BAB 3 ANALISIS DAN PERANCANGAN

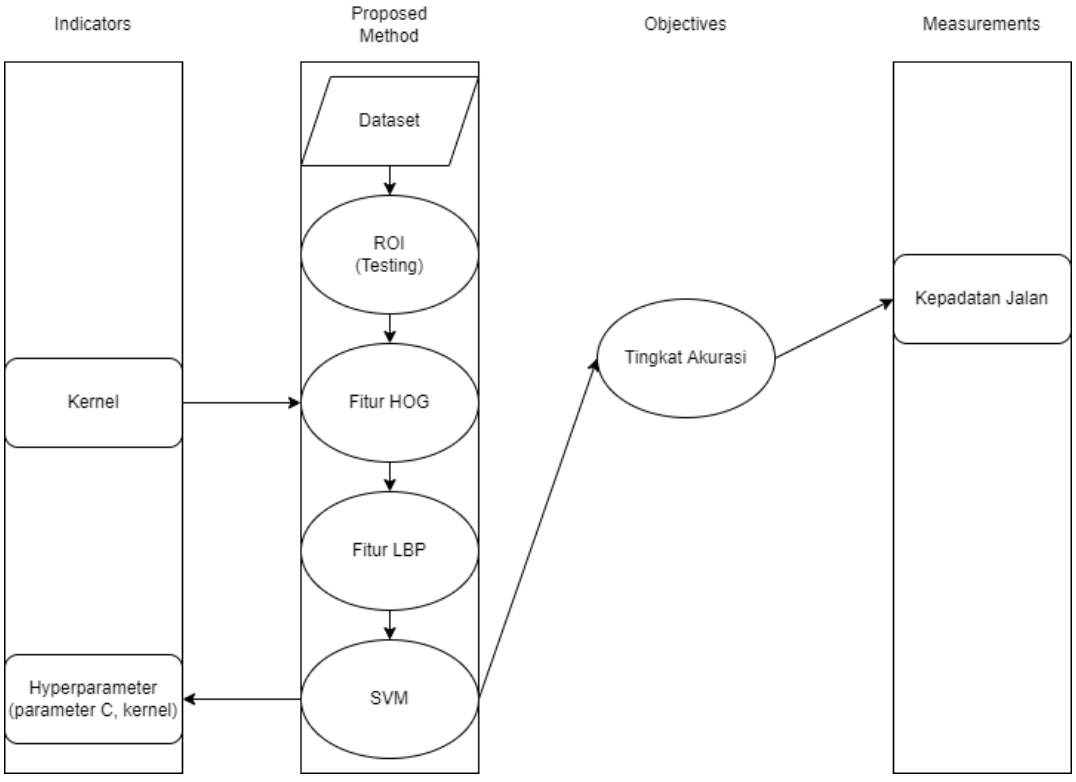
3.1 Analisis Masalah

Pada sub bab latar belakang, sudah dijelaskan pentingnya peranan lampu lalu lintas dalam kelancaran jalan. Pada penelitian ini, akan dibentuk sistem untuk mendeteksi kepadatan pada sebuah jalan tertentu. *Input* yang dipakai akan dibagi menjadi 2 jenis *cell*; *cell* yang berisi *traffic* dan *cell* yang tidak berisi *traffic*.

Input dalam sistem ini merupakan sebuah video jalan yang diambil oleh sebuah cctv dan memiliki dimensi 360x268 *pixel*. Sistem akan memiliki *output* gambar yang diambil dari video yang terbagi dalam beberapa *cell*. Sebuah gambar disebut padat jika gambar tersebut memiliki *cell Traffic* yang lebih banyak dibanding *cell Not-Traffic*.

3.2 Kerangka Pemikiran

Pada bab ini akan dijelaskan analisis masalah yang akan diatasi beserta metode dan cara kerja dari perangkat lunak yang akan dikembangkan. Seperti pada



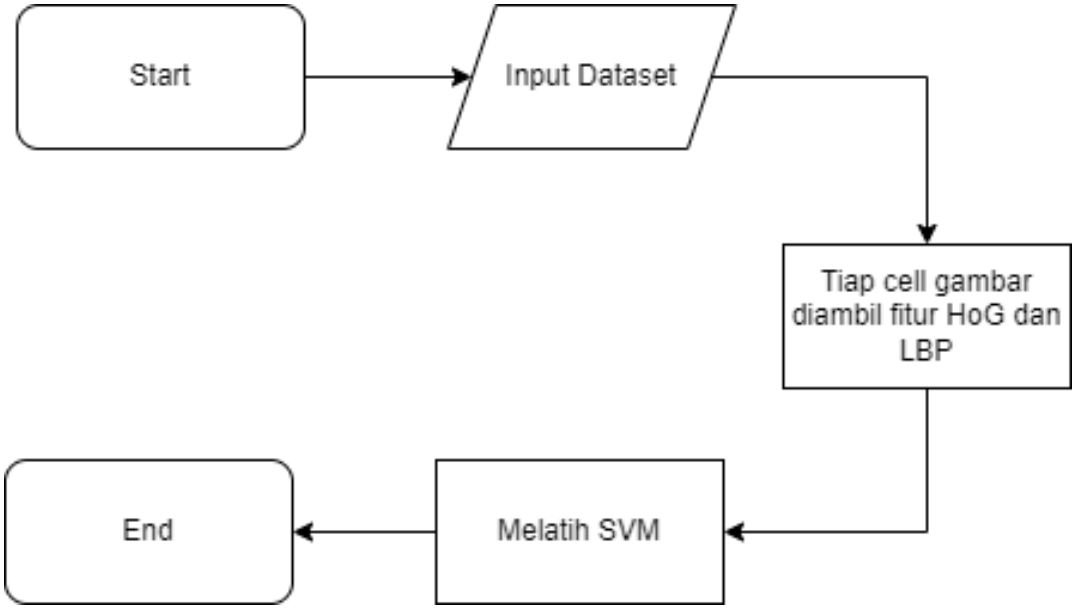
Gambar 3.1 Kerangka Pemikiran

gambar 3.1, langkah awal yang harus dilakukan yaitu pengumpulan *input* berupa video jalan yang diambil dari *CCTV* yang dipasang di dekat jalan. Setelah itu akan

diambil fiturnya menggunakan HOG dan LBP. Setelah fitur didapat, kepadatan jalan akan dihtung dengan SVM.

3.3 Urutan Proses Global

Urutan proses global akan dipisah menjadi 2, yaitu untuk *training* dan *testing*. Berikut urutan proses global untuk *training*.



Gambar 3.2 *Flowchart Global Training*

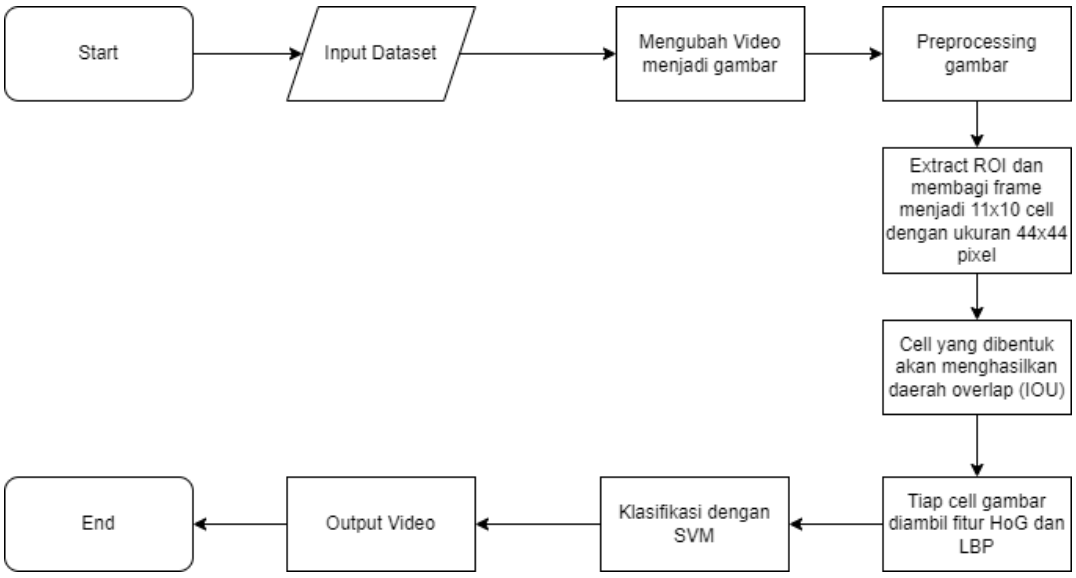
Berikut uraian dari *flowchart* pada gambar 3.2 :

1. *Input dataset*
   Untuk *training*, *dataset* yang digunakan merupakan gambar *cell* yang berukuran 44x44 *pixel*. *Dataset* berjumlah 20915 gambar dan dimasukkan ke dalam 2 kelas; *Traffic* dan *Not-Traffic*.

2. Mengekstrak fitur HoG dari tiap gambar
   Pada setiap gambar, akan diambil fitur HOG. Setiap gambar akan dibagi menjadi *sub cell* yang berukuran 3x3 dan dihitung dengan 8 orientasi. Fitur HOG yang akan didapat merupakan fitur vector sebanyak 5408.

3. Mengekstrak fitur LBP dari tiap gambar
   Fitur LBP diambil dari tiap gambar. LBP hanya bisa menerima gambar *grayscale* sebagai *input*. LBP memerlukan 2 *parameter* yang harus disediakan, *parameter* untuk *radius* dan *parameter* untuk jumlah titik di sekitar *radius* luar yang nilainya bergantung pada berbagai faktor seperti ukuran sel. Fitur LBP akhir yang didapat sebanyak 26.

4. Melatih SVM
   Fitur HOG dan LBP yang telah didapat akan digunakan untuk klasifikasi. Klasifikasi dilakukan menggunakan SVM dengan *kernel linear*.



Gambar 3.3 *Flowchart Global Testing*

Berikut uraian dari *flowchart* pada gambar 3.3 :

1. *Input dataset*
   Untuk *testing*, *dataset* yang digunakan merupakan 7 video yang masing memiliki 25 fps dan memiliki durasi 5 menit

2. Menentukan *Region of Interest* (RoI)
   Setiap *frame* ke-8 dari video, akan diambil *frame*nya. *Frame* tersebut akan dibagi menjadi *grid* dengan ukuran 10x11 *cell*, dan tiap *cell*nya berukuran 44x44 *pixel*.

3. *Cell Overlap*
   Cell yang dibentuk akan *overlap* dengan *cell* lainnya dan akan menghasilkan daerah *overlap*(IOU) yang berukuran 10x44 *pixel* untuk *cell* yang *overlap* ke samping dan 44x10 *pixel* untuk *cell* yang *overlap* ke bawah.

4. Mengekstrak fitur HoG dari tiap *cell*
   Pada setiap *cell*, akan diambil fitur HOG. Setiap *cell* akan dibagi menjadi *sub cell* yang berukuran 3x3 dan dihitung dengan 8 orientasi. Fitur HOG yang akan didapat merupakan fitur vector dengan 5408.

5. Mengekstrak fitur LBP dari tiap *cell*

Fitur LBP diambil dari tiap *cell*. LBP hanya bisa menerima gambar *grayscale* sebagai *input*. LBP memerlukan 2 *parameter* yang harus disediakan, *parameter* untuk *radius* dan *parameter* untuk jumlah titik di sekitar *radius* luar yang nilainya bergantung pada berbagai faktor seperti ukuran sel. Fitur LBP akhir yang didapat sebanyak 26.

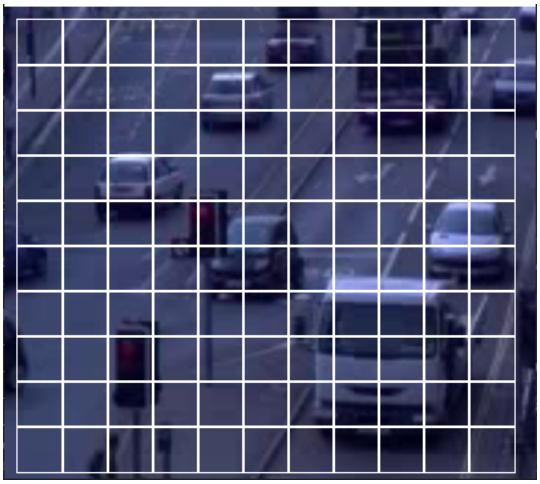
6. Klasifikasi dengan SVM
   Setiap *cell* yang sudah dicari nilai HOG dan LBPnya akan diklasifikasi dengan model yang didapat dari tahap *training*.

7. *Output* Video
   Output merupakan video *real-time* dan setiap *frame* ke-8 akan dibentuk *grid* dengan ukuran 10x11 *cell*. Setiap *cell* akan diberi border merah/hijau tergantung dari hasil klasifikasi.

3.4 Analisis Kasus

Dataset yang digunakan untuk *training* berupa gambar berukuran 44x44 *pixel* yang diambil dari video dan dibagi menjadi 2 kelas; *Traffic* dan *Not-Traffic*. *Dataset* untuk *testing* merupakan sebuah video jalan yang diambil dari CCTV. Video cctv akan dianalisis dan setiap *frame* akan dibagi menjadi 2 jenis; padat dan tidak padat. Sebuah *frame* dapat disebut padat jika *cell Traffic* lebih banyak dibanding *cell Not-Traffic*

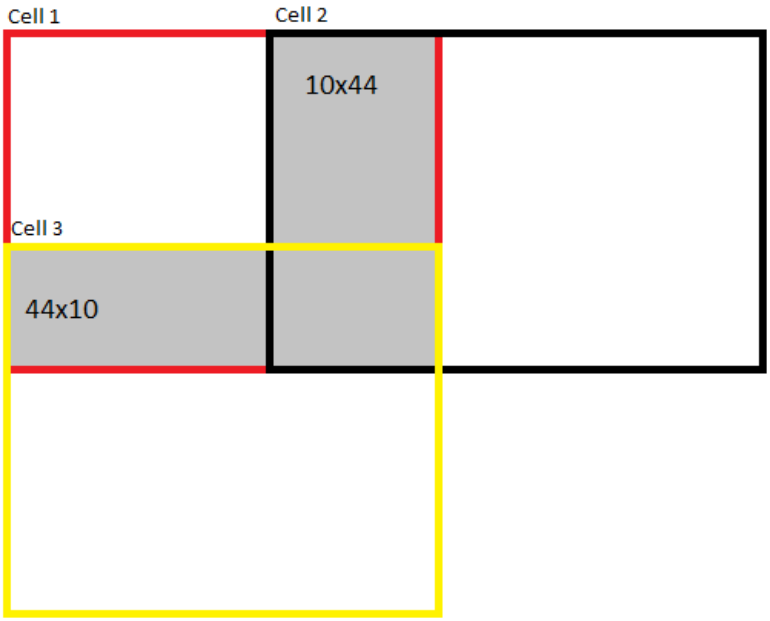
Pada tahap awal, setiap 8 *frame* dari video akan disimpan sebagai gambar dengan menggunakan *library* OpenCV. Setelah itu, setiap gambar akan dibagi menjadi *grid* yang berisi 10x11 *cell* dan tiap *cell* memiliki ukuran 44x44 *pixel*[1].



Gambar 3.4 *Grid pada ROI*

Cell pertama akan diletakkan pada posisi (0,0), untuk *cell* seterusnya akan

diletakkan setiap 34 *pixel* kesamping maupun kebawah dan akan menyebabkan *cell overlap* dengan *cell* lainnya dan akan menghasilkan daerah *overlap*(IOU). *Cell* yang *overlap* ke samping memiliki IOU berukuran 10x44 *pixel*, sedangkan untuk *cell* yang *overlap* ke bawah memiliki IOU berukuran 44x10 *pixel*.



Gambar 3.5 *Cell overlap.*

Gambar 3.5 menunjukkan contoh *cell overlap* dengan *cell* lainnya. Daerah yang berwarna abu merupakan daerah *overlap*(IOU).

Setiap *cell* yang didapat akan dibagi menjadi 2 kelas; *cell Traffic* dan *cell Not-Traffic*. Sebuah *cell* akan masuk kelas *Traffic* jika 30% dari isi *cell* tersebut merupakan bagian dari kendaraan.

Gambar 3.6 Kelas *Not-Traffic*(kiri) & Kelas *Traffic*(kanan). Tiap *pixel* memiliki ukuran 44x44 *pixel*

Dari 1 *frame* yang sudah didapat RoI, maka dapat disimpulkan, sebuah jalan

dapat disebut padat jika jumlah *cell traffic* lebih banyak daripada jumlah *cell not-traffic*. Dan, sebuah jalan disebut tidak padat jika jumlah *cell traffic* lebih sedikit daripada jumlah *cell not-traffic*.

Gambar 3.7 Contoh gambar padat dan tidak padat.

Dari gambar diatas, dapat dilihat bahwa gambar kiri disebut padat karena gambar tersebut memiliki 59 *cell* kendaraan dari 110 total *cell*. Dan gambar kanan disebut tidak padat karena mamiliki 39 *cell* kendaraan dari 110 total *cell*.

3.5 Algoritme *Histogram of Oriented Gradients*

Berikut langkah-langkah algoritme *Histogram of Oriented Gradients* yang diambil berdasarkan langkah-langkah pada bab 2.

Algorithm 1 *Histogram of Oriented Gradients*

1: *Input* merupakan gambar *cell* yang berukuran 44x44 *pixel*
2: Ubah gambar *input* menjadi *grayscale*
3: Hitung nilai gradien *pixel* gambar menggunakan Persamaan 2.2 dan 2.3 dengan *kernel* Sobel
4: Hitung nilai *magnitude* dan sudut(*angle*) tiap *pixel* dengan menggunakan Persamaan 2.5 dan 2.6
5: Setiap *pixel* dibagi menjadi 8x8 *cell* untuk membentuk sebuah *block*. untuk setiap *block*, hitung nilai 9-*point histogram* dan menghasilkan 9 *bins*
6: Untuk setiap *cell* di dalam *block*, hitung nilai *bin* ke-j dan j+1
7: Gabung 4 *block* dari 9-*point histogram matrix* untuk menghasilkan *block* baru
8: Normalisasi tiap *block* dengan *L2 Norm*
9: *Output* merupakan vektor fitur dari gambar *input*

3.6 Algoritme *Local Binary Pattern*

Berikut langkah-langkah algoritme *Local Binary Pattern* yang diambil berdasarkan langkah-langkah pada bab 2.

Algorithm 2 *Local Binary Pattern*

1: *Input* merupakan gambar *cell* yang berukuran 44x44 *pixel*
2: Ubah gambar menjadi *grayscale*
3: Untuk setiap *pixel*, pilih tetangga P
4: Ambil *pixel* tengah dan jadikan *threshold* untuk tetangga P
5: Setel ke 1 jika nilai *pixel* yang berdekatan lebih besar atau sama dengan nilai *pixel* tengah, setel ke 0 jika tidak
6: Hitung nilai LBP dengan menggunakan Persamaan 2.18
7: *Output* merupakan vektor fitur dari gambar *input*

3.7 Algoritme *Support Vector Machine*

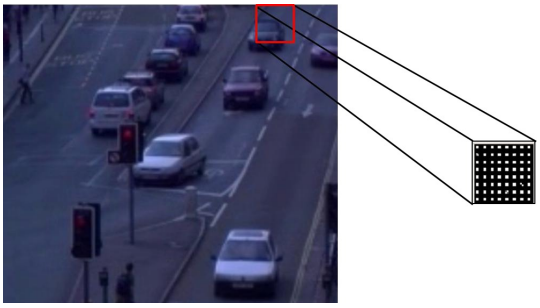
Berikut langkah-langkah algoritme *Support Vector Machine* yang diambil berdasarkan langkah-langkah pada bab 2.

Algorithm 3 *Support Vector Machine*

1: *Input* merupakan model yang berisi vector fitur HOG dan LBP
2: *Kernel* yang digunakan merupakan *kernel linear*
3: Latih setiap *cell* dengan menggunakan model untuk mengklasifikasikannya menjadi traffic atau not-traffic
4: Hitung kepadatan jalan dalam satu *frame* dengan menghitung banyak *cell Traffic*
5: Jika *cell Traffic* lebih banyak, maka jalan dalam *frame* tersebut merupakan padat
6: Jika *cell Not-Traffic* lebih banyak, maka jalan dalam *frame* tersebut merupakan tidak padat
7: *Output* merupakan klasifikasi *cell* (*Traffic* dan *Not-Traffic*)

3.8 *Histogram of Oriented Gradient*

Setiap *cell* yang sudah dibentuk, akan diambil fitur HoG dan LBP nya. Fitur HOG diekstrak dari setiap *cell* yang telah diubah menjadi *grayscale*. Untuk setiap *cell*, dibagi lagi menjadi beberapa *sub cell* untuk menghitung HOG-nya. Setiap *cell* dibagi menjadi *sub cell* yang berukuran 3x3 *pixel* dan dihitung dengan lebih dari 8 orientasi. Untuk mencari nilai fitur HOG-nya, akan digunakan *kernel* Sobel yang berukuran 5x5. Fitur HOG yang dihasilkan sebanyak 1568 dimensi.



Gambar 3.8 Fitur HOG yang diambil dari 1 *cell*[1]

Berikut adalah perhitungan fitur HOG pada posisi (0,0) :

17	17	17
17	16	16
18	17	17

Gambar 3.9 Nilai *pixel* ROI 3x3 pada posisi (0,0)

Untuk mendapat nilai HOG pada posisi (1,1), akan dicari nilai G_x dan G_y.

$$G_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} * \begin{bmatrix} 17 & 17 & 17 \\ 17 & 16 & 16 \\ 18 & 17 & 17 \end{bmatrix} = \begin{bmatrix} (17*-1) & (17*0) & (17*1) \\ (17*-2) & (16*0) & (16*2) \\ (18*-1) & (17*0) & (17*1) \end{bmatrix} = -3$$

$$G_x = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} * \begin{bmatrix} 17 & 17 & 17 \\ 17 & 16 & 16 \\ 18 & 17 & 17 \end{bmatrix} = \begin{bmatrix} (17*-1) & (17*-2) & (17*-1) \\ (17*0) & (16*0) & (16*0) \\ (18*1) & (17*2) & (17*1) \end{bmatrix} = 1$$

Setelah didapat nilai G_x dan G_y, maka harus juga dicari nilai *magnitude*(μ) dan arah(*angle*). Kedua nilai ini nanti akan dipetakan ke *Histogram of Gradients*.

$$\begin{aligned} Magnitude(\mu) &= |-3| + |1| \\ &= 3 + 1 \\ &= 4 \end{aligned}$$

$$\begin{aligned} Angle &= |\tan^{-1}(G_y/G_x)| \\ &= |\tan^{-1}|(1/-3)| \\ &= 0.333 \end{aligned}$$

Berikut adalah Histogram of Gradients yang telah diisi dengan nilai *magnitude* dan *angle* yang telah didapat. Baris kedua merupakan *bin* yang dipakai, dan baris pertama diisi dengan nilai *magnitude* yang nilai *bin*nya sama dengan nilai *angle*. Misal, nilai *angle* yang didapat = 0.333 dan nilai *magnitude* yang didapat = 4 maka nilai *magnitude* akan dimasukkan ke baris *value* yang berada di kolom yang sama dengan kolom *bin* 0 (dimasukkan ke *bin* 0 karena nilai *angle* dekat dengan 0).

Value	4								
Bin	0	20	40	60	80	100	120	140	160

Gambar 3.10 *Histogram of Gradients*

Kolom lain dalam Histogram of Gradients diisi jika ada nilai *magnitude* dan *angle* yang sesuai. Misalnya didapat nilai *angle* 80 dan nilai *magnitude* 2, maka pada kolom *bin* 80, value akan diisi dengan nilai 2.

Value	4				2				
Bin	0	20	40	60	80	100	120	140	160

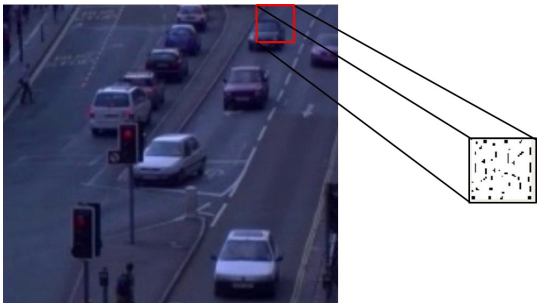
Gambar 3.11 *Histogram of Gradients*

Setelah dihitung histogramnya, maka setiap nilai dalam histogram harus di normalisasi dengan menggunakan persamaan pada bab 2, dalam contoh perhitungan ini nilai f_{bi} yang diambil adalah dari blok pertama sebesar 1. Nilai ε yang digunakan sebesar 1e-7 atau 10^{-7}

$$f_{bi} \leftarrow \frac{f_{bi}}{\sqrt{\|f_{bi}\|^2 + \varepsilon}}$$
$$f_{bi} \leftarrow \frac{5}{5^2 + 10^{-7}}$$
$$f_{bi} \leftarrow 0.2$$

3.9 *Local Binary Pattern*

Fitur LBP diekstrak dari *cell* yang sama yang dibuat pada ROI. LBP membutuhkan gambar *grayscale* sebagai *input*. Ada 2 *parameter* yang harus di definisikan, radius dan jumlah titik di sekitar radius luar yang nilainya bergantung pada berbagai faktor seperti ukuran *cell*. Pada tiap *cell*, akan dihasilkan histogram dan akan dinormalisasikan. Fitur LBP yang dihasilkan sebanyak 26 dimensi.



Gambar 3.12 Fitur LBP yang diambil dari 1 *cell*[1]

Setelah didapat, keduanya digabungkan dan menghasilkan 1594(1568+26) vektor fitur dimensi.

Pada *cell* yang sama dengan yang dipakai HOG, berikut perhitungannya : Untuk mencari nilai LBP pada posisi (1,1) dalam Gambar3.9, maka tiap *pixel* tetangganya harus dicari nilai $fLBP(x)$ sesuai dengan arah jam dan dimulai dari kiri atas. Berikut perhitungan pada tetangga 0 dan tetangga 1 :

$$\begin{aligned} fLBP(x) &= b(I_0 - I_c)2^j \\ &= b(17-16)2^0 \\ &= b(1) \end{aligned}$$

$$\begin{aligned} fLBP(x) &= b(I_1 - Ic)2^j \\ &= b(17-16)2^1 \\ &= b(2) \end{aligned}$$

1	2	4
128	LBP	0
128	32	16

Gambar 3.13 Nilai fLBP(x) yang sudah didapat

Sesuai dengan persamaan pada bab 2, maka tiap nilai yang sudah didapat akan di ubah menjadi berikut :

1	1	1
1	LBP	1
1	1	1

Gambar 3.14 Nilai fLBP(x) yang sudah diubah

Setelah itu, akan dihitung nilai LBP pada posisi(1,1).

1	1	1	1	1	1	1	1
2^0	2^1	2^2	2^3	2^4	2^5	2^6	2^7

$$\begin{aligned} LBP &= (2^0)*1+(2^1)*1+(2^2)*1+(2^3)*1+(2^4)*1+(2^5)*1+(2^6)*1+(2^7)*1 \\ &= 1+2+4+8+16+32+64+128 \\ &= 255 \end{aligned}$$

1	1	1
1	255	1
1	1	1

3.10 *Support Vector Machine*

Model SVM akan dibentuk dari vector fitur HOG dan LBP yang berjumlah 26 untuk fitur LBP dan 5408 untuk fitur HOG. Klasifikasi SVM dilakukan pada masing-masing *cell* dan kemudian hitung kepadatan lalu lintas dengan menghitung jumlah *cell* yang memiliki lalu lintas. Jika jumlah *cell* yang memiliki lalu lintas lebih banyak, maka kepadatan lalu lintas lebih banyak dan sebaliknya. Dengan melakukan ini, akan didapatkan estimasi akurat lalu lintas saat ini di jalur tertentu.

Pada tahap *training*, nilai SVM dihitung pada setiap *cell* dan membutuhkan *parameter* C, yang pada penelitian ini bernilai 5 dan sebuah *kernel*. *Kernel* yang digunakan dalam penelitian ini merupakan *kernel linear*. Setelah dihitung nilai SVMnya, setiap *cell* akan diklasifikasikan menjadi 2 kelas yaitu; *traffic* dan *not traffic*. Hasil *testing* akan dibandingkan dengan *dataset cell* yang sudah ada dan *model* SVM akan disimpan untuk digunakan pada tahap *testing*. Pada tahap *testing*, *model* SVM yang sudah didapat pada tahap *training* akan di *test* dengan video *dataset*. Tahap *training* dijalankan dengan menggunakan *library* sklearn, dan untuk menguji *model* SVM akan menggunakan *library* pickle.

Misal, nilai HOG dan LBP yang sudah didapat adalah sebagai berikut (0.577, 0.577) dan (0.066, 0.066). Pertama, harus dihitung panjang vektornya.

$$
\begin{aligned}
(HOG)||x|| &= \sqrt{x1^2 + x2^2} \\
&= \sqrt{0.577^2 + 0.577^2} \\
&= 0.817 \\
(LBP)||y|| &= \sqrt{y1^2 + y2 + 2} \\
&= \sqrt{0.0656^2 + 0.0656^2} \\
&= 0.078
\end{aligned}
$$

Setelah didapat panjang vektornya, akan dihitung arah vektornya

$$
\begin{aligned}
w(x1) &= \frac{x1}{||x||} \\
&= \frac{0.577}{0.816} \\
&= 0.707 \\
w(x2) &= \frac{x2}{||x||} \\
&= \frac{0.577}{0.816} \\
&= 0.707 \\
w(y1) &= \frac{y1}{||y||} \\
&= \frac{0.065}{0.078} \\
&= 0.837 \\
w(y2) &= \frac{y2}{||y||} \\
&= \frac{0.065}{0.816} \\
&= 0.547
\end{aligned}
$$

Setelah didapat panjang dan arah vektornya, kemudian dapat dicari nilai dot productnya dengan *kernel linear*.

$$
\begin{aligned}
K(x,y) &= (x_1 * y_1) + (x_2 * y_2) \\
&= (0.577 * 0.0656) + \\
&(0.577 * 0.0656) \\
&= 0.063
\end{aligned}
$$

Terakhir, dapat dihitung *hyperplane* dengan asumsi nilai *threshold*nya 0 untuk masing-masing fitur HOG dan LBP.

$$
\begin{aligned}
(x1) &= w * x1 \\
&= 0.707 * 0.577 \\
&= 0.408 \\
(x2) &= w * x2 \\
&= 0.707 * 0.577 \\
&= 0.408 \\
(y1) &= w * y1 \\
&= 0.707 * 0.0656 \\
&= 0.408 \\
(y2) &= w * y2 \\
&= 0.707 * 0.0656 \\
&= 0.408
\end{aligned}
$$