non Manusia

    Merupakan hasil deteksi dari metode yang sudah dilakukan. Area yang dideteksi terdapat manusia akan diberi tanda kotak berwarna kuning. Sedangkan area yang dideteksi tidak terdapat menusia akan diabaikan.

11. Hitung Jumlah Manusia

    Merupakan proses untuk menghitung berapa jumlah manusia didalam layar video tersebut.

3.1.3 Analisis Metode

Metode pada penelitian ini menggunakan 3 metode utama dan 2 metode tambahan. Metode pertama adalah *Histograms* of *Oriented Gradients*. Yang kedua yaitu metode *Local Binary Pattern*. Metode tersebut merupakan rangkaian dari beberapa tahap komputasi pada citra digital untuk mengekstraksi ciri fitur manusia pada citra. Sedangkan metode ketiga adalah algoritma *Support Vector Machine* yang berfungsi untuk mengklasifikasi manusia pada citra digital. Sedangakan 2 metode tambahan yaitu sliding windows dan non maxima supression.

Sebelum masuk tahapan ekstraksi ciri atau tekstur, video tersebut diekstraksi frame per framenya terlebih dahulu yang nantinya akan dilakukan sliding windows pada frame per frame tersebut untuk pendeteksian manusia. Contoh pada sebuah video berdurasi 09.36 dengan frame rate 5 fps, untuk mendapatkan seluruh frame maka dikalikan durasi dengan fps dan didapatkan hasil 48 frame pada video dengan durasi 09.36. Berikut adalah deretan frame dari video.



Gambar 3.3 Deretan Frame Pada Video

Setelah itu setiap frame pada video dipotong-potong hingga menghasilkan citra dari frame. Setiap citra tersebut dilakukan sliding windows untuk melakukan pendeteksian.

3.1.3.1 Metode Local Binary Pattern (LBP)

LBP merupakan salah satu metode ekstraksi ciri yang mendeskripsikan tekstur. LBP membandingkan nilai biner piksel pada pusat citra dengan nilai piksel tetangganya. Metode local binary pattern adalah metode untuk ekstraksi ciri yang akan menghasilkan ciri dari sebuah citra. Terdapat beberapa proses untuk mengekstraksi tekstur LBP yaitu sebagai berikut :

1. Mengkonversi Citra RGB ke Citra Grayscale

Citra yang menjadi input sistem merupakan citra RGB (Red-Green-Blue). Proses dalam tahap ini adalah mengubah nilai warna RGB pada setiap pixel citra menjadi nilai derajat keabuan. Banyak metode yang dapat digunakan untuk mengkonversi nilai *pixel* dari citra RGB menjadi *grayscale*, namun metode *grayscaling* yang digunakan adalah metode *Luminosity* seperti pada persamaan 2.7.

Misalkan pada video diambil piksel berukuran 620 x 360 diambil salah satu frame untuk diekstraksi seperti pada Gambar 3.4, lalu diambil matriks dari gambar tersebut :



Gambar 3.4 Citra RGB

Misalkan dari Gambar 3.4 diambil matriks RGB seperti pada tabel Tabel 3.1.

Tabel 3.1 Channel RGB

	Piksel (1,y)	Piksel (2,y)	Piksel (3,y)		Piksel (620,y)
Piksel (x,1)	Red = 121 Green = 119 Blue= 104	Red = 112 Green = 112 Blue= 100	Red = 113 Green = 114 Blue= 100	.....	Red = 147 Green = 140 Blue= 132
Piksel (x,2)	Red = 118 Green = 115 Blue= 106	Red = 110 Green = 110 Blue= 102	Red = 110 Green = 111 Blue= 97	.....	Red = 134 Green = 124 Blue= 115
Piksel (x,3)	Red = 111 Green = 111 Blue= 99	Red = 104 Green = 106 Blue= 93	Red = 104 Green = 105 Blue= 89	.....	Red = 119 Green = 100 Blue= 93
Piksel (x,4)	Red = 1 Green = 1 Blue= 3	Red = 11 Green = 13 Blue= 8	Red = 104 Green = 106 Blue= 85	.....	Red = 174 Green = 149 Blue= 129
Piksel (x,5)	Red = 5 Green = 5 Blue= 9	Red = 9 Green = 10 Blue= 5	Red = 93 Green = 92 Blue= 72	.....	Red = 164 Green = 137 Blue= 118
Piksel (x,6)	Red = 8 Green = 8 Blue= 0	Red = 6 Green = 7 Blue= 2	Red = 86 Green = 85 Blue= 64	.....	Red = 163 Green = 136 Blue= 117
Piksel (x,7)	Red = 8 Green = 7 Blue= 12	Red = 7 Green = 7 Blue= 5	Red = 86 Green = 85 Blue= 67	.....	Red = 162 Green = 135 Blue= 118
	.....	.....	.....	.....	.....
Piksel (x,360)	Red =96 Green = 90 Blue= 74	Red =90 Green = 84 Blue= 68	Red =90 Green = 84 Blue= 68	.....	Red = 142 Green = 130 Blue= 114

Contoh kita ambil matriks pada piksel (1,1).

Maka hasil dari perhitungan dari RGB ke grayscale adalah sebagai berikut :

Grayscale = (0.299 * *R*) + (0.587 * *G*) + (0.114 * *B*)

Grayscale = (0.299 * 121) + (0.587 * 119) + (0.114 * *104*)

Grayscale = 117,888 = 118

Jadi didapatkan nilai grayscale dari piksel (1,1) dari citra sebesar 118. Berikut pada Tabel 3.2 adalah matriks citra grayscale.

Tabel 3.2 Matriks Citra Grayscale

	Piksel (1,y)	Piksel (2,y)	Piksel (3,y)		Piksel (620,y)
Piksel (x,1)	118	111	112	.....	141
Piksel (x,2)	115	109	109	.....	126
Piksel (x,3)	110	104	103	.....	105
Piksel (x,4)	1	12	103	.....	154
Piksel (x,5)	5	9	90	.....	143
Piksel (x,6)	8	6	83		142
	.....	.....	.....	.....	
Piksel (x,360)	90	84	84	.....	132

Proses perhitungan seperti pada contoh sebelumnya akan dilakukan pada citra RGB dari pixel dari posisi *f*(0,0) sampai *f*(*n*-1,*n*-1). Contoh hasil dari konversi citra RGB ke citra *grayscale* bisa dilihat pada Gambar 3.5 berikut ini:



Gambar 3.5 Citra Grayscale

2. Fitur Local Binary Pattern

LBP merupakan salah satu metode ekstraksi ciri yang mendeskripsikan tekstur. LBP membandingkan nilai biner piksel pada pusat citra dengan nilai piksel tetangganya. Metode local binary pattern adalah metode untuk ekstraksi ciri yang akan menghasilkan ciri dari sebuah citra. Berikut adalahtabel bobot LBP.

Tabel 3.3 Bobot LBP

1	2	4
128		8
64	32	16

Pada Tabel 3.3 adalah bobot dari LBP yang kemudian akan dibandingkan dengan nilai threshold. Kemudian menyusun 8 nilai biner tersebut searah jarum jam atau sebaliknya. Misal diambil matriks 3x3 dari Tabel 3.2 pada piksel (2,2) , (2,3) dan (2,4).

118	111	112
115	109	109
110	104	103

Pattern = 11110011 **LBP = 1+2+4+8+0+0+64+128 = 206**

115	109	109
110	104	103
1	12	103

Pattern = 11100000 **LBP = 1+2+4+0+0+0+0+0 = 7**

110	104	103
1	12	103
5	9	90

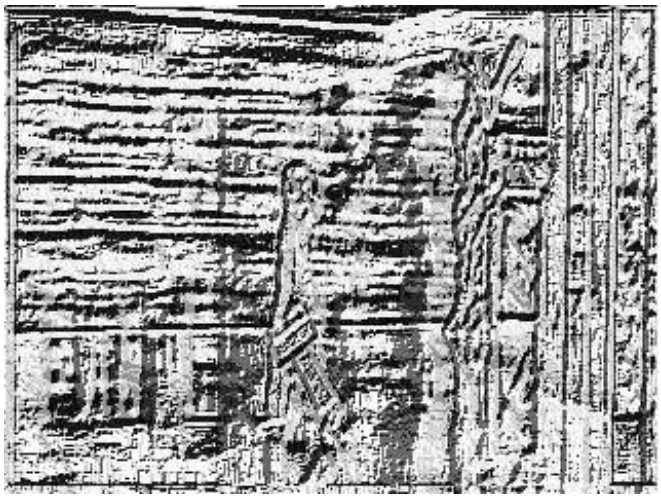
Pattern = 11111000 **LBP = 1+2+4+8+16+0+0+0 = 32**

Sehingga menghasilkan matriks LBP seperti pada tabel 3.4.

Tabel 3.4 Matriks Citra LBP

	Piksel (x,1)	Piksel (x,2)	Piksel (x,3)		Piksel (x,618)
Piksel (1,y)	206	135	143	.....	31
Piksel (2,y)	7	255	55	.....	247
Piksel (3,y)	32	23	0	.....	184
Piksel (4,y)	151	150	23	.....	255
Piksel (5,y)	150	214	31	.....	253
Piksel (6,y)	189	159	66		191
	.....	.....	.....	.....	
Piksel (358,y)	251	237	255	.....	252

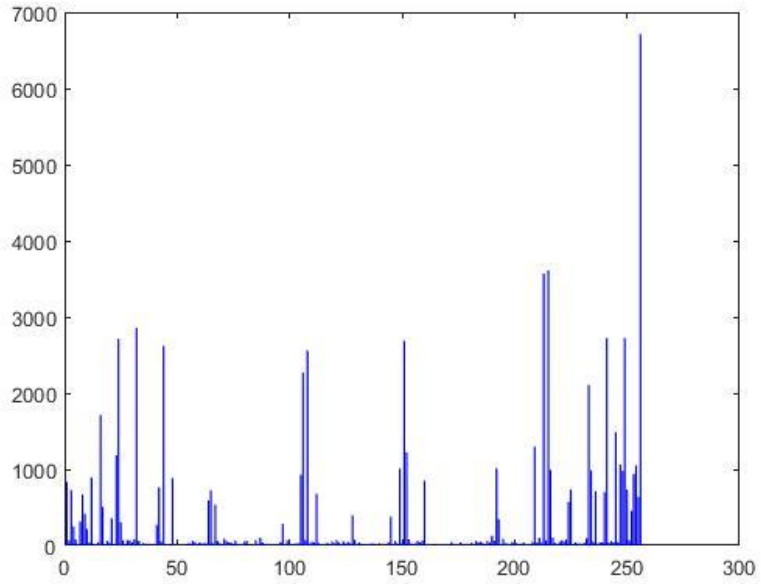
Berikut pada Gambar 3.6 adalah contoh dari tekstur local binary pattern.



Gambar 3.6 Ekstraksi Ciri Citra LBP

3. Histogram LBP

Langkah selanjutnya nilai setiap piksel pada citra hasil LBP dilakukan pemetaan kedalam histogram dari matriks hasil citra LBP. Pemetaan dipetakan kedalam 256 bin. Berikut pada Gambar 3.7 adalah Histogram dari LBP.



Gambar 3.7 Histogram LBP

4. Feature Vector

Feature Vector dari citra LBP didapatkan dari hasil histogram LBP dimana nilai semua histogram dibagi dengan jumlah semua histogram yang memvoting histogram LBP tersebut.

Berikut adalah feature vector dari citra LBP.

0:0.124143	1:0.009836	2:0.108495	3:0.108495	4:0.011624
5:0.046945	6:0.099851	7:0.061699	8:0.032191	9:0.006259
10:0.02787	11: 0.002981	12: 0.001937	.....	255: 0,39235

Gambar 3.8 Feature Vector LBP

3.1.3.2 Metode Histogram of Oriented Gradient

Histogram of Oriented Gradients adalah pendeskripsi tekstur. HOG Secara umum bertujuan untuk mencari garis tepi objek-objek yang ada dalam citra yang sedang diproses berdasarkan gradien yang dibentuk dari gradasi antar pixel.

Sedangkan *Histogram* berkaitan dengan perhitungan seberapa banyak pixel yang memiliki garis tepi dengan orientasi arah yang spesifik. Berikut adalah tahapan metode HOG.

1. Ubah ke *Grayscale*

Sama seperti pada metode LBP, pada metode HOG pun proses pertama citra dikonversikan terlebih dahulu ke cirra *grayscale.*

2. Hitung *Gradient*

Proses selanjutnya yang dilakukan pada tahapan metode HOG yaitu menghitung nilai gradient dari citra grayscale. Adapun metode untuk mencari nilai gradient yaitu dengan metode sobel.:

$$S(x) = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \text{ dan } S(y) = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

Filter kernel tersebut akan dikonvolusikan dengan citra grayscale. Matriks citra grayscale diinisialisasi dengan *I.* Berikut adalah perhitungan dari operasi konvolusi.

$$Ix = I * S(x) \quad \text{dan} \quad Iy = I * S(y)$$

Misalkan diambil matriks citragrayscale dari gambar 3.4 dengan piksel 5 x 5.

$$I \begin{bmatrix} p_{11} & \cdots & a_{15} \\ \vdots & \ddots & \vdots \\ p_{51} & \cdots & a_{55} \end{bmatrix} \rightarrow I = \begin{bmatrix} 118 & 111 & 112 & 112 & 112 \\ 115 & 109 & 109 & 108 & 110 \\ 110 & 104 & 103 & 105 & 103 \\ 1 & 12 & 103 & 120 & 116 \\ 5 & 9 & 90 & 102 & 94 \end{bmatrix}$$

Sebagai contoh akan dihitung nilai konvolusi dari *Ix* (nilai *I* pada sumbu *x*) dan *Iy* (nilai *I* pada sumbu *y*) pada posisi *p*22 dengan filter persamaan 2.10 sebagai berikut:

$$Ix(p_{22}) = \begin{bmatrix} p_{11} & p_{12} & p_{13} \\ p_{21} & p_{22} & p_{23} \\ p_{31} & p_{32} & p_{33} \end{bmatrix} \text{ x } \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 118 & 111 & 112 \\ 115 & 109 & 109 \\ 110 & 104 & 103 \end{bmatrix} \text{ x } \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

= |(118*-1) + (115*-2) + (110*-1) + (112*1) + (109*2) + (103*1)|

= |-118 -230 -110 + 112 + 109 + 103 | = | -558 + 324 | = 234

Jadi nilai *Ix* pada piksel p_{22} adalah 234.

$$Iy(p_{22}) = \begin{bmatrix} p_{11} & p_{12} & p_{13} \\ p_{21} & p_{22} & p_{23} \\ p_{31} & p_{32} & p_{33} \end{bmatrix} \text{x} \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

$$= \begin{bmatrix} 118 & 111 & 112 \\ 115 & 109 & 109 \\ 110 & 104 & 103 \end{bmatrix} \text{ x } \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

= |(118*1) + (111*2) + (112*1) + (110*-1) + (104*-2) + (103*-1)|

= | 118+ 222 + 112 -110 -208 -103 | = | 452 – 421 | = 31

Jadi nilai *Iy* pada pixel p_{22} adalah 31.

Setelah didapatnya nilai I_x dan I_y pada koordinat maka dapat dihitung nilai *gradient magnitude* dan orientasi gradien dengan menggunakan persamaan seperti dibawah ini.

$$|G| = \sqrt{Ix^2 + Iy^2}$$

$|G| = \sqrt{234^2 + 31^2} = \ |G| = \sqrt{54756 + 961}$ = 236,0445

Maka didapatkan nilai gradient magnitude sebesar 236,0445

$\theta = \arctan \frac{Iy}{Ix} = \arctan \frac{234}{31} =$ 7,55

Maka didapatkan nilai orientasi gradient sebesar 7,55.

Berikut pada Gambar 3.9 adalah hasil citra dengan perhitungan gradient.



Gambar 3.9 Citra Hasil Perhitungan Gradient

Pada Tabel 3.5 adalah nilai orientasi gradient dari Gambar 3.9.

Tabel 3.5 Matriks Hasil Orientasi Gradient

	Piksel(x,1)	Piksel(x,2)	Piksel(x,3)	Piksel(x,4)		Piksel(x,620)
Piksel(1,y)	160	109	109	108	.....	178
Piksel(2,y)	109	236	9	7	.....	130
Piksel(3,y)	154	97	9	12	.....	107
Piksel(4,y)	106	139	6	13	.....	144
Piksel(5,y)	11	85	108	27	.....	134
Piksel(6,y)	6	75	95	9	.....	135
Piksel(x,y)	.....	.....	.....	.....	.....	.....
Piksel(1,360)	124	84	84	83	.....	176

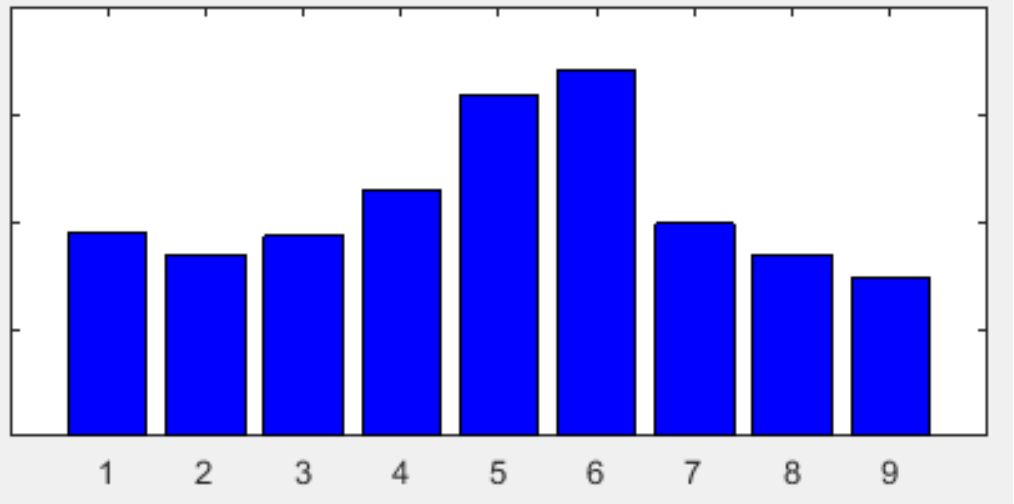
3. Histogram dan Orientasi Cell

Pada tahap selanjutnya nilai magnitude dan orientasi gradien tiap piksel akan digunakan pada tahap selanjutnya. Citra dibagi kedalam kelompok pixel yang disebut cell berukuran 8x8 dilanjutkan dengan tahap kalkulasi dengan membuat histogram pada tiap cell.

Pada tiap piksel dalam cell akan dikelompokan untuk orientasi histogram dalam cell. Tiap-tiap piksel pada cell akan memberikan vote untuk orientasi

histogram dalam cell. Vote antara 0^o - 180^o . Jadi dalam sebuah cell terdapat 64 gradient magnitude. Dari 64 gradient magnitude tersebut dipetakan kedalam 9 bin histogram.

Nilai gradient magnitude akan disebar dalam histogram seperti pada Gambar 3.10 berikut.



Gambar 3.10 Histogram

4. Normalisasi Block

Dari cell yang telah dibentuk kemudian dikelompokkan lagi dalam *block* berupa persegi panjang (R-HOG). Satu buah *block* berukuran 2x2 *cell*. R-HOG diwakili oleh tiga parameter yaitu jumlah cell per *block*, jumlah *pixel* per *cell*, dan jumlah bin per histogram. Dalam Gambar 3.11 erdapat contoh representasi block.



Gambar 3.11 Gambar Block dan Cell

Dalam Gambar 3.11 terlihat *block* 1 dan *block* 2 terdapat area yang saling tumpang tindih. Area ini perlu dilakukan normalisasi kontras. Normalisasi dilakukan dengan menerapkan persamaan yaitu:

$$b = \frac{b}{\sqrt{b^2 + e}}$$

Misal dari gambar diatas diambil nilai bloknya.

$$b = \frac{0.000981}{\sqrt{0.000981^2 + \ 2{,}71828}}$$

$$b = 0.00059225$$

Variabel b merupakan nilai blok fitur dan variabel e merupakan bilangan positif yang bernilai kecil untuk mencegah pembagian nol. Nilai normalisasi kemudian digabungkan menjadi satu vektor menjadi fitur vektor HOG yang mana langkah selanjutnya dilakukan normalisasi dengan persamaan :

$$h = \frac{h}{||h||^2 + e}$$

Nilai normalisasi dibagi dengan nilai cell histogram yang mengalami penumpukan. Berikut adalah nilai normalisasi *overlaping*.

$$h = \frac{0.0167821}{||0.0167821||^2 + e}$$

$$h = \frac{0.0167821}{3.427278}$$

$$h = 0.004897$$

5. Detector Window

Tahap terakhir dalam metode HOG adalah mengumpulkan nilai *histograms of oriented gradients* dalam *detector window*. Semua histogram untuk setiap *block* akan digabungkan untuk menghasilkan sebuah *feature vector*.

Tiap *block* terdapat 4 cell dan nilai tiap cell akan diberikan vote kedalam 9 bin, maka didapatkan 46 nilai fitur tiap block, dalam detector window yang berukuran 64 x 128 terdapat 105 block, sehingga 105 x 36 = 3780. Maka akan didapatkan feature vector hog yang berjumlah 3780.

Maka akan menghasilkan nilai feature vector seperti berikut.

0:0.004897	1:0.1534	2:0.77104	3:0.15227	4:0.4201
5:0.2234	6:0.15199	7:0.4763	8:0.0968	9:0.0882
10:0.3185	11:0.1946	12:0.6064	.....	3780:0.4912

Gambar 3.12 Feature Vector HOG

3.1.3.3 Metode Support Vector Machine

Pada metode ini feature vector gabungan hasil dari citra HOG dan LBP kemudian diolah oleh SVM. Feature vector dengan rentang 3780 digabungkan dengan feature vector dengan rentang 256, maka didapatkan matriks feature vector 4036x1. Feature Vector didapat dari citra training positif dan negatif. Citra positif merupakan citra yang terdapat manusia, sedangkan citra negatif adalah citra yang tidak

terdapat manusia berupa gedung-gedung, ruangan di dalam gedung, jalanan, dan pemandangan lansekap.Dianotasikan sebagai berikut.

$$y_i \in \{-1, +1\}$$

Yang mana i = 1, 2, 3,..... *l* dimana *l* adalah banyaknya data. Pada penelitian ini terdapat 2 kelas maka jumlah hyperlane yang ada adalah (2.(2-1)/2=1.

Sebagai contoh, karena jumlah data *vector* dan *feature vector* yang sangat besar maka akan digunakan nilai *vector* pada *cell* pertama citra positif dan negatif begitu juga feature vector diambil 9 nilai pertama dari dari 4036 bins seperti pada Tabel 3.8.

Tabel 3.6 Contoh Feature Vector Dari Citra

Citra	Kelas	Feature Vector	Kelas
	Positif	0:0.631242 1:0.613269 2:0.241379 3:0.041825 4:0.523147 5:0.347685 6:0.178963 7:0.213741 8:0.596831	1
	Negatif	0:0.137843 1:0.312798 2:0.431698 3:0.021642 4:0.021743 5:0.114591 6:0.054973 7:0.261231 8:0.141985	-1
	Positif	0:0.925129 1:0.752484 2:0.447370 3:0.175951 4:0.445637 5:0.086025 6:0.177341 7:0.276194 8:0.276638	1

Secara garis besar cara kerja SVM adalah sebagai berikut :

1. Feature Vector manusia yang didapat dari gabungan perhitungan metode HOG dan LBP, yaitu matriks yang berukuran 1 x 4036, yang akan dijadikan data masukan awal. Data yang digunakan terbagi 2 yaitu data *Training* dan data *Testing*.
2. Data masukan awal yang berjumlah 4140 yang telah didapatkan ekstraksi cirinya akan digabungkan dari citra ke-1 sampai citra ke-4140 dalam matriks *x* yang berukuran 4140x4036. Sedangkan untuk jumlah kelasnya yang berjumlah 2 kelas dibuatlah matriks *y* dengan ukuran 4140x1.

Tabel 3.7 TabelX dan TabelY

Jumlah Data	Y	X
1	1	Feature Vector ke-1
2	1	Feature Vector ke-2
3	1	Feature Vector ke-3
....	....	.....
2416	1	Feature Vector ke-2416
2417	-1	Feature Vector ke-2417
2418	-1	Feature Vector ke-2418
....	....	.....
4140	-1	Feature Vector ke-4140

3. Karena data masih berada dalam masukan *input space*, ukuran data yang sangat besar dan *Non-Linearly Separable Data*, maka harus ditransformasikan ke ruang yang lebih besar (*feature* space). Dalam praktiknya, *feature space* biasanya memiliki dimensi yang lebih tinggi dari *input space*, sehingga mengakibatkan komputasi yang sangat besar karena ada kemungkinan *feature space* memiliki *feature* yang tak terhingga, dan untuk mengatsi masalah ini, pada SVM digunakan "kernel trick*" $K(x.x_i)$*

   Misalkan data yang ada berasal dari tabel 3.7 di atas. Kernelisasi menggunakan fungsi linier *kernel* yang didefiniskan sebagai $K(x_i,x_j) = x^T x$. Matriks kernel K dihitung dengan dimensi *l* x *l*, dimana l adalah banyaknya data.

$x1$ = [0.631242 0.613269 0.241379 0.041825 0.523147 0.347685 0.178963 0.213741 0.596831]

$x2$ = [0.137843 0.312798 0.431698 0.021642 0.021743 0.114591 0.054973 0.261231 0.141985]

x3 = [0.925129 0.752484 0.447370 0.175951 0.445637 0.086025 0.177341 0.276194 0.276638]

$x_1^T x_1$ = [0.631242 0.613269 0.241379 0.041825 0.523147 0.347685 0.178963 0.213741 0.596831] ^{T} *
[0.631242 0.613269 0.241379 0.041825 0.523147 0.347685 0.178963 0.213741 0.596831]
= 0.39847

$x_1^T x_2$ =[0.631242 0.613269 0.241379 0.041825 0.523147 0.347685 0.178963 0.213741 0.596831] ^{T} *
[0.137843 0.312798 0.431698 0.021642 0.021743 0.114591 0.054973 0.261231 0.141985]
= 0.08701

$x_1^T x_3$ = [0.631242 0.613269 0.241379 0.041825 0.523147 0.347685 0.178963 0.213741 0.596831] ^{T} *
[0.925129 0.752484 0.447370 0.175951 0.445637 0.086025 0.177341 0.276194 0.276638]
= 0.0.58398

$x_2^T x_1$ = [0.137843 0.312798 0.431698 0.021642 0.021743 0.114591 0.054973 0.261231 0.141985] ^{T} *
[0.631242 0.613269 0.241379 0.041825 0.523147 0.347685 0.178963 0.213741 0.596831]
= 0,08701

$x_2^T x_2$ = [0.137843 0.312798 0.431698 0.021642 0.021743 0.114591 0.054973 0.261231 0.141985] ^{T} *
[0.137843 0.312798 0.431698 0.021642 0.021743 0.114591 0.054973 0.261231 0.141985]
= 0,01900

$x_2^T x_3$ = [0.137843 0.312798 0.431698 0.021642 0.021743 0.114591 0.054973 0.261231 0.141985] ^{T} *

[0.925129 0.752484 0.447370 0.175951 0.445637 0.086025 0.177341 0.276194 0.276638]

= 0,12752

$x_3{}^T x_1$ = [0.925129 0.752484 0.447370 0.175951 0.445637 0.086025 0.177341 0.276194 0.276638] T *

[0.631242 0.613269 0.241379 0.041825 0.523147 0.347685 0.178963 0.213741 0.596831]

= 0,58398

$x_3{}^T x_2$ = [0.925129 0.752484 0.447370 0.175951 0.445637 0.086025 0.177341 0.276194 0.276638] T *

[0.137843 0.312798 0.431698 0.021642 0.021743 0.114591 0.054973 0.261231 0.141985]

= 0,12752

$x_3{}^T x_3$ = [0.925129 0.752484 0.447370 0.175951 0.445637 0.086025 0.177341 0.276194 0.276638] T *

[0.925129 0.752484 0.447370 0.175951 0.445637 0.086025 0.177341 0.276194 0.276638]

= 0,85586

Setelah dilakukan perhitungan pada seluruh nilai x pada data *training*, maka matriks yang terbentuk dari hasil perhitungan adalah sebagai berikut.

$$x_i^T x_j = \begin{bmatrix} 0.39847 & 0.08701 & 0.58398 \\ 0.08701 & 0.19001 & 0.12752 \\ 0.58398 & 0.12752 & 0.85586 \end{bmatrix}$$

Kemudian tahap selanjutnya adalah melakukan perhitungan terhadap y. Untuk nilai y adalah nilai dari label yang diberikan.

$y_1{}^T y_1$ = 1

$y_1{}^T y_2$ = -1

$y_1{}^T y_3$ = 1

$y_2{}^T y_1$ = -1

$$y_2^T y_2 = 1$$

$$y_2^T y_3 = -1$$

$$y_3^T y_1 = 1$$

$$y_3^T y_2 = -1$$

$$y_3^T y_3 = 1$$

Sehingga matriks yang terbentuk dari hasil perhitungan $y_i^T y_j$ hasilnya adalah sebagai berikut.

$$y_i^T y_j = \begin{bmatrix} 1 & -1 & 1 \\ -1 & 1 & -1 \\ 1 & -1 & 1 \end{bmatrix}$$

Tahap selanjutnya mengubah setiap *data training* menjadi nilai vektor (*support vector*) = $\frac{x}{y}$ agar mendapatkan nilai ai. Nilai x didapatkan menggunakan persamaan 2.31 kernel *linear* sebagai berikut.

$X_{T1} = x_1^T x_1 + x_1^T x_2 + x_1^T x_3 =$ 0,398466 + 0,087012 + 0,58398 = 1,069459

$X_{T2} = x_2^T x_1 + x_2^T x_2 + x_2^T x_3 =$ 0,087012 + 0,19001 + 0,127523 = 0,233536

$X_{T3} = x_3^T x_1 + x_3^T x_2 + x_3^T x_3 =$ 0,58398 + 0,127523 + 0,855864 = 1,567367

Sehingga didapatkan untuk nilai x untuk setiap data *training* sebagai berikut.

Tabel 3.8 Nilai X Pada Citra Training

Data Training	T1	T2	T3
X	1,069459	0,233536	1,567367

Begitu juga dengan nilai y didapatkan menggunakan persamaan 2.32 kernel *linear* untuk y yaitu sebagai berikut.

$Y_{T1} = y_1^T y_1 + y_1^T y_2 + y_1^T y_3 =$ 1 - 1 + 1 = 1

$Y_{T2} = y_2^T y_1 + y_2^T y_2 + y_2^T y_3 =$ -1 + 1 - 1 = -1

$Y_{T3} = y_3^T y_1 + x_3^T y_2 + y_3^T y_3 =$ 1 - 1 + 1 = 1

Sehingga didapatkan untuk nilai y untuk setiap *data training* seperti pada tabel 3.9.

Tabel 3.9 Nilai Y Untuk Setiap Citra Training

Data Training	T1	T2	T3
Y	1	-1	1

Setelah nilai x dan y didapatkan, substitusikan nilai xn yang didapat dari xT_2 dan yn dari yT_2 yang disubtitusikan ke persamaan 2.30 sebagai berikut.

$$\sqrt{x_n^2 \quad + y_n^2 \quad} = \sqrt{0{,}233536^2 + (-1)^2} = 1{,}026908$$

Karena $\sqrt{x_n^2 \quad + y_n^2 \quad} < 2$, maka $\varphi\left[\frac{x}{y}\right] = \left[\frac{x}{y}\right]$

$$\varphi \mathrm{T}_1 = \begin{bmatrix} 1{,}069459 \\ 1 \end{bmatrix}, \varphi \mathrm{T}_2 = \begin{bmatrix} 0{,}233536 \\ -1 \end{bmatrix}, \varphi \mathrm{T}_2 = \begin{bmatrix} 1{,}567367 \\ 1 \end{bmatrix}$$

Setelah itu masing-masing *support vector* diberi nilai bias 1, untuk mendapatkan jarak tegak lurus yang optimal dengan mempertimbangkan vektor positif, serta membantu mendapatkan nilai b atau nilai *hyperplane* yaitu sebagai berikut.

Tabel 3.10 Nilai φ Untuk Setiap Citra Training

Data Training	T1	T2	T3
Support Vector Bias	$\begin{bmatrix} 1{,}069459 \\ 1 \\ 1 \end{bmatrix}$	$\begin{bmatrix} 0{,}233536 \\ -1 \\ 1 \end{bmatrix}$	$\begin{bmatrix} 1{,}567367 \\ 1 \\ 1 \end{bmatrix}$

Setelah mendapat nilai *support vector*, kemudian tahap selanjutnya adalah mencari nilai ai, didapatkan dengan mengalikan setiap data *training* menggunakan persamaan 2.33 sebagai berikut.

Menghitung a_i pada T1

$$a1 \begin{bmatrix} 1{,}069459 \\ 1 \\ 1 \end{bmatrix} T \begin{bmatrix} 1{,}069459 \\ 1 \\ 1 \end{bmatrix} = 1{,}143743\ a_1 + 1a_1 + 1a_1 = 3{,}143743\ a_1$$

$$a2 \begin{bmatrix} 1{,}069459 \\ 1 \\ 1 \end{bmatrix} T \begin{bmatrix} 0{,}233536 \\ -1 \\ 1 \end{bmatrix} = 0{,}249757\ a_2 - 1a_2 + 1a_2 = 0{,}249757\ a_2$$

$$a3\begin{bmatrix}1{,}069459\\1\\1\end{bmatrix}T\begin{bmatrix}1{,}567367\\1\\1\end{bmatrix} = 1{,}676235\ a_3 + 1a_3 + 1a_3 = 3{,}676235\ a_3$$

Menghitung a_i pada T2

$$a1\begin{bmatrix}0{,}233536\\-1\\1\end{bmatrix}T\begin{bmatrix}1{,}069459\\1\\1\end{bmatrix} = 0{,}249757\ a_1 - 1a_1 + 1a_1 = 0{,}249757$$

$$a2\begin{bmatrix}0{,}233536\\-1\\1\end{bmatrix}T\begin{bmatrix}0{,}233536\\-1\\1\end{bmatrix} = 0{,}054539\ a_2 + 1a_2 + 1a_2 = 2{,}054539$$

$$a2\begin{bmatrix}0{,}233536\\-1\\1\end{bmatrix}T\begin{bmatrix}1{,}567367\\1\\1\end{bmatrix} = 0{,}366037\ a_3 + - 1a_3 + 1a_3 = 0{,}366037$$

Menghitung ai pada T3

$$a1\begin{bmatrix}1{,}567367\\1\\1\end{bmatrix}T\begin{bmatrix}1{,}069459\\1\\1\end{bmatrix} = 1{,}676235\ a_1 + 1a_1 + 1a_1 = 3{,}676235$$

$$a2\begin{bmatrix}1{,}567367\\1\\1\end{bmatrix}T\begin{bmatrix}0{,}233536\\-1\\1\end{bmatrix} = 0{,}366037\ a_2 - 1a_2 + 1a_2 = 0{,}366037$$

$$a3\begin{bmatrix}1{,}567367\\1\\1\end{bmatrix}T\begin{bmatrix}1{,}567367\\1\\1\end{bmatrix} = 2{,}456639\ a_3 + 1a_3 + 1a_3 = 4{,}456639$$

Setelah dilakukan perhitungan pada seluruh *data training* kemudian cari parameter ai menggunakan persamaan 2.33 sebagai berikut.

$$3{,}144\ a1 + 0{,}250\ a2 + 3{,}676\ a3 = 1 \quad \text{....(1)}$$

$$0{,}250\ a1 + 2{,}054\ a2 + 0{,}366\ a3 = -1 \quad \text{....(2)}$$

$$3{,}676\ a1 + 0{,}366\ a2 + 4{,}457\ a3 = 1 \quad \text{....(3)}$$

Persamaan diatas termasuk kedalam permasalahan Sistem Persamaan Linier Tiga Variabel (SPLTV), untuk mendapatkan nilai a_1, a_2, dan a_3 maka dapat digunakan metode substitusi dan eliminasi sebagai berikut.

$$3{,}676\ a3 = 1 - 3{,}144\ a1 - 0{,}250\ a2$$

$$a3 = 0{,}272 - 0{,}855\ a1 - 0{,}068\ a2 \quad \text{....(4)}$$

Dari persamaan (2) subtitusikan dengan persamaan (4).

0,250 a1 + 2,054 a2 + 0,366 (0,272 – 0,855 a1 – 0,068 a2) = -1

0,250 a1 + 2,054 a2 + 0,1 – 0,313 a1 -0,025 a2 = -1

-0,063 a1 + 2,029 a2 + 0,1 = -1

-0,063 a1 + 2,029 a2 = -1,1 .....(5)

Dari persamaan (3) subtitusikan dengan persamaan (4).

3,676 a1 + 0,366 a2 + 4,457 (0,272 – 0,855 a1 – 0,068 a2) = 1

3,676 a1 + 0,366 a2 +1,212 – 3,811 a1 – 0,303 a2 = 1

-0,135 + 0,063 a2 +1,212 = 1

-0,135 a1+ 0,063 a2 = -0,212 ..(6)

Dari persamaan (5) dan (6) dapat diselesaikan dengan metode eliminasi sebagai berikut.

-0,063 a1 + 2,029 a2 = -1,1 x -0,135

-0,135 a1 + 0,063 a2 = -0,212 x 0,063

0,009 a1 – 0,274 a2 = 0,149

0,009 a1 - 0,004 a2 = 0,013 -

-0,27 a2 = 0,136

$$a2 = \frac{0,136}{-0,27}$$

a2 = -0,005

Untuk mendapatkan nilai a_1 substitusikan nilai a_2 ke persamaan (6) sebagai berikut.

-0,135 a1+ 0,063 a2 = -0,212

-0,135 a1 - 0,063 (0,005) = -0,212

-0,135 a1 - 0,0003 = -0,212

-0,135 a1 = -0,212 + 0,0003

-0,135 a1 = -0,2117

$$a1 = \frac{-0,2117}{-0,135}$$

a1 = 1,568

Untuk mendapatkan nilai $a3$ substitusikan nilai $a1$ dan $a2$ ke persamaan (1) sebagai berikut.

$$3{,}144\ a1 + 0{,}250\ a2 + 3{,}676\ a3 = 1$$
$$3{,}144\ (1{,}568) + 0{,}250\ (-0{,}005) + 3{,}676\ a3 = 1$$
$$4{,}930 - 0{,}001 + 3{,}676\ a3 = 1$$
$$4{,}929 + 3{,}676\ a3 = 1$$
$$3676\ a3 = 1 - 4{,}929$$
$$a3 = \frac{-3{,}929}{3{,}676}$$
$$a3 = -1{,}069$$

Sehingga didapatlah nilai untuk $a1$, $a1$, dan $a3$ sebagai berikut berikut.

$$a1 = 1{,}568 \qquad a2 = -0{,}005 \qquad a3 = 1{,}069$$

Karena nilai ai adalah bernilai nol atau positif, maka nilai $a2$ dan $a3$ tidak dihitung ketahap selanjutnya. Setelah didapatkan nilai ai masukkan ke persamaan 2.34 sebagai berikut.

$$W = 1{,}568 \begin{bmatrix} 1{,}069 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 1{,}676 \\ 1{,}568 \\ 1{,}568 \end{bmatrix}$$

Kemudian hasil yang didapatkan melalui perhitungan dengan menggunakan persamaan 2.34 diatas, maka digunakan persamaan 2.35 didapatkan.

$$w = \begin{bmatrix} 1{,}676 \\ 1{,}568 \end{bmatrix}, \text{ dan } b = 1{,}568$$

Maka didapatkanlah nilai *hyperplane* untuk mengklasifikasikan kedua kelas yaitu 1,568.

Langkah selanjutnya adalah tahap pengujian dari SVM. Data diambil dari feature vector gabungan LBP dan HOG. Pada tahap ini data testing di kernelisasi dengan fungsi linear kernel.

Tabel 3.11 Tahap Testing

Data Citra	Feature Vector (x)	Label(y)
T1	0:0.631242 1:0.613269 2:0.241379 3:0.041825 4:0.523147 5:0.347685 6:0.178963 7:0.213741 8:0.596831	1
T2	0:0.137843 1:0.312798 2:0.431698 3:0.021642 4:0.021743 5:0.114591 6:0.054973 7:0.261231 8:0.141985	-1

Tabel 3.12 Tahap Testing (Lanjutan)

Data Citra	Feature Vector (x)	Label(y)
T3	0:0.925129 1:0.752484 2:0.447370 3:0.175951 4:0.445637 5:0.086025 6:0.177341 7:0.276194 8:0.276638	1
$T_{Testing}$	0:0.124143 1:0.009836 2:0.108495 3:0.108495 4:0.011624 5:0.046945 6:0.099851 7:0.061699 8:0.032191	0

Selanjutnya yaitu melakukan kernelisasi menggunakan fungsi kernel *linier*.

$x1$ = [0.631242 0.613269 0.241379 0.041825 0.523147 0.347685 0.178963 0.213741 0.596831]

$x2$ = [0.137843 0.312798 0.431698 0.021642 0.021743 0.114591 0.054973 0.261231 0.141985]

x3 =[0.925129 0.752484 0.447370 0.175951 0.445637 0.086025 0.177341 0.276194 0.276638]

$x4$ = [0.124143 0.009836 0.108495 0.108495 0.011624 0.046945 0.099851 0.061699 0.032191]

$x_1^T x_1$ = [0.631242 0.613269 0.241379 0.041825 0.523147 0.347685 0.178963 0.213741 0.596831] ^{T} *
[0.631242 0.613269 0.241379 0.041825 0.523147 0.347685 0.178963 0.213741 0.596831]
= 0.39847

$x_1^T x_2$ =[0.631242 0.613269 0.241379 0.041825 0.523147 0.347685 0.178963 0.213741 0.596831] ^{T} *
[0.137843 0.312798 0.431698 0.021642 0.021743 0.114591 0.054973 0.261231 0.141985]
= 0.08701

$x_1^T x_3$ = [0.631242 0.613269 0.241379 0.041825 0.523147 0.347685 0.178963 0.213741 0.596831] ^{T} *
[0.925129 0.752484 0.447370 0.175951 0.445637 0.086025 0.177341 0.276194 0.276638]

= 0.58398

$x_1^T x_4$ = [0.631242 0.613269 0.241379 0.041825 0.523147 0.347685 0.178963 0.213741 0.596831] ^{T} *
[0.124143 0.009836 0.108495 0.108495 0.011624 0.046945 0.099851 0.061699 0.032191]
= 0.187796

$x_2^T x_1$ = [0.137843 0.312798 0.431698 0.021642 0.021743 0.114591 0.054973 0.261231 0.141985] ^{T} *
[0.631242 0.613269 0.241379 0.041825 0.523147 0.347685 0.178963 0.213741 0.596831]
= 0,08701

$x_2^T x_2$ = [0.137843 0.312798 0.431698 0.021642 0.021743 0.114591 0.054973 0.261231 0.141985] ^{T} *
[0.137843 0.312798 0.431698 0.021642 0.021743 0.114591 0.054973 0.261231 0.141985]
= 0,01900

$x_2^T x_3$ = [0.137843 0.312798 0.431698 0.021642 0.021743 0.114591 0.054973 0.261231 0.141985] ^{T} *
[0.925129 0.752484 0.447370 0.175951 0.445637 0.086025 0.177341 0.276194 0.276638]
= 0,12752

$x_2^T x_4$ = [0.137843 0.312798 0.431698 0.021642 0.021743 0.114591 0.054973 0.261231 0.141985] ^{T} *
[0.124143 0.009836 0.108495 0.108495 0.011624 0.046945 0.099851 0.061699 0.061699]
= 0.101184

$x_3^T x_1$ = [0.925129 0.752484 0.447370 0.175951 0.445637 0.086025 0.177341 0.276194 0.276638] ^{T} *
[0.631242 0.613269 0.241379 0.041825 0.523147 0.347685 0.178963 0.213741 0.596831]

= 0,58398

$x_3^T x_2$ = [0.925129 0.752484 0.447370 0.175951 0.445637 0.086025 0.177341 0.276194 0.276638] ^{T} *
[0.137843 0.312798 0.431698 0.021642 0.021743 0.114591 0.054973 0.261231 0.141985]
= 0,12752

$x_3^T x_3$ = [0.925129 0.752484 0.447370 0.175951 0.445637 0.086025 0.177341 0.276194 0.276638] ^{T} *
[0.925129 0.752484 0.447370 0.175951 0.445637 0.086025 0.177341 0.276194 0.276638]
= 0,85586

$x_3^T x_4$ = [0.925129 0.752484 0.447370 0.175951 0.445637 0.086025 0.177341 0.276194 0.276638] ^{T} *
[0.124143 0.009836 0.108495 0.108495 0.011624 0.046945 0.099851 0.061699 0.032191]
= 0,242749

$X_4^T x_1$ = [0.124143 0.009836 0.108495 0.108495 0.011624 0.046945 0.099851 0.061699 0.061699] ^{T} *
[0.631242 0.613269 0.241379 0.041825 0.523147 0.347685 0.178963 0.213741 0.596831]
= 0,187796

$X_4^T x_2$ = [0.124143 0.009836 0.108495 0.108495 0.011624 0.046945 0.099851 0.061699 0.032191] ^{T} *
[0.137843 0.312798 0.431698 0.021642 0.021743 0.114591 0.054973 0.261231 0.141985]
= 0,101184

$X_4^T x_3$ = [0.124143 0.009836 0.108495 0.108495 0.011624 0.046945 0.099851 0.061699 0.032191] ^{T} *
[0.925129 0.752484 0.447370 0.175951 0.445637 0.086025 0.177341 0.276194 0.276638]

= 0,242749

$X_4{}^T x_4$ = [0.124143 0.009836 0.108495 0.108495 0.011624 0.046945 0.099851 0.061699 0.032191]^T *
[0.124143 0.009836 0.108495 0.108495 0.011624 0.046945 0.099851 0.061699 0.032191]
= 0.056203

Setelah dilakukan perhitungan pada seluruh nilai x pada data testing, maka matriks yang terbentuk dari hasil perhitungan adalah sebagai berikut.

$$= \begin{bmatrix} 0.39847 & 0.08701 & 0.58398 & 0{,}187796 \\ 0{,}08701 & 0{,}01900 & 0{,}12752 & 0{,}101184 \\ 0{,}58398 & 0{,}12752 & 0{,}85586 & 0.242748 \\ 0{,}187796 & 0{,}101184 & 0.242748 & 0{,}056203 \end{bmatrix}$$

Kemudian tahap selanjutnya adalah melakukan perhitungan terhadap y. Untuk nilai y adalah nilai dari label yang diberikan.

$y_1{}^T y_1 = 1$

$y_1{}^T y_2 = -1$

$y_1{}^T y_3 = 1$

$y_1{}^T y_4 = 0$

$y_2{}^T y_1 = -1$

$y_2{}^T y_2 = 1$

$y_2{}^T y_3 = -1$

$y_2{}^T y_4 = 0$

$y_3{}^T y_1 = 1$

$y_3{}^T y_2 = -1$

$y_3{}^T y_3 = 1$

$$y_3^T y_3 = 0$$

$$y_4^T y_1 = 0$$

$$y_4^T y_2 = 0$$

$$y_4^T y_3 = 0$$

$$y_4^T y_4 = 0$$

Sehingga matriks yang terbentuk dari hasil perhitungan $yi^T yj$ hasilnya adalah sebagai berikut :

$$\begin{bmatrix} 1 & -1 & 1 & 0 \\ -1 & 1 & -1 & 0 \\ 1 & -1 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

Tahap selanjutnya mengubah setiap *training* menjadi nilai vektor (support vector) = $\binom{x}{y}$ agar mendapatkan nilai ai. Nilai x didapatkan menggunakan persamaan 2.31 kernel *linear* sebagai berikut.

$X_{T1} = x_1^T x_1 + x_1^T x_2 + x_1^T x_3 + x_1^T x_4 = 0.398466 + 0.087012 + 0.58398 + 0.187796$
$= 1.257254$

$X_{T2} = x_2^T x_1 + x_2^T x_2 + x_2^T x_3 + x_2^T x_4 = 0.087012 + 0.19001 + 0.127523 + 0.101184$
$= 0.505729$

$X_{T3} = x_3^T x_1 + x_3^T x_2 + x_3^T x_3 + x_3^T x_4 = 0.58398 + 0.127523 + 0.855864 + 0.242749$
$= 1.810116$

$X_{T4} = x_4^T x_1 + x_4^T x_2 + x_4^T x_3 + x_4^T x_4 = 0.404437 + 0.088316 + 0.59273 + 0.056203$
$= 1.141686$

Sehingga didapatkan nilai x untuk setiap citra sebagai berikut.

Tabel 3.13 Nilai X Pada Setiap Data

Testing	T1	T2	T3	$T_{Testing}$
X	1.257254	0.505729	1.810116	1.141686

Begitu juga dengan nilai y didapatkan menggunakan persamaan 2.32 kernel *linear* untuk y yaitu sebagai berikut.

$$Y_{T1} = y_1^T y_1 + {}_{y1}{}^T y_2 + y_1^T y_3 + y_1^T y_4 = 1 - 1 + 1 + 0 = 1$$

$$Y_{T2} = y_2^T y_1 + y_2^T y_2 + y_2^T y_3 + y_2^T y_4 = -1 + 1 - 1 + 0 = -1$$

$$Y_{T3} = y_3^T y_1 + x_3^T y_2 + y_3^T y_3 + y_3^T y_4 = 1 - 1 + 1 + 0 = 1$$

$$Y_{Testing} = y_4^T y_1 + x_4^T y_2 + y_4^T y_3 + y_4^T y_4 = 0 + 0 + 0 + 0 = 0$$

Sehingga didapatkan nilai y untuk setiap citra sebagai berikut.

Tabel 3.14 Nilai Y Pada Setiap Data

Testing	T1	T2	T3	$T_{Testing}$
Y	1	-1	1	0

Seteleh nilai x dan y didapatkan, maka untuk nilai x dan y dari *citra testing* di subtitusikan ke dalam persamaan 2.30 sebagai berikut.

$$\sqrt{x_n^2 + y_n^2} = \sqrt{1.141686^2 + 0^2} = 1.141686$$

Karena $\sqrt{x_n^2 + y_n^2} < 2$, maka $\varphi\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} x \\ y \end{bmatrix}$

$$\varphi \mathrm{T}_{\mathrm{esting}} = \begin{bmatrix} 1.141686 \\ 0 \end{bmatrix}$$

Selanjutnya untuk mendapatkan nilai kelas dari *citra testing*, lakukan perhitungan tersebut ke persamaan berikut, dengan nilai w^T didapatkan dari hasil *training*.

$$\mathrm{w}^{\mathrm{T}} \cdot \varphi_{\mathrm{Testing}} = [1.676 \quad 1.568] \begin{bmatrix} 1.141686 \\ 0 \end{bmatrix} = 1.91346$$

Tabel 3.15 Bobot Support Vector

$\mathbf{w^T \cdot \varphi_{Testing}}$	Kelas
$wT.\varphi T_{Testing} \leq 1.568$	-1 Bukan Manusia
$wT.\varphi TTesting \geq 1.568$	+1 (Manusia)

Hasil *testing* adalah 1.91346 sehingga $T_{Testing}$ lebih besar dari pada hasil nilai b pada tahap *training* yaitu 1.568 yang berarti $T_{Testing}$ termasuk dalam kelas 1 (manusia).

Dalam penelitian ini, tahapan *learning* dan pengujian tersebut dilakukan dengan menggunakan *library* LIBSVM dari Matlab, adapun penggunaannya yaitu yaitu sebagai berikut.

Tabel 3.16 Library SVM

No	Source Code
1	% Mengambil semua citra learning pada direktori
2	posFiles = getImagesInDir('./Images/Training/Positive/', true);
3	negFiles = getImagesInDir('./Images/Training/Negative/', true);
4	% Memberi label pada setiap citra.
5	y_train = [ones(length(posFiles), 1); zeros(length(negFiles), 1)];
6	% Menggambungkan semua gambar learning dalam satu file list
7	fileList = [posFiles, negFiles];
8	% Mengambil matriks hasil feature vector gabungan
9	X_train = zeros(length(fileList), (3780+256));
10	fprintf('Computing descriptors for %d training windows: ', length(fileList));
11	fprintf('Computing descriptors for %d training windows: ', length(fileList));
12	%progress bar
13	h = waitbar(0,'Please wait...');
14	h = waitbar(0,'Please wait...');
15	for i = 1 : length(fileList)
16	waitbar(i/length(fileList),h);
17	% Get the next filename.
18	imgFile = char(fileList(i));
19	% Cetak Iterasi
20	printIteration(i);
21	%fprintf('%s\n', imgFile);
22	% Load the image into a matrix.
23	img = imread(imgFile);
24	% Calculate the HOG descriptor for the window.
25	H = getHOGDescriptor(hog, img);
26	[output lbpFeature]=lbp_demo(img);
27	lbpFeature = lbpFeature/max(lbpFeature);
28	H = [H; lbpFeature'];
29	X_train(i, :) = H';
30	end
31	close(h);
32	fprintf('\n');
33	% Train the SVM.
34	% Store the resulting weights, theta, in the 'hog' structure.
35	fprintf('\nTraining linear SVM classifier...\n');
36	hog.theta = train_svm(X_train, y_train, 1.0);
37	save('hog_model.mat', 'hog');
38	% Evaluate the SVM over the training data.
39	p = X_train * hog.theta;
40	% Recognize as a pedestrian if the confidence is over 0.
41	numRight = sum((p > 0) == y_train);

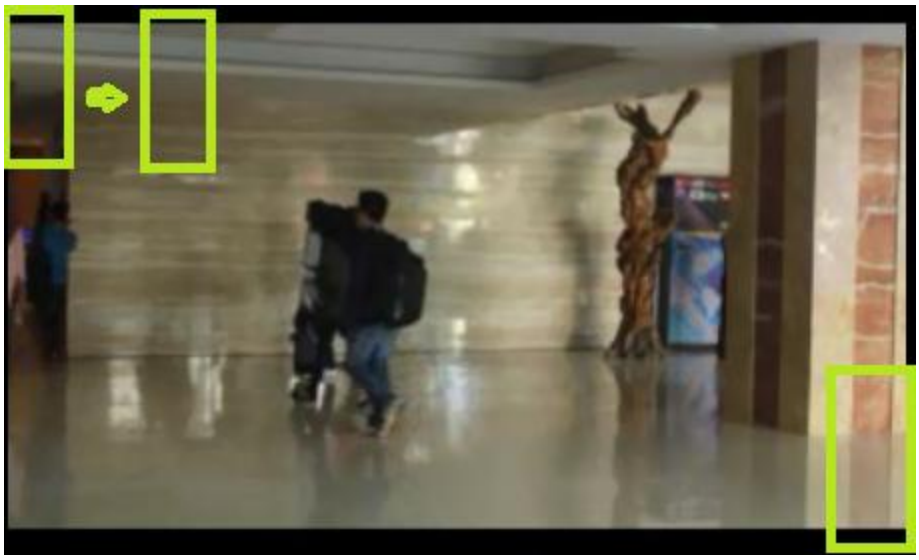
Tabel 3.17 Library SVM (Lanjutan)

No	Source Code
42	fprintf('\nTraining accuracy: (%d / %d) %.2f%%\n', numRight,length(y_train), numRight / length(y_train) * 100.0);
43	function theta = train_svm(trainXC, trainY, C)
44	numClasses = max(trainY);
45	w0 = zeros(size(trainXC,2)*numClasses, 1);
46	w = minFunc(@my_l2svmloss, w0, struct('MaxIter', 1000, 'MaxFunEvals', 1000), ...
47	trainXC, trainY, numClasses, C)
48	theta = reshape(w, size(trainXC,2), numClasses);
49	function [loss, g] = my_l2svmloss(w, X, y, K, C)
50	[M,N] = size(X);
51	theta = reshape(w, N,K);
52	Y = bsxfun(@(y,ypos) 2*(y==ypos)-1, y, 1:K);
53	margin = max(0, 1 - Y .* (X*theta));
54	loss = (0.5 * sum(theta.^2)) + C*mean(margin.^2);
55	loss = sum(loss);
56	g = theta - 2*C/M * (X' * (margin .* Y));
57	% adjust for intercept term
58	loss = loss - 0.5 * sum(theta(end,:).^2);
59	g(end,:) = g(end, :) - theta(end,:);
60	g = g(:);

3.1.3.4 Sliding Windows

Dalam tahapan perhitungan manusia yaitu pada proses sliding window berukuran 64 x 128 menghasilkan feature vektor yang telah dikernelisasi dan dilakukan pengujian sehingga menghasilkan nilai +1 maka akan diberi kotak bahwa dalam jendela berukuran 64 x 128 tersebut adalah manusia.

Berikut pada Gambar 3.13 adalah proses dari sliding windows.



Gambar 3.13 Sliding Windows Untuk Pendeteksian

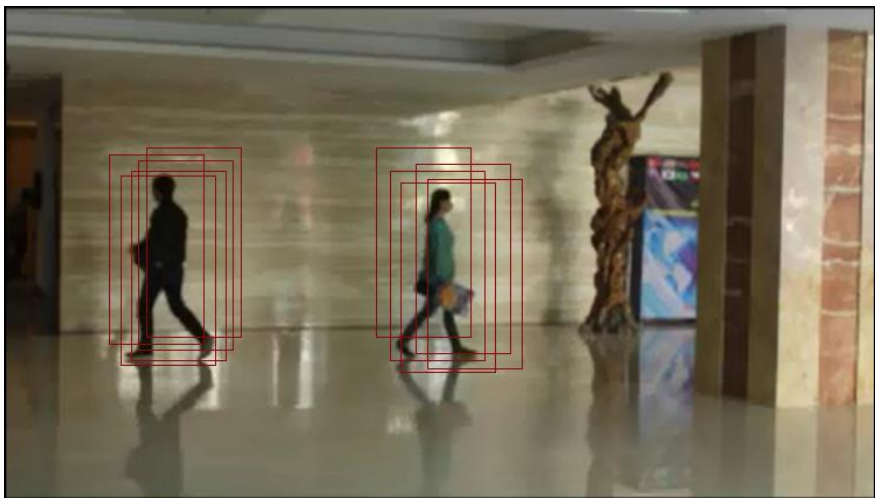
Berikut adalah pseudocode dari sliding windows:

```
Procedure    SlidingWindows(input    Frame    LebarFrame,
TinggiFrame : integer)
Deklarasi :
     JendelaTinggi, JendelaLebar, jendela : integer;
Algoritma :
     JendelaLebar ← 64;
     JendelaTinggi ← 128;
     for j ← 1 to TinggiFrame – JendelaTinggi +1
        for i ← 1 to LebarFrame – JendelaLebar +1
          jendela ← frame(j to j + jendelaTinggi -1, i to
i + JendelaLebar - 1);
          HOG(jendela);
          LBP(jendela);
        Endfor
     Endfor
EndeProcedure
```

Gambar 3.14 Pseudo Code Sliding WIndows

3.1.3.5 Non-Maximum Supression

Dikarenakan pada seuatu objek manusia terdapat lebih dari beberapa kotak maka harus dinormalisasi dengan Non-Maximum Supression. Berikut pada Gambar 3.15 adalah konsep dari Non-Maximum Supression.



Gambar 3.15 Kotak Klasifikasi Manusia

Gambar 3.15 adalah hasil dari klasifikasi manusia, pada gambar tersebut didapatkan lebih dari satu kotak pada satu objek manusia. Langkah selanjutnya adalah pengambilan nilai dengan kotak klasifikasi tertinggi dari setiap kotak-kotak deteksi.



Gambar 3.16 Pengambilan Kotak Dengan Nilai Tertinggi

Setelah didapatkan nilai kotak dengan klasifikasi tertinggi dari semua kotak deteksi maka kotak yang saling bertumpuk dieliminasi. Berikut adalah hasil akhir dari proses normalisasi Non-Maximum Supression.



Gambar 3.17 Eliminasi Kotak yang Bertumpuk

Untuk menghitung jumlah manusia pada sebuah video maka dilakukan perhitungan padasetiap kotak yang telah dinormalisasi dengan Non-Maxima Supression.

3.2 Analisis Kebutuhan Non-Fungsional

Analisis kebutuhan nonfungsional bertujuan untuk mengetahui kebutuhan sistem yang dibuat dalam pengujian metode deteksi manusia. Analisis yang dilakukan meliputi analisis pada kebutuhan perangkat keras sistem dan kebutuhan perangkat lunak sistem.

3.2.1 Analisis Kebutuhan Perangkat Keras

Dalam pembangunan dan untuk menjalankan perangkat lunak pengujian metode HOG-LBP dan SVM dibutuhkan seperangkat komputer. Pada saat membangun perangkat lunak, komputer yang digunakan memiliki kriteria seperti pada tabel 3.18.

Tabel 3.18 Perangkat Keras Saat Pembangunan

No.	Perangkat Keras	Spesifikasi
1.	Processor	Processor AMD A6-5400 K @3.6 Ghz
2	RAM	RAM 4 GB
3.	HDD	Hardisk 320GB

3.2.2 Analisis Kebutuhan Perangkat Lunak

Terdapat beberapa perangkat lunak yang digunakan saat pembangunan perangkat lunak deteksi dan perhitungan pada manusia dengan metode HOG-LBP dan SVM seperti pada tabel 3.19.

Tabel 3.19 Perangkat Lunak Saat Pembangunan

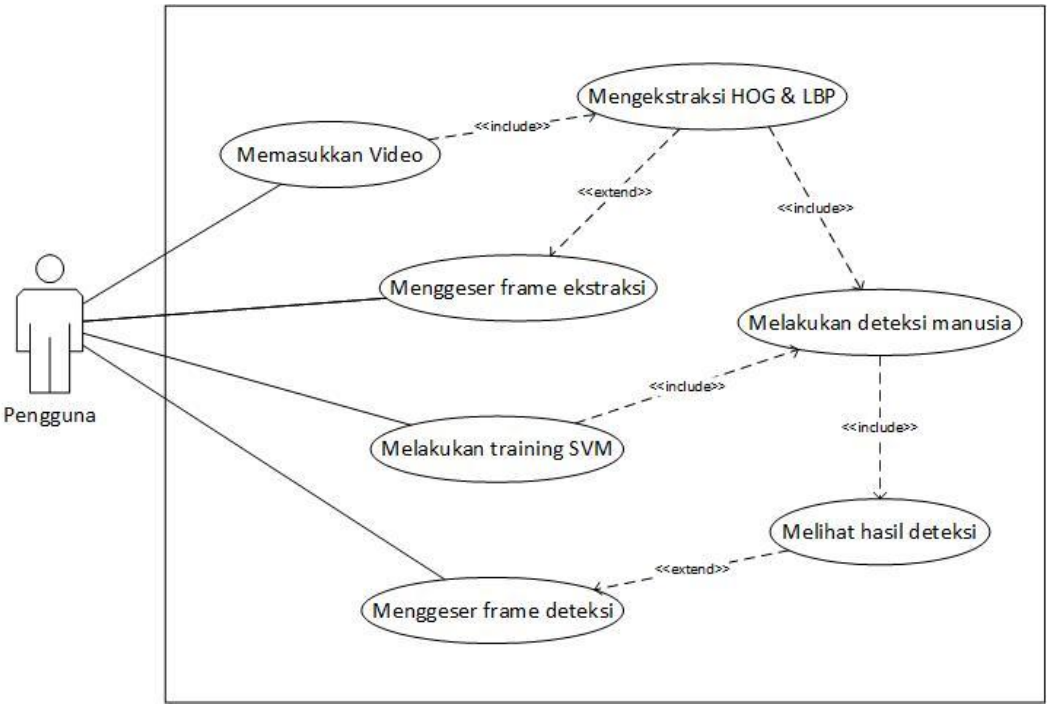
No.	Perangkat Lunak	Keterangan
1.	Sistem Operasi	Windows 10
2	*Integrated Development Environtment* (IDE)	Matlab R2016a
3.	Pemodelan	Microsoft Visio 2016

3.3 Analisis Kebutuhan Fungsional

Analisis dan pemodelan pada proses dan kebutuhan perangkat lunak diperlukan agar perangkat lunak terdefinisi dengan baik. Pemodelan perangkat lunak pengujian metode HOG - LBP akan dibuat dengan diagram UML mengunakan Microsoft Visio 2016.

3.3.1 Diagram Use Case

Pada sistem yang dibangun saat ini terdapat satu actor yaitu pengguna. Interaksi antara pengguna dengan pendeteksian dan perhitungan manusia akan digambarkan pada diagram usecase pada Gambar 3.18.



Gambar 3.18 Diagram Use Case

Diagram *use case* pada Gambar 3.18 terdapat seorang *actor* dan 3 definisi *use case* dengan penjelasan sebagai berikut:

1. Actor / Aktor

Aktor adalah pihak yang dapat mengkases use case dalam sistem. Aktor dalam sistem ini disebut dengan Pengguna.

2. Definisi Use Case

Fungsi-fungsi yang dimiliki oleh sistem digambarkan dalam *use case.* Definisi dari use case ada pada tabel 3.20.

Tabel 3.20 Definisi Use Case

No	Use Case	Definisi
1.	Memasukkan Video	Memilih video yang akan dilakukan pengujian.
2.	Melakukan *Training* SVM	Melakukan training gambar positif dan negative yang berasal dari dataset.
3.	Mengekstraksi HOG & LBP	Melakukan ekstraksi frame pada video dan menampilkan hasil ekstraksi HOG-LBP.
4.	Melakukan Deteksi Manusia	Melakukan proses ekstraksi HOG & LBP pada frame yang telah di ekstraksi dan mengklasifikasi objek tersebut manusia atau bukan.
5.	Melihat Hasil Deteksi	Melihat semua frame yang telah dideteksi.

Tabel 3.21 Definisi Use Case (Lanjutan)

No	Use Case	Definisi
6.	Menggeser Frame Ekstraksi	Melihat frame-frame yang telah diekstraksi dari video
7.	Menggeser Frame Deteksi	Melihat frame-frame yang telah dideteksi.

1. Skenario Use Case

Skenario *use case* merupakan hasil instansiasi dari setiap *use case*. Berikut adalah skenario use case :

Tabel 3.22 Skenario Use Case

No.	1
Use Case	Memasukkan Video
Tujuan	Untuk meimilih video yang akan dideteksi
Aktor	Pengguna
Kondisi Awal	Pengguna berada pada tampilan utama sistem
Kondisi Akhir	Memutar video yang akan dideteksi
Skenario Utama	
1.	Pengguna menekan button load video
2.	Sistem menampilkan direktori.
3.	Pengguna memilih video.
4	Sistem menampilkan video.
No.	2
Use Case	Melakukan *Training* SVM
Tujuan	Untuk melatih sistem agar dapat menyimpan data dari vector manusia.
Aktor	Pengguna
Kondisi Awal	Pengguna berada pada tampilan utama sistem
Kondisi Akhir	Sistem memberi pesan bahwa dataset telah dilakukan pelatihan.
Skenario Utama	
1.	Pengguna menekan button training SVM detector.
2.	Sistem memilih data pada direktori.
3.	Sistem mengkonversi gambar menggunakan Metode HOG dan LBP.
4.	Sistem melakukan learning SVM pada feature vector untuk menghasilkan model vector klasifikasi manusia.
5.	Sistem memberi pesan bahwa training telah dilakukan.
No.	3
Use Case	Mengekstraksi HOG & LBP
Tujuan	Untuk memecah frame pada video dan menampilkan hasil ekstraksi pada semua frame video.
Aktor	Pengguna
Kondisi Awal	Pengguna berada pada tampilan utama sistem
Kondisi Akhir	Sistem memberi pesan bahwa ekstraksi telah dilakukan.
Skenario Utama	
1.	Pengguna menekan button HOG & LBP Extraction
2.	Sistem melakukan ekstraksi frame pada video.
Skenario Utama	
3.	Sistem melakukan ekstraksi HOG & LBP.
4.	Sistem menampilkan hasil ekstraksi HOG & LBP

Tabel 3.23 Skenario Use Case (Lanjutan)

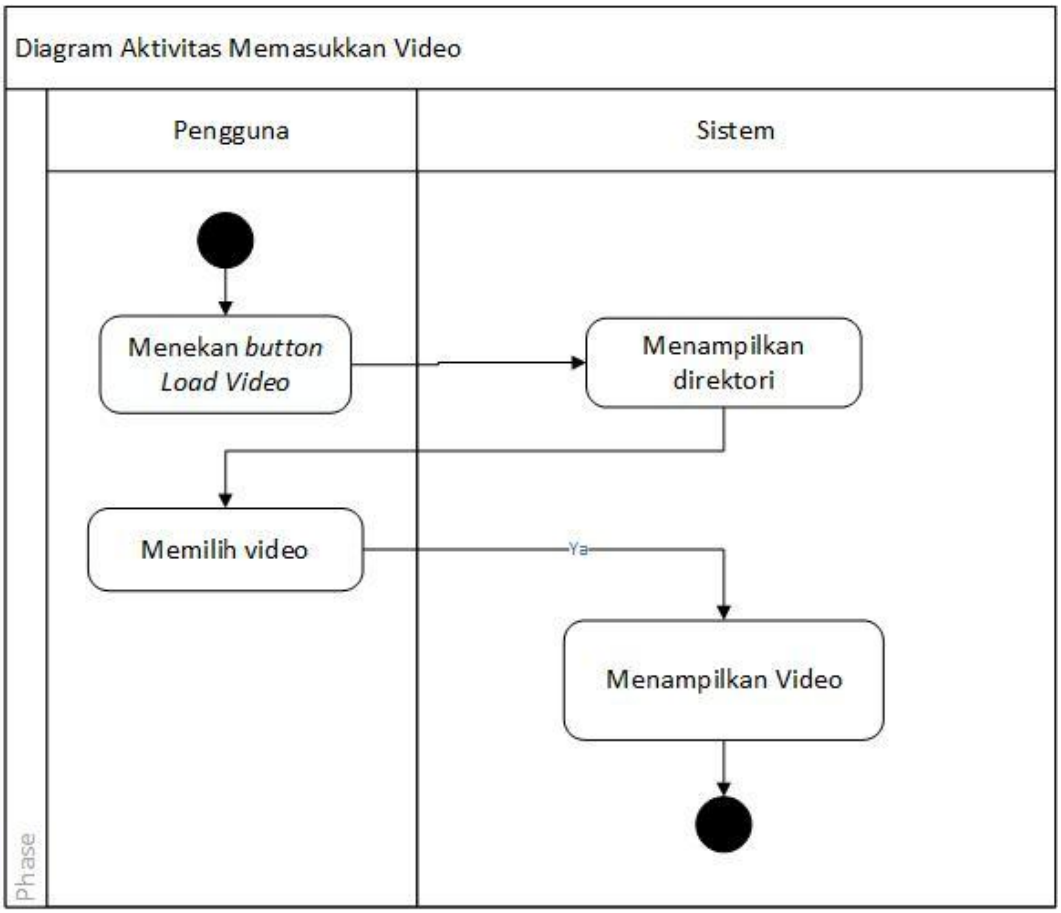
No.		4
Use Case		Melakukan Deteksi Manusia
Tujuan		Untuk mendeteksi dan menghitung manusia yang ada pada video
Aktor		Pengguna
Kondisi Awal		Pengguna berada pada tampilan utama sistem
Kondisi Akhir		Menampilkan pesan bahwa deteksi telah dilakukan
Skenario Utama		
1.	Pengguna menekan button tracking people.	
2.	Sistem melakukan sliding windows pada frame dengan ukuran piksel 64 x 128.	
3.	Sistem melakukan Ekstraksi LBP dan mendapatkan nilai *feature vetctor*.	
4.	Sistem melakukan Ekstraksi HOG dan mendapatkan *feature vector*.	
5.	Sistem menggabungkan feature vector.	
6.	Sistem melakukan kernelisasi dengan SVM.	
7.	Sistem melakukan klasifikasi apakah objek tersebut manusia atau bukan.	
No.		5
Use Case		Melihat Hasil Deteksi
Tujuan		Untuk melihat hasil deteksi pada semua frame video.
Aktor		Pengguna
Kondisi Awal		Pengguna berada pada tampilan utama sistem
Kondisi Akhir		Menampilkan dan memutar video dengan deteksi manusia yang ada pada video
Skenario Utama		
1.	Pengguna menekan button play with video.	
2.	Sistem menampilkan dan memutar deteksi manusia.	
No.		6
Use Case		Menggeser Frame Ekstraksi
Tujuan		Untuk melihat salah satu frame dengan hasil ekstraksi HOG & LBP
Aktor		Pengguna
Kondisi Awal		Pengguna berada pada tampilan utama sistem
Kondisi Akhir		Menampilkan frame dengan ekstraksi HOG & LBP
Skenario Utama		
1.	Pengguna menggeser *slider* pada bagian ekstraksi fitur.	
2.	Sistem menampilkan frame dengan ekstraksi HOG-LBP.	
No.		7
Use Case		Menggeser Frame Deteksi
Tujuan		Untuk melihat salah satu frame dengan deteksi manusia.
Aktor		Pengguna
Kondisi Awal		Pengguna berada pada tampilan utama
Kondisi Akhir		Menampilkan frame dengan deteksi manusia
Skenario Utama		
1.	Pengguna menggeser *slider* pada bagian deteksi manusia.	
2.	Sistem menampilkan frame dengan deteksi manusia.	

3.3.2 Diagram Aktivitas

Diagram aktivitas (*Activity Diagram*) memodelkan alur kerja (*workflow*) sebuah proses bisnis dan urutan aktivitas dalam suatu proses. Adapun rincian dari activity diagram sistem dapat dilihat pada gambar 3.19 sampai gambar 3.25.

3.3.2.1 Diagram Aktivitas Memasukkan Video

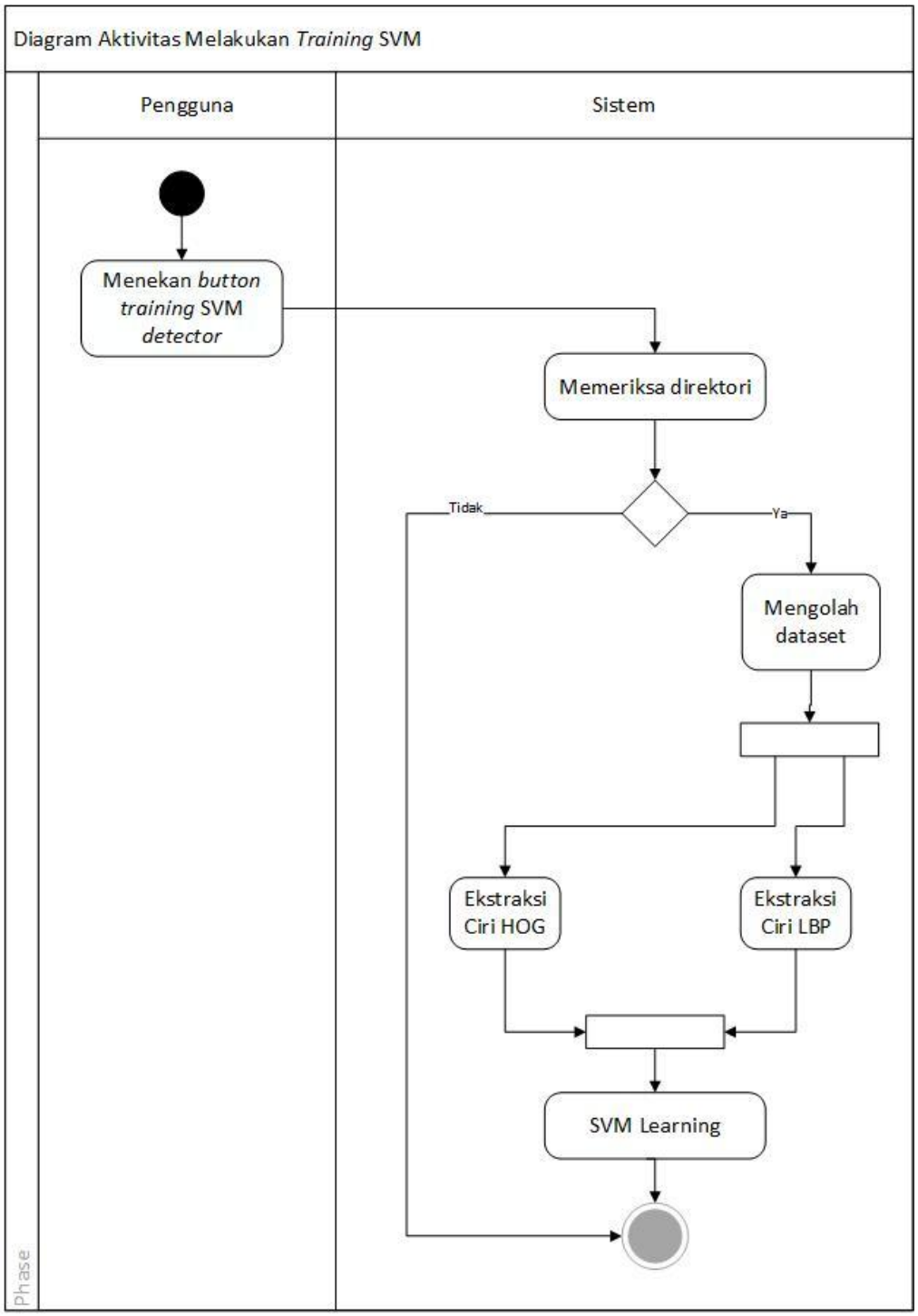
Pada gambar 3.19 adalah diagram aktivitas memasukkan video.



Gambar 3.19 Diagram Aktivitas Memasukkan Video

3.3.2.2 Diagram Aktivitas Melakukan Training SVM

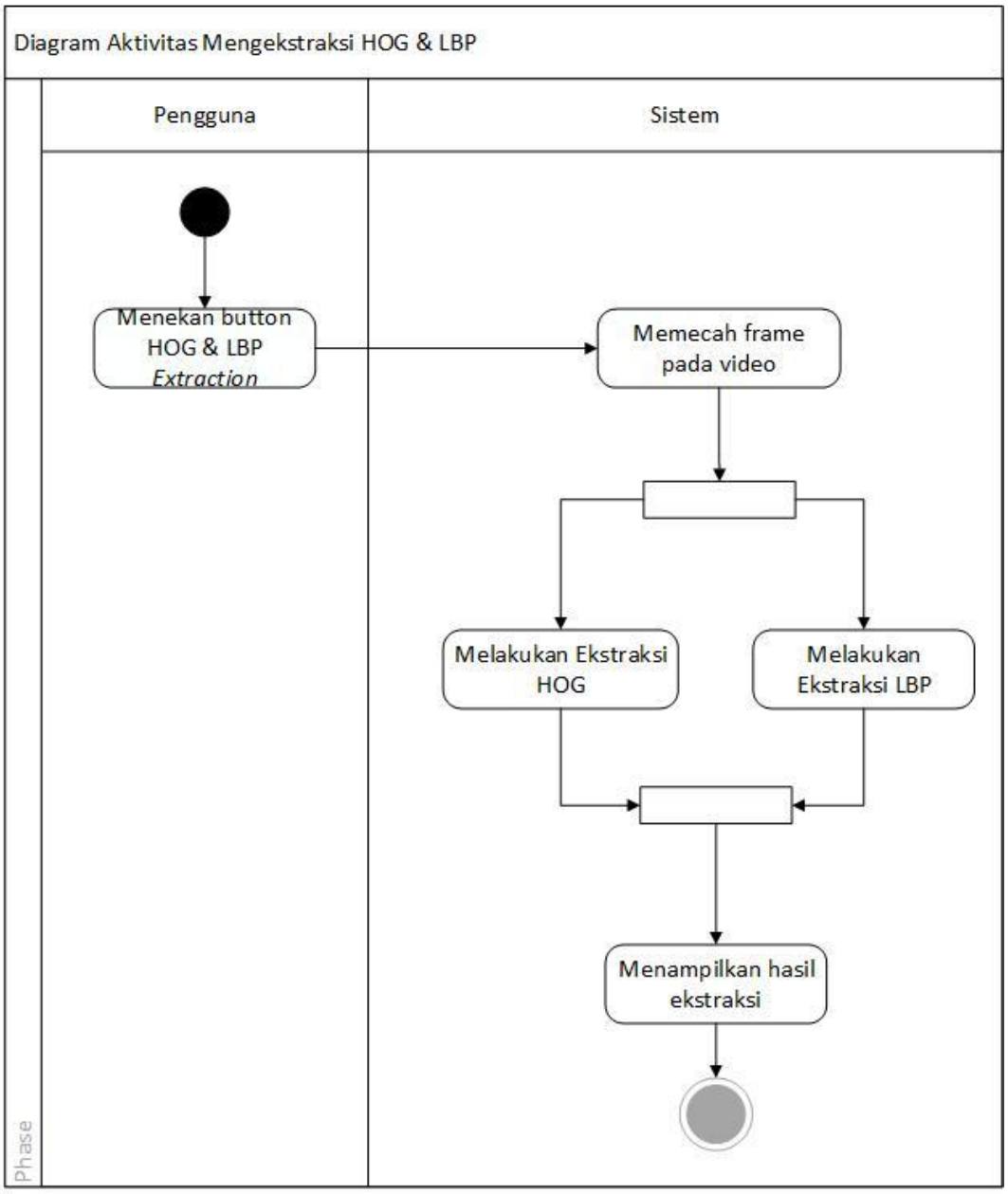
Pada Gambar 3.20 adalah diagram aktivitas melakukan training SVM.



Gambar 3.20 Diagram Aktivitas Melakukan Training SVM

3.3.2.3 Diagram Aktivitas Mengekstraksi HOG & LBP

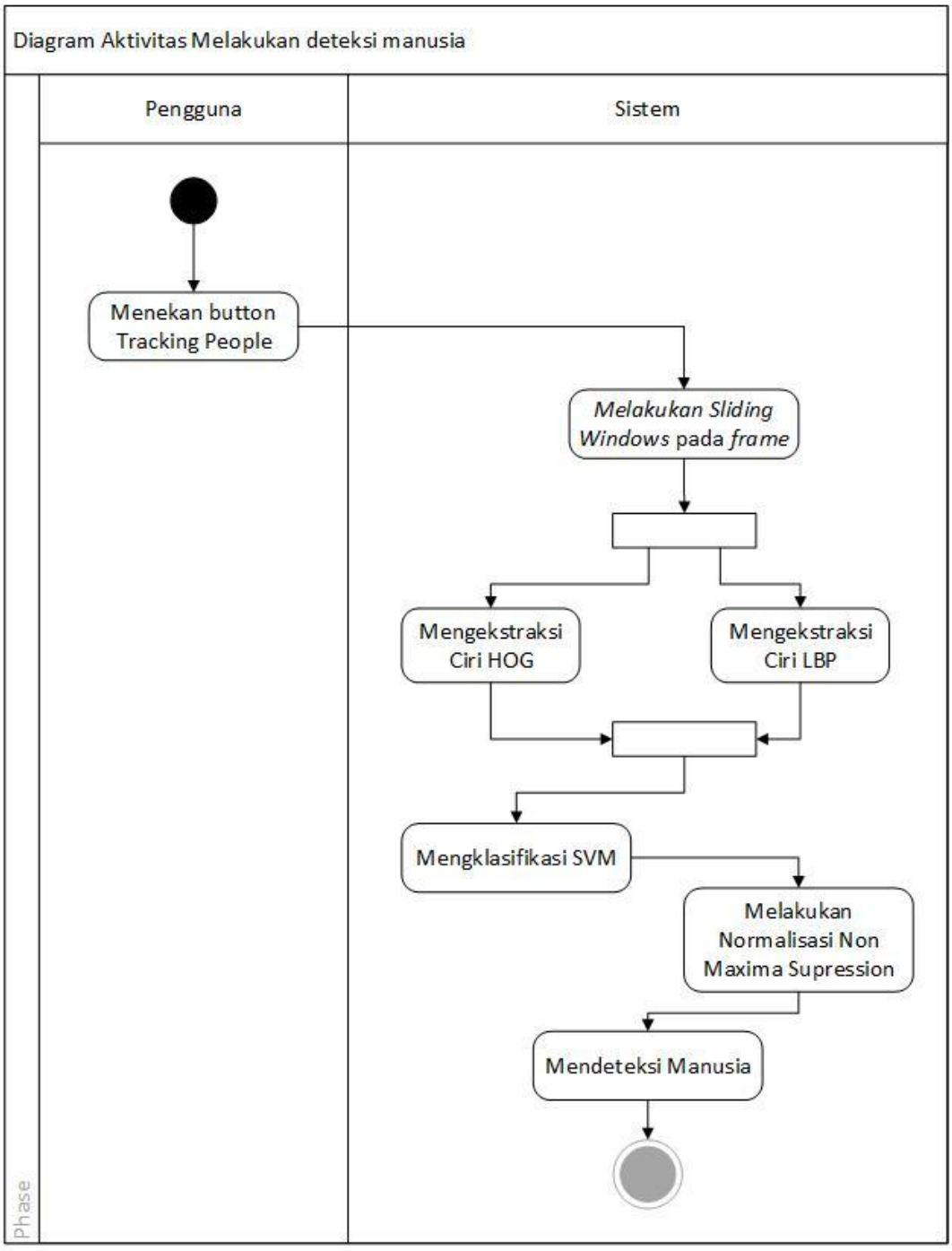
Pada Gambar 3.21 adalah diagram aktivitas mengwkstraksi HOG & LBP.



Gambar 3.21 Diagram Aktivitas Mengekstraksi HOG & LBP

3.3.2.4 Diagram Aktivitas Melakukan Deteksi Manusia

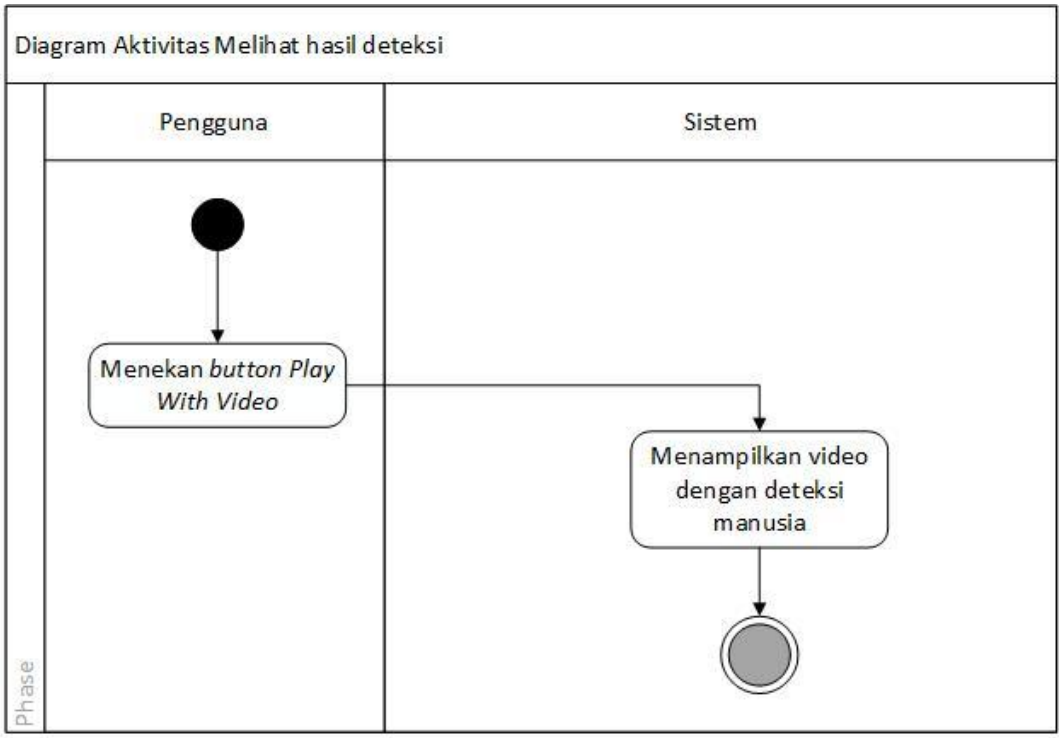
Pada Gambar 3.22 adalah diagram aktivitas melakukan deteksi manusia.



Gambar 3.22 Diagram Aktivitas Melakukan Deteksi Manusia

3.3.2.5 Diagram Aktvitas Melihat Hasil Deteksi

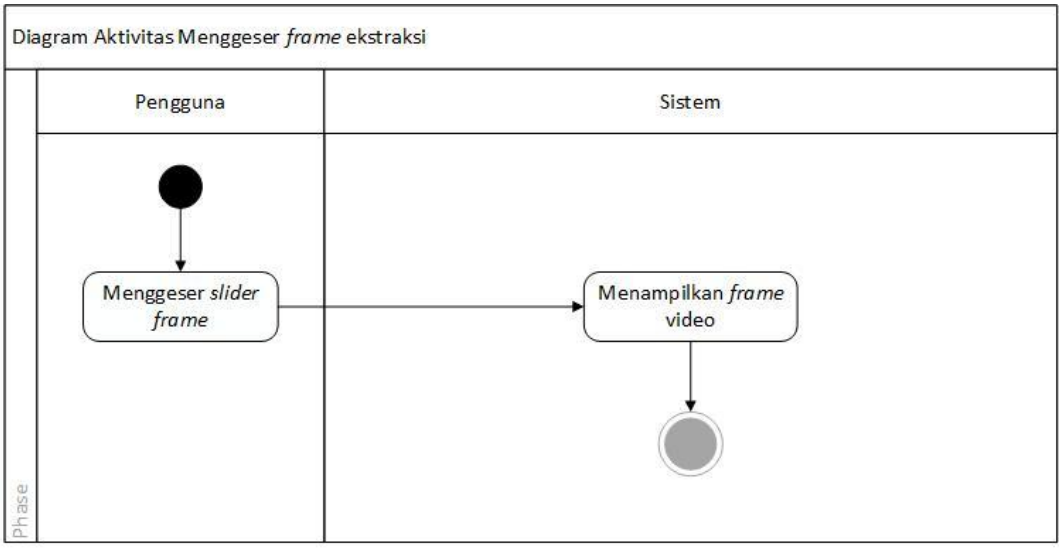
Pada Gambar 3.23 adalah diagram aktivitas melihat hasil deteksi.



Gambar 3.23 Diagram Aktivitas Melihat Hasil Deteksi

3.3.2.6 Diagram Aktivitas Menggeser Frame Ekstraksi

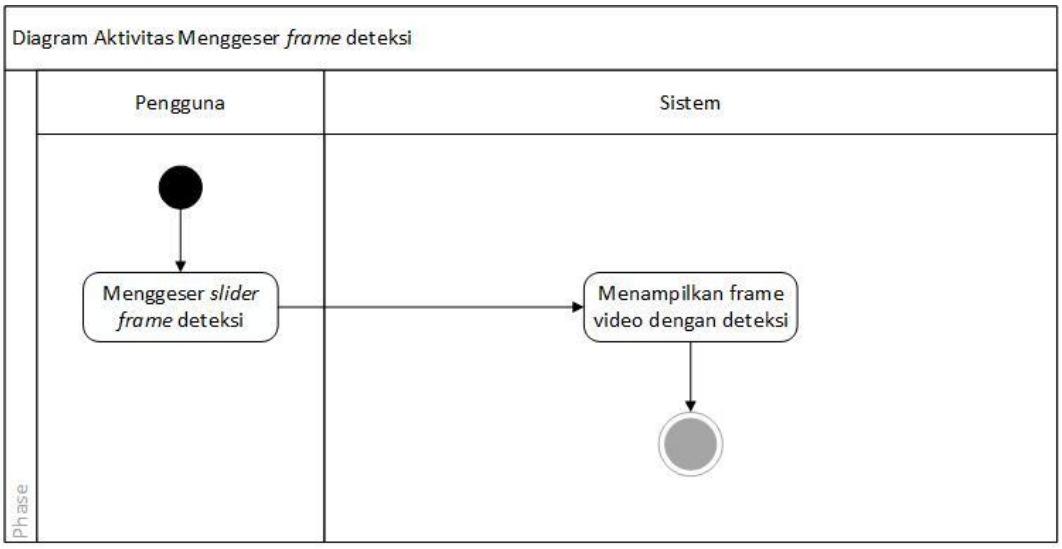
Pada Gambar 3.24 adalah diagram aktivitas menggeser *frame ekstraksi.*



Gambar 3.24 Diagram Aktivitas Menggeser Frame Ekstraksi

3.3.2.7 Diagram Aktivitas Menggeser Frame Deteksi

Pada Gambar 3.25 adalah diagram aktivitas menggeser *frame* deteksi.



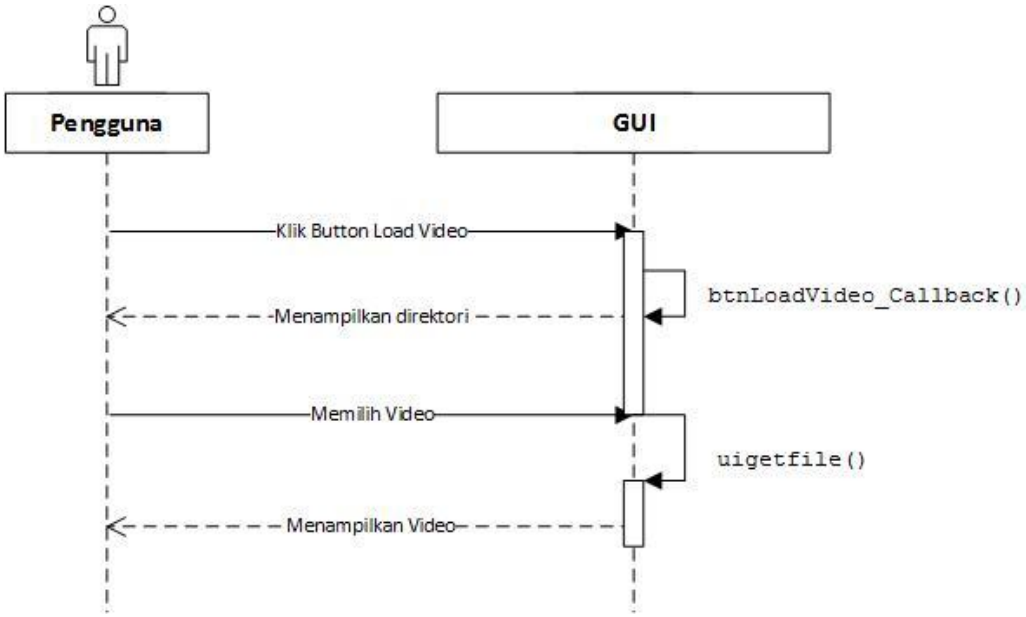
Gambar 3.25 Diagram Aktivitas Menggeser Frame Deteksi

3.3.3 Diagram Sequence

Sequence diagram menggambarkan interaksi antara sejumlah objek dalam urutan waktu. Adapun rincian dari sequence diagram sistem dapat dilihat pada Gambar 3.26 sampai Gambar 3.32.

3.3.3.1 Diagram Sequence Memasukkan Video

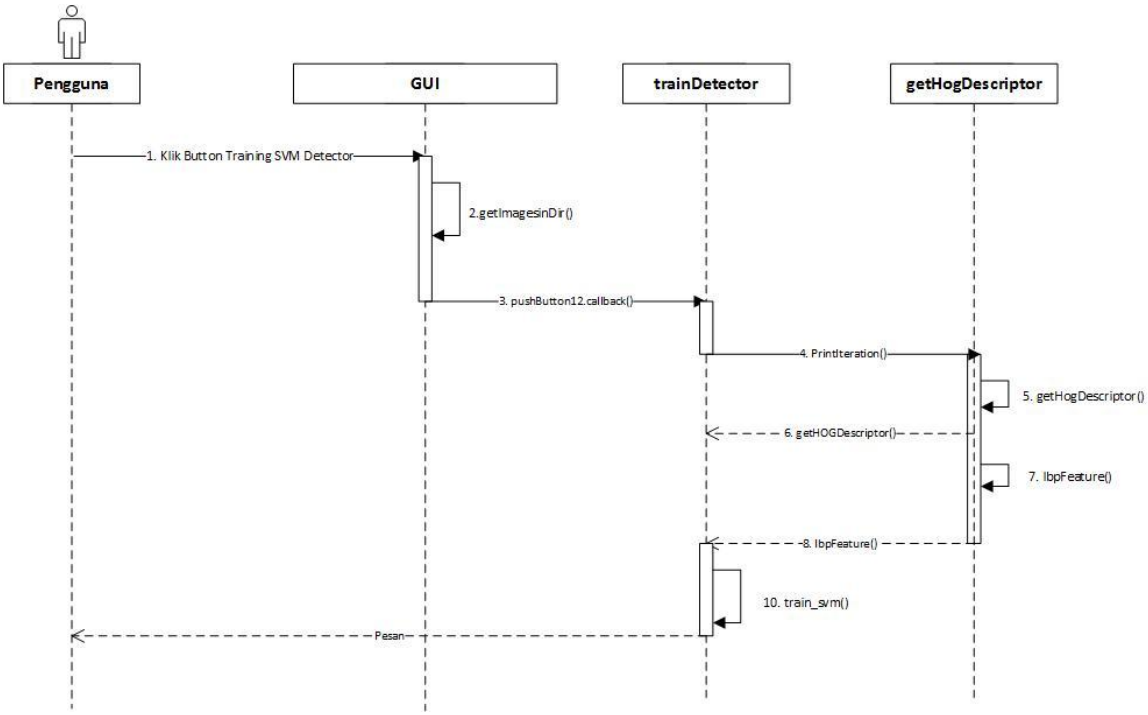
Pada Gambar 3.26 adalah diagram sequence memasukkan video.



Gambar 3.26 Diagram Sequence MemasukkanVideo

3.3.3.2 Diagram Sequence Melakukan Training SVM

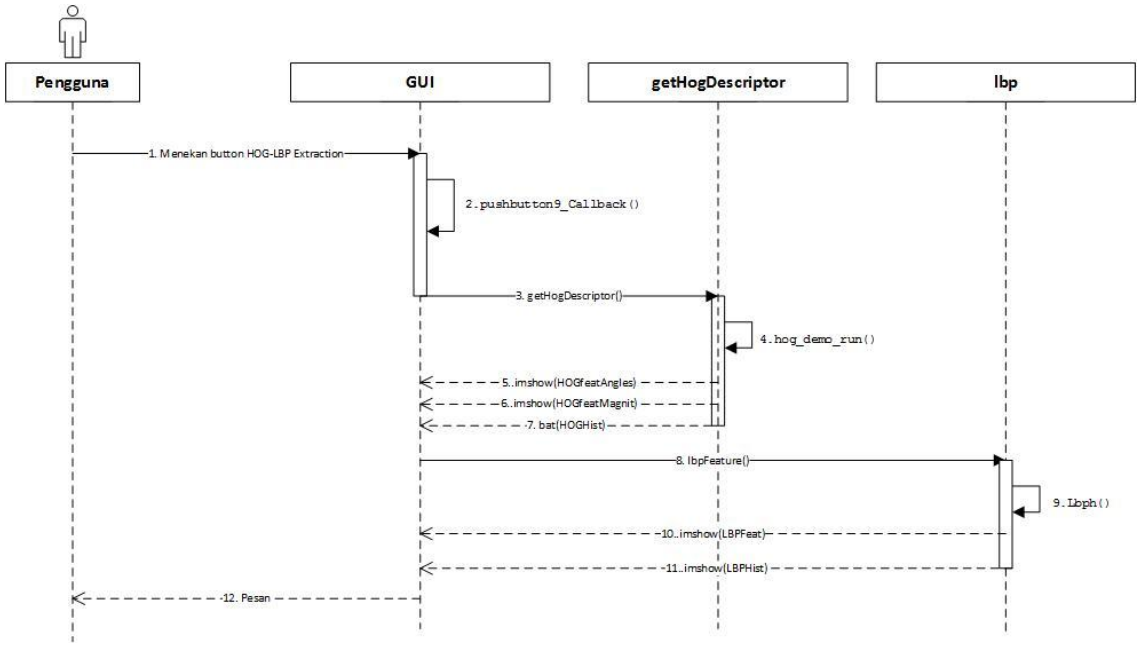
Pada Gambar 3.27 adalah diagram sequence melakukan training SVM.



Gambar 3.27 Diagram Sequence Melakukan Training SVM

3.3.3.3 Diagram Sequence Mengekstraksi HOG & LBP

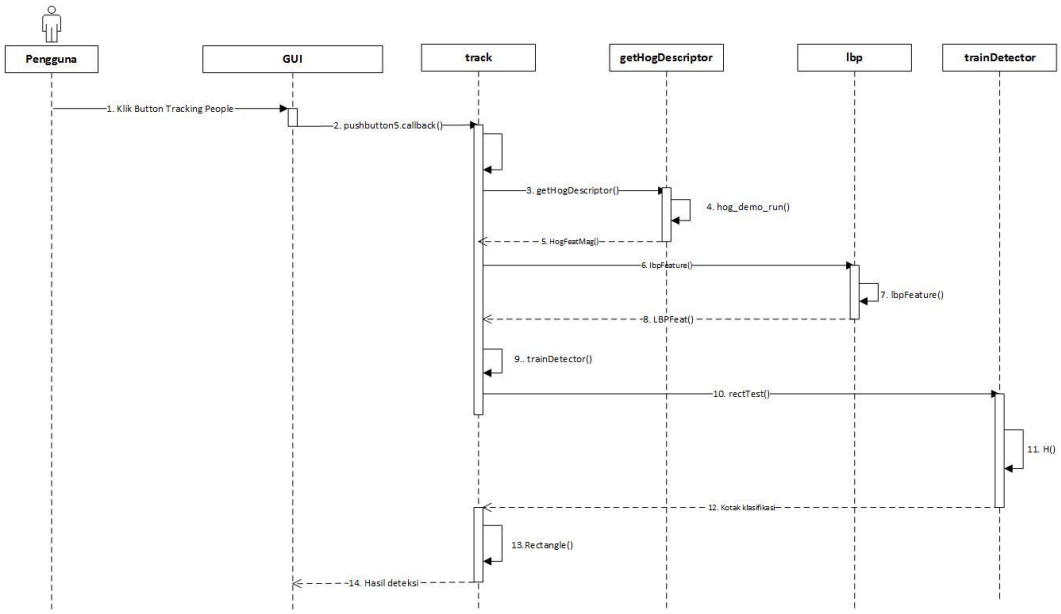
Pada Gambar 3.28 adalah diagram sequence mengesktraksi HOG & LBP.



Gambar 3.28 Diagram Sequence Mengekstraksi HOG & LBP

3.3.3.4 Diagram Sequence Melakukan Deteksi Manusia

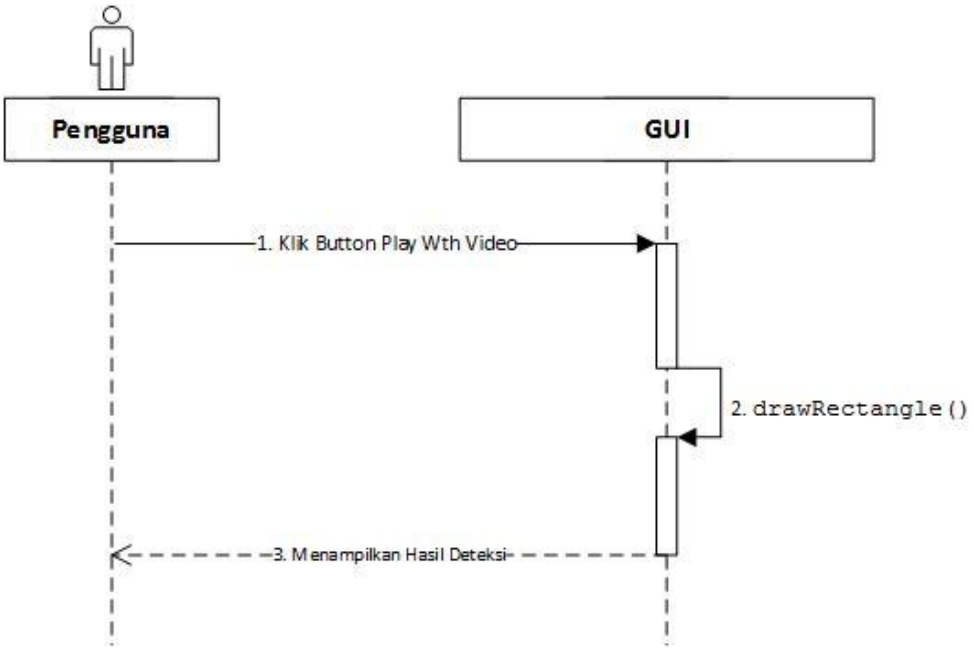
Pada Gambar 3.29 adalah diagram sequence mengekstraksi HOG & LBP.



Gambar 3.29 Diagram Sequence Melakukan Deteksi Manusia

3.3.3.5 Diagram Sequence Melihat Hasil Deteksi

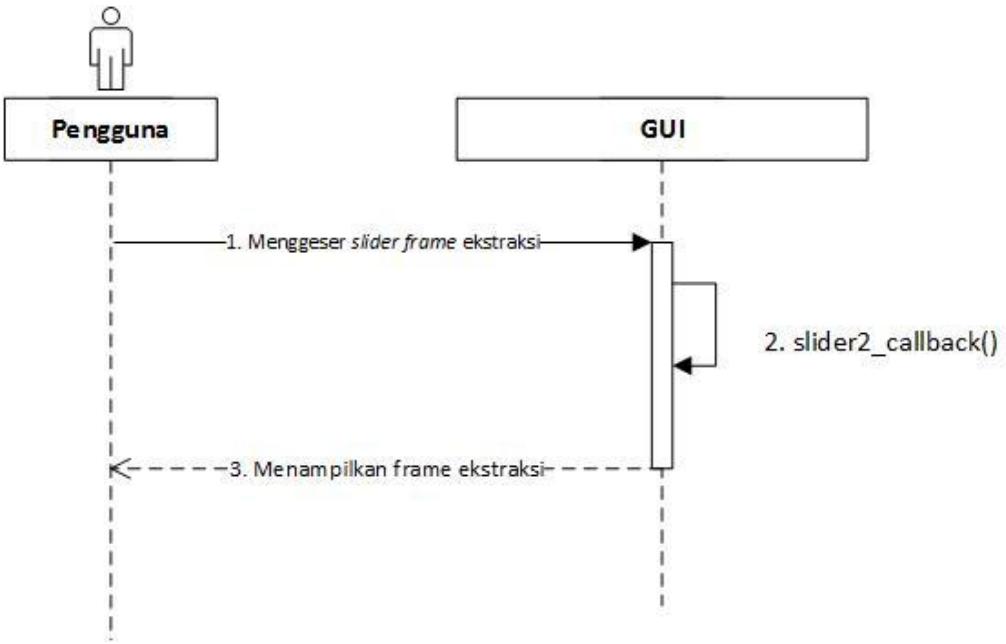
Pada Gambar 3.30 adalah diagram sequence melihat deteksi.



Gambar 3.30 Diagram Melihat Hasil Deteksi

3.3.3.6 Diagram Sequence Menggeser Frame Ekstraksi

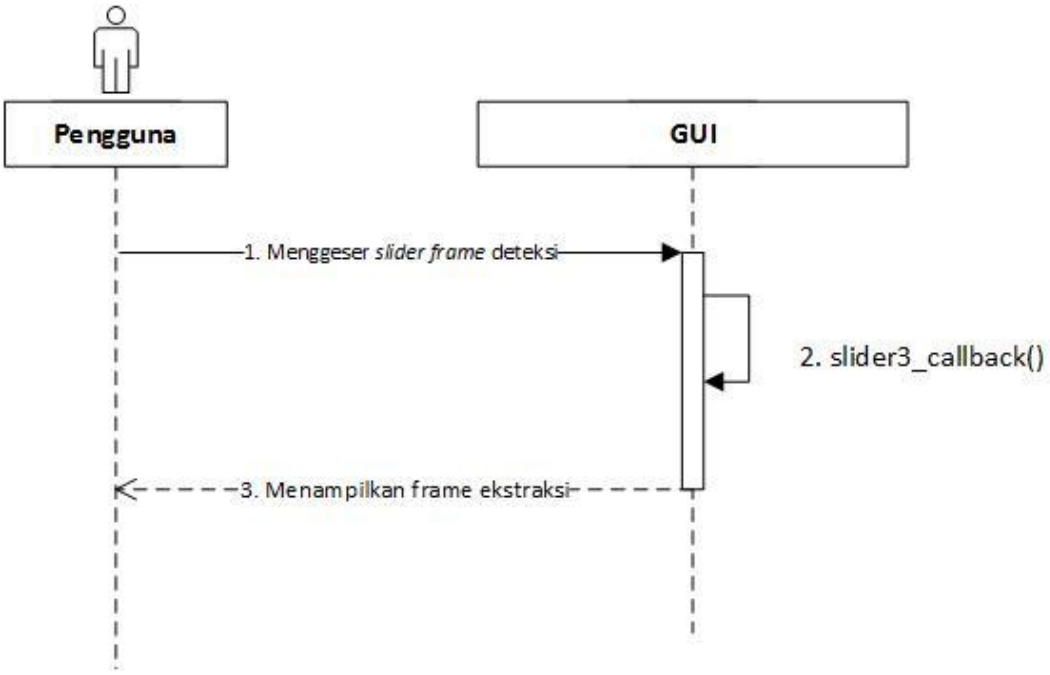
Pada Gambar 3.31 adalah diagram sequence menggeser *frame* ekstraksi.



Gambar 3.31 Diagram Sequence Menggeser Frame Ekstraksi

3.3.3.7 Diagram Sequence Menggeser Frame Deteksi

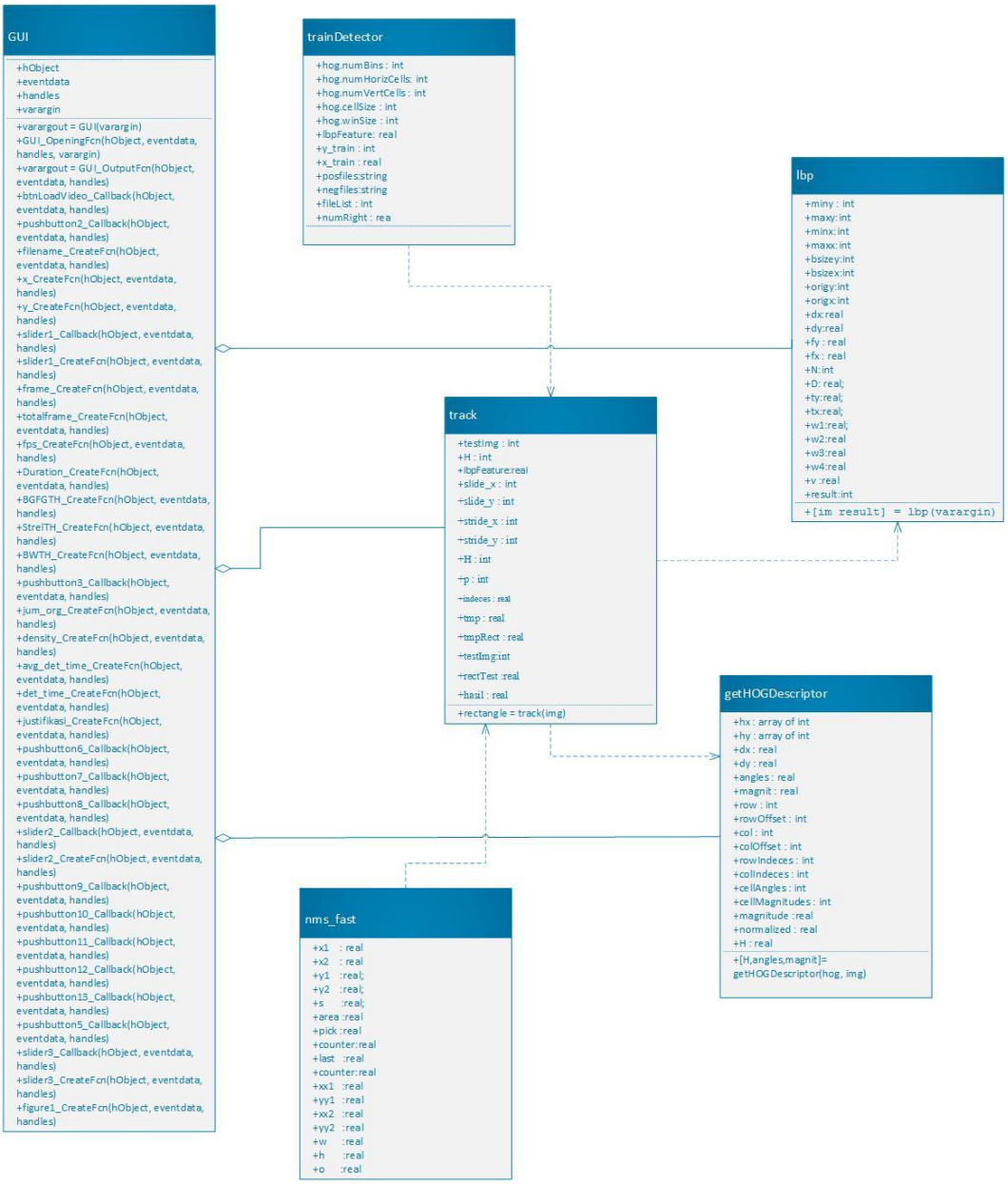
Pada Gambar 3.32 adalah diagram sequence menggeser deteksi.



Gambar 3.32 Diagram Sequence Menggeser Frame Deteksi

3.3.4 Diagram Class

Class diagram membantu menggambarkan struktur kelas-kelas dari suatu sistem dan merupakan tipe diagram yang paling sering ditemui dalam pemodelan sistem berbasis *object-oriented.* Adapun rincian dari class diagram sistem dapat dilihat pada Gambar 3.33.



Gambar 3.33 Diagram Class

3.4 Perancangan Simulasi

Perancangan sistem merupakan penggambaran dan perencanaan dari beberapa elemen sistem yang terpisah ke dalam suatu kesatuan model yang utuh. Tahapan ini meliputi perancangan antarmuka, perancangan pesan dan jaringan semantik.

3.4.1 Perancangan Antarmuka

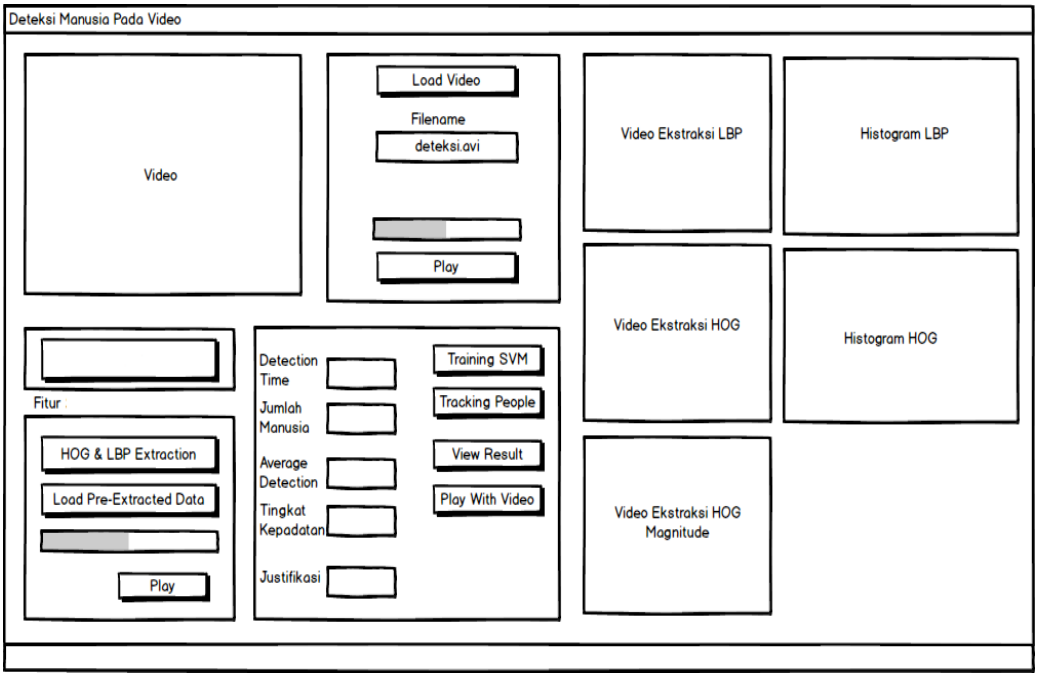
Perancangan Antarmuka mendeskripsikan rencana tampilan dari setiap tampilan yang akan digunakan. Perancangan terdiri dari perancangan antarmuka, dan pesan.

3.4.1.1 Perancangan Antarmuka Sistem

Perancangan antarmuka dibuat untuk memberikan gambaran rancangan perangkat lunak yang akan dibangun, sehingga akan mempermudah dalam pembangunan aplikasi. Gambar 3.34 merupakan rancangan antarmuka sistem. Dalam rancangan antarmuka yang dibangun terdapat fitur antara lain:

1. Menu Utama

F01



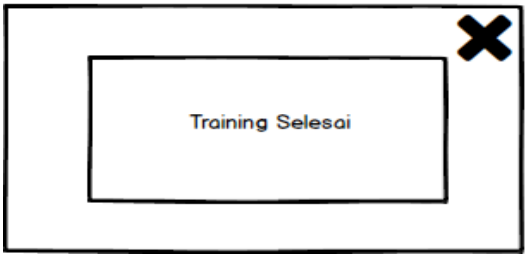
Keterangan :
Button Load Video menambahkan video.
Button Play untuk memutar video yang telah ditambahkan.
Button HOG & LBP Extraction untuk melakukan ekstraksi pada video.
Button Load Pre-Extracted Data untuk menampilkan hasil ekstraksi video ketika program telah dikeluarkan.
Button Play pada fitur 2 untuk memutar video hasil ekstraksi HOG-LBP.
Button Training SVM untuk melakukan pelatihan pada sistem dengan SVM.
Button Tracking people untuk melakukan pendeteksian dan menghitung jumlah manusia pada Video.
Button View Tracking Result untuk melihat hasil pendeteksian dan penghitungan jumlah manusia.
Button Play With Video untuk memutar video yang telah diuji.

Gambar 3.34 Perancangan Antarmuka Menu Utama

3.4.1.2 Perancangan Pesan

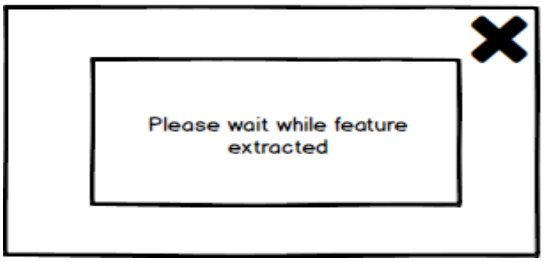
Perancangan Antarmuka mendeskripsikan rencana tampilan dari setiap tampilan yang akan digunakan. Perancangan terdiri dari perancangan form dan pesan.

1. Perancangan pesan tombol ketika melakukan training data latih (M01)



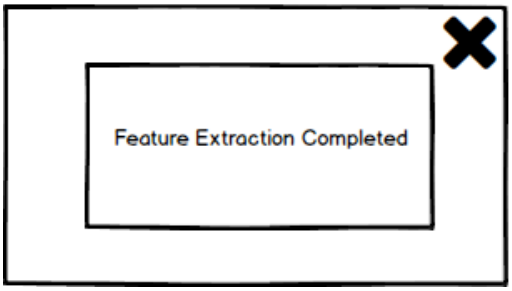
Gambar 3.35 Perancangan Pesan M01

2. Perancangan pesan tombol ketika frame pada video sedang diekstraksi (M02).



Gambar 3.36 Perancangan Pesan M02

3. Perancangan pesan tombol ketika frame pada video sudah diekstraksi (M03).



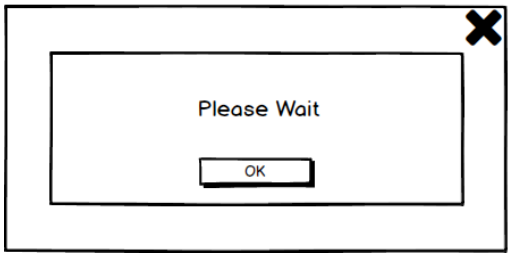
Gambar 3.37 Perancangan Pesan M03

4. Perancangan pesan tombol ketika video sudah dilakukan pendeteksian (M04)



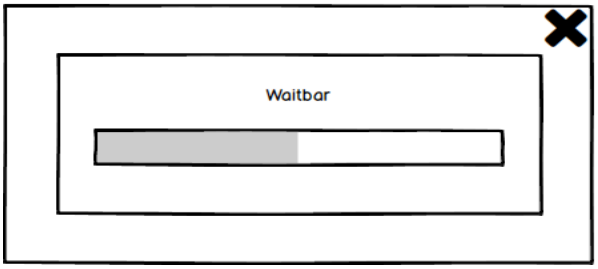
Gambar 3.38 Perancangan Pesan M04

5. Perancangan pesan tombol ketika menunggu proses. (M05).



Gambar 3.39 Perancangan Pesan M05

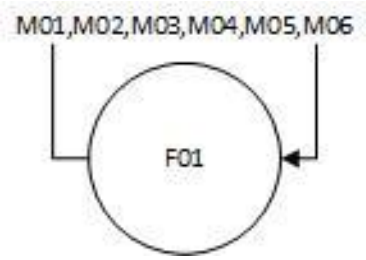
6. Perancangan pesan tombol ketika proses sedang berjalan (M06).



Gambar 3.40 Perancangan Pesan M06

3.4.1.3 Jaringan Semantik

Jaringan semantik merupakan gambaran pengetahuan grafis yang menunjukan hubungan antar berbagai objek. Jaringan semantik terdiri dari lingkaran-lingkaran yang menunjukan objek dan informasi tentang objek-objek tertentu, untuk lebih jelasnya dapat dilihat pada Gambar 3.41.



Gambar 3.41 Jaringan Semantic

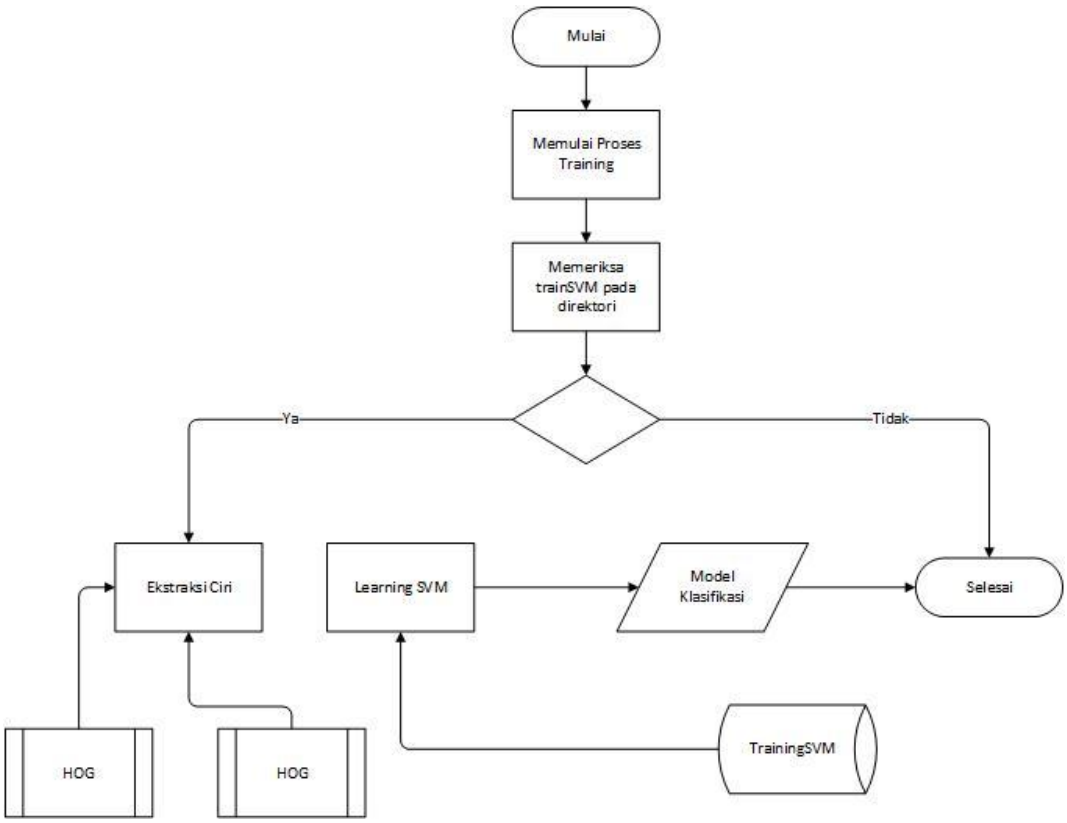
3.5 Perancangan Prosedural

Perancangan prosedural merupakan tahap terahir dari pada proses perancangan. Perancangan prosedural terdiri dari beberapa *flowchart* dari

subsistem aplikasi yang akan dibangun. Berikut adalah perancangan pesan dari subsistem aplikasi yang akan dibangun dapat dilihat pada Gambar 3.42 dan Gambar 3.43.

3.5.1 Perancangan Prosedural Training Dataset

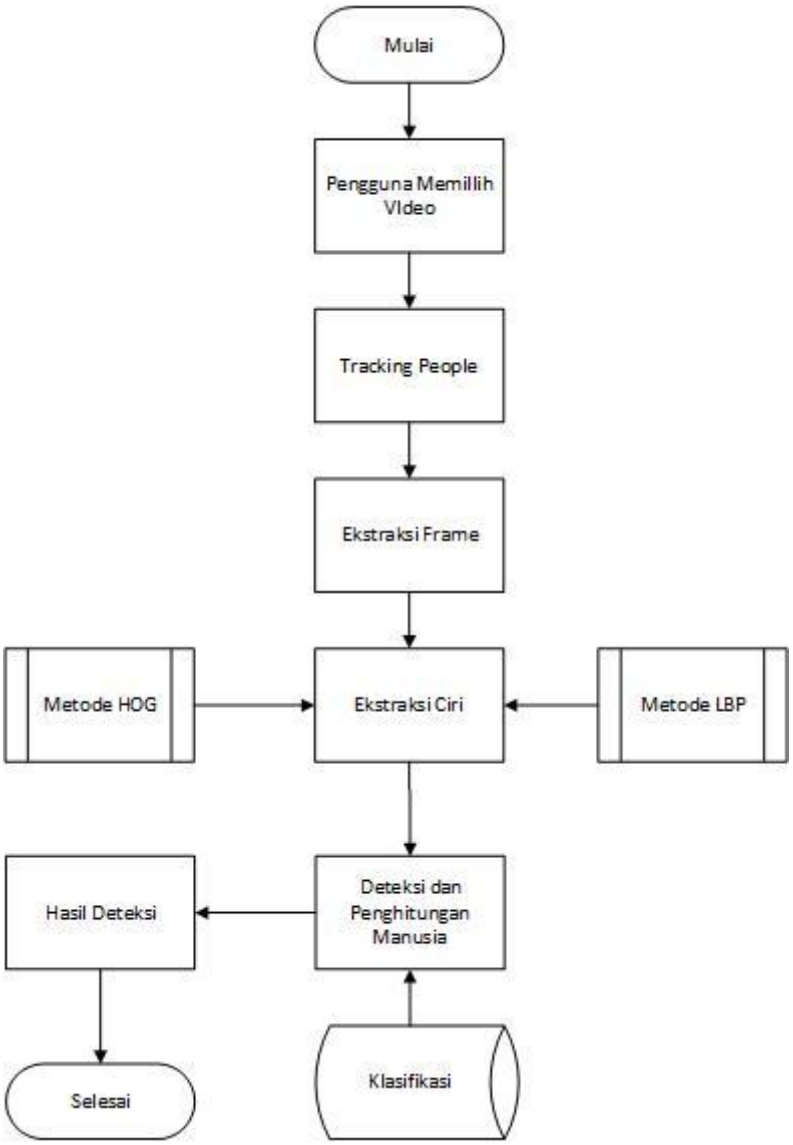
Gambar 3.42 adalah diagram alir rancangan prosedural untuk proses training dataset.



Gambar 3.42 Prosedural Learning

3.5.2 Perancangan Prosedural Pengujian

Gambar 3.43 adalah rancangan prosedural untuk proses pendeteksian dan penghitungan manusia dan proses menyimpan gambar hasil deteksi.



Gambar 3.43 Prosedural Pengujian