

**PENERAPAN METODE *HISTOGRAM OF ORIENTED  
GRADIENTS* UNTUK DETEKSI MOBIL**

**TUGAS AKHIR**

Diajukan sebagai syarat untuk menyelesaikan  
Program Studi Strata-1 Departemen Teknik Informatika

**Disusun Oleh:**  
**Satria Nugraha**  
**1115021**



**INSTITUT  
TEKNOLOGI  
HARAPAN  
BANGSA**

*Veritas vos liberabit*

**PROGRAM STUDI INFORMATIKA  
INSTITUT TEKNOLOGI HARAPAN BANGSA  
BANDUNG  
2019**

# BAB I

## PENDAHULUAN

### 1.1 Latar Belakang

Kemajuan teknologi di bidang transportasi membantu manusia dalam berpindah tempat antar lokasi dalam waktu singkat. Kemajuan teknologi ini dapat dilihat pada teknologi yang digunakan pada kendaraan bermotor seperti penggunaan sistem kemudi otomatis atau *self driving*. Pentingnya sistem kemudi otomatis adalah untuk membantu manusia meringankan pekerjaan dalam mengemudi dan menurunkan potensi kecelakaan akibat kelelahan dalam menyetir [1]. Sistem kemudi otomatis membutuhkan deteksi objek. Salah satu objek yang perlu di deteksi adalah mobil. Proses deteksi mobil diawali dengan pengambilan citra dari *dashboard* mobil, kemudian citra diproses untuk setiap *frame*. Dari setiap frame yang diperoleh kemudian diolah untuk mendeteksi posisi mobil, sehingga kecelakaan dapat dihindari.

Proses deteksi mobil meliputi tahap *preprocessing*, ekstraksi fitur, tahap klasifikasi. Tahap *preprocessing* berfungsi untuk mendeteksi *Region of Interest* (ROI), yaitu daerah dimana proses deteksi dilakukan dan juga untuk menentukan arah darimana citra diambil. Tahap ekstraksi fitur berfungsi untuk mengambil ciri dari citra untuk digunakan dalam membedakan objek. Tahap morfologi berfungsi untuk mengurangi jumlah objek dengan menghilangkan objek yang tidak sesuai kategori. Sedangkan tahap klasifikasi menentukan dan mengelompokkan objek yang serupa. Mobil yang telah dideteksi dihitung dan ditandai.

*Preprocessing* yang dilakukan yaitu *grayscale* dan menentukan ROI untuk memperkecil penggunaan memori [2]. ROI yang dimaksud berupa jalan dimana mobil akan melaju. ROI yang digunakan, dibuat secara manual berupa citra biner berdasarkan data latih yang digunakan.

Banyak penelitian telah meneliti metode yang akan digunakan untuk deteksi mobil, diantaranya adalah Haar-like, LBP, dan HOG. Metode ini merupakan *Feature Based Methods* yang direpresentasikan dengan baik untuk ekstraksi fitur. Berdasarkan penelitian yang dilakukan oleh [3], pendekatan berbasis fitur HOG mencapai hasil terbaik sehingga akan digunakan pada penelitian ini. Dengan fitur LBP, ditemukan banyak *false positives* yang terdeteksi. Fitur seperti Haar berkinerja terburuk, yang mungkin disebabkan oleh keragaman model sehingga mencegah kecocokan yang baik untuk area pola yang jernih dan berbayang [3].

SVM (*Support Vector Machine*), kNN (*k nearest neighborhood*), dan RF (*Random Forest*) merupakan metode klasifikasi yang populer. Ketiga metode ini menghasilkan hasil yang baik, tetapi performa kNN menurun ketika ruang fitur tidak stabil dan RF tidak bekerja

baik pada ruang vektor yang rumit. SVM terbukti dapat menangani masalah klasifikasi dengan efisien [4]. Dikarenakan metode SVM memiliki efisiensi yang cukup baik dan cocok dengan metode HOG dengan akurasi sebesar 83% [5], maka ditentukan *classifier* untuk penelitian ini menggunakan metode SVM.

Pada penelitian ini, *dataset* yang digunakan adalah citra mobil. Dataset yang digunakan terdiri dari 3 jenis data latih yang diambil dari sumber yang berbeda. Penelitian ini menggunakan *dataset UIUC Car Image Database*, *GTI Database*, dan *KITTI Database*. Arah pengambilan kamera untuk *dataset* dilakukan dari posisi *horizontal*. Untuk data latih dibagi menjadi 2 jenis yaitu mobil (kondisi utuh, tidak ada bagian mobil yang terpotong) dan bukan mobil (jalanan, motor, rumah, pejalan kaki, dan sebagainya). Kemudian untuk proses pengujian akan dilakukan analisis untuk kasus kondisi mobil dimana fitur yang terdapat pada citra uji tidak utuh. Pada dataset juga terdapat *template* untuk *Region of Interest* dimana berupa citra biner yang berfungsi untuk menandai lokasi dimana mobil akan melaju yang kemudian dimanfaatkan untuk daerah deteksi mobil.

### 1.2 Rumusan Masalah

Berdasarkan latar belakang yang telah diuraikan di atas, terdapat beberapa permasalahan yang dapat dirumuskan sebagai berikut:

1. Bagaimana menentukan *Region of Interest* yang tepat berdasarkan posisi kamera CCTV?
2. Berapa nilai akurasi pada hasil ekstraksi fitur dengan menggunakan metode *Histogram of Oriented Gradients* dan klasifikasi menggunakan metode *Support Vector Machine* untuk deteksi mobil?

### 1.3 Tujuan Penelitian

Berdasarkan rumusan masalah di atas, maka dapat dideskripsikan tujuan dari penelitian ini, yaitu:

1. Menentukan *Region of Interest* yang sesuai terhadap posisi kamera CCTV.
2. Melakukan penerapan metode *Histogram of Oriented Gradients* dan *Support Vector Machine* pada sistem deteksi mobil.
3. Menghitung nilai akurasi deteksi mobil dengan menggunakan metode *Confusion Matrix*.

### 1.4 Batasan Masalah

Agar penelitian ini lebih fokus dan terarah, maka penulis akan memberikan beberapa batasan masalah sebagai berikut:

1. *Region of Interest* (ROI) yang digunakan adalah jalan raya dimana mobil melaju berdasarkan posisi kamera CCTV di *dashboard* mobil.

2. Dataset yang digunakan adalah citra mobil, merupakan citra *grayscale* dengan ekstensi PGM berukuran 64 x 64 piksel dengan *depth bit* 8.
3. Posisi pengambilan citra hanya dilakukan dari belakang mobil saja, yang terdiri dari 4 bagian yaitu jauh, dekat, kiri, dan kanan.

### 1.5 Kontribusi Penelitian

Menerapkan metode *Histogram of Oriented Gradients* untuk melakukan ekstraksi fitur pada citra dan *Support Vector Machine* untuk melakukan klasifikasi objek.

### 1.6 Metodologi Penelitian

Berikut ini merupakan metodologi penelitian yang dilakukan dalam penelitian ini:

#### 1. Pengumpulan Data

Data latih yang digunakan berupa citra digital yang diambil dari <http://cogcomp.org/Data/Cars/>, [http://www.gti.ssr.upm.es/data/Vehicle\\_database.html](http://www.gti.ssr.upm.es/data/Vehicle_database.html), dan <http://www.cvlibs.net/datasets/kitti/> dan data uji berupa citra video dari <http://www.youtube.com>.

#### 2. Analisis dan Perancangan

Menganalisis dan merancang sistem yang akan dibuat berdasarkan *dataset* yang sudah diperoleh sebelumnya.

#### 3. Implementasi

Mengimplementasikan hasil perancangan yang telah dibuat sebelumnya dalam bentuk aplikasi untuk melakukan ekstraksi fitur.

#### 4. Pengujian

Melakukan pengujian terhadap aplikasi yang telah dibuat dan menghitung nilai akurasi dari hasil deteksi mobil. Selain itu akan dilakukan pengujian untuk kasis fitur tidak utuh dan analisis setiap *parameter* masukan pada setiap metode untuk mendapatkan nilai dari *parameter* yang sesuai untuk kasus *dataset* mobil.

### 1.7 Sistematika Penulisan

Laporan penelitian ini disusun berdasarkan sistematika penulisan berikut ini:

#### **BAB I Pendahuluan**

Bab ini menjelaskan tentang latar belakang, rumusan masalah, tujuan penelitian, batasan masalah, metodologi penelitian, dan sistematika penulisan.

#### **BAB II Landasan Teori**

Bab ini menjelaskan teori mengenai tinjauan pustaka, tinjauan studi, dan tinjauan

objek.

**BAB III      Analisis dan Perancangan**

Bab ini menjelaskan analisa terhadap masalah, solusi, dan perancangan sistem.

**BAB IV      Implementasi dan Pengujian**

Bab ini menjelaskan proses implementasi dan pengujian terhadap aplikasi yang dibuat.

**BAB V      Kesimpulan dan Saran**

Bab ini menjelaskan kesimpulan dari aplikasi yang telah dibuat dan saran untuk pengembangan aplikasi lain.

## BAB II

### LANDASAN TEORI

#### 2.1 Tinjauan Pustaka

Pada bagian ini akan membahas teori-teori terkait yang digunakan pada penelitian ini.

##### 2.1.1 Citra Digital

Citra digital merupakan representasi dari fungsi intensitas cahaya dalam bentuk diskrit pada bidang dua dimensi. Citra tersusun oleh sekumpulan piksel (*picture element*) yang memiliki koordinat  $(x,y)$  dan intensitas  $f(x,y)$ . Koordinat  $(x,y)$  menunjukkan letak/posisi piksel dalam suatu citra, sedangkan intensitas  $f(x,y)$  menunjukkan nilai intensitas warna citra [6]. Untuk membuat citra digital, diperlukan pengubahan data pengindraan kontinu menjadi bentuk digital. Tahap ini melibatkan dua proses, yaitu pengambilan sampel dan kuantisasi. Gambar 2.1 merupakan contoh dari citra digital.



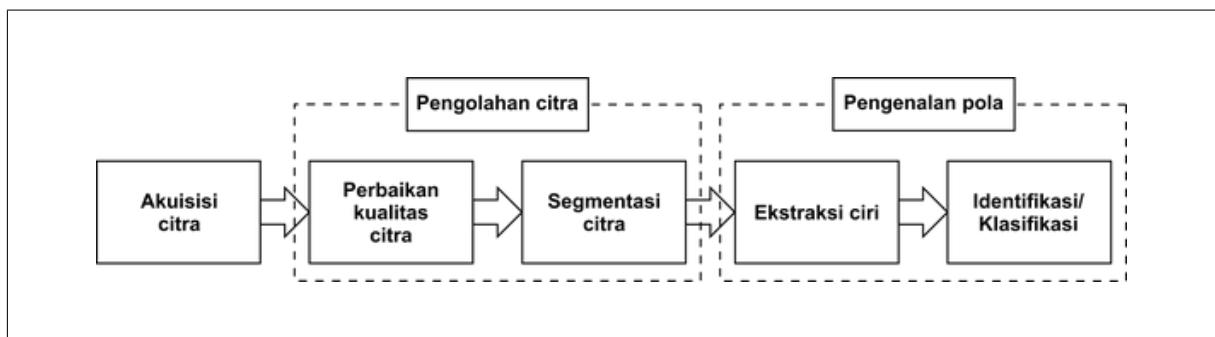
Gambar 2.1 Contoh Citra Digital [6]

Suatu citra dapat bersifat kontinu sehubungan dengan koordinat  $x$  dan  $y$ , dan juga dalam intensitas. Untuk mengubahnya menjadi bentuk digital, harus dicoba fungsi dalam koordinat dan intensitas. *Sampling* adalah proses untuk menentukan warna pada piksel tertentu pada citra dari sebuah gambar yang kontinu. Pada proses *sampling* biasanya dicari warna rata-rata dari gambar analog yang kemudian dibulatkan. Proses *sampling* sering juga disebut proses digitalisasi. *Sampling* merupakan bagian dari metodologi statistika. Adakalanya, dalam

proses *sampling*, warna rata-rata yang didapat di relasikan ke level warna tertentu. Contohnya apabila dalam citra hanya terdapat 16 tingkatan warna abu-abu, maka nilai rata-rata yang didapat dari proses sampling harus diasosiasikan ke 16 tingkatan tersebut. Proses mengasosiasikan warna rata-rata dengan tingkatan warna tertentu disebut dengan kuantisasi.

### 2.1.2 Pengolahan Citra

Pengolahan citra adalah salah satu cabang dari ilmu informatika (Komputer). Pengolahan citra berfokus pada usaha untuk melakukan transformasi suatu citra/gambar menjadi citra lain dengan menggunakan teknik tertentu. Berikut ini merupakan langkah-langkah yang umumnya dilakukan dalam merancang sebuah sistem pengolahan citra dan pengenalan pola:



Gambar 2.2 Proses Pengolahan Citra dan Pengenalan Pola

Operasi yang dilakukan untuk mengubah suatu citra menjadi citra lain dapat dikategorikan berdasarkan tujuan transformasi maupun cakupan operasi yang dilakukan terhadap citra. Berdasarkan tujuan transformasi operasi pengolahan citra dikategorikan menjadi peningkatan kualitas citra (*Image Enhancement*) dan pemulihan citra (*image restoration*). Operasi peningkatan kualitas citra bertujuan untuk meningkatkan fitur tertentu pada citra. Sedangkan operasi pemulihan citra bertujuan untuk mengembalikan kondisi citra pada kondisi yang diketahui sebelumnya akibat adanya *noise* yang menyebabkan penurunan kualitas citra. Berdasarkan cakupan operasi yang dilakukan terhadap citra, operasi pengolahan citra dikategorikan menjadi operasi titik, lokal, dan global.

Operasi titik merupakan operasi yang dilakukan terhadap setiap piksel pada citra yang keluarannya hanya ditentukan oleh nilai piksel itu sendiri. Operasi titik dapat dibagi menjadi tiga macam, yaitu berdasarkan intensitas, geometri, dan gabungan keduanya. Contoh operasi titik berdasar intensitas adalah operasi pengambangan (*thresholding*) yaitu membuat citra biner, operasi negatif (*negative image*) yaitu membuat citra negatif, pemotongan citra (*clipping*), dan pencerahan citra (*image brightening*) yaitu menambahkan atau mengurangi konstanta untuk memperbaiki kecerahan pada citra. Operasi lokal merupakan operasi yang dilakukan terhadap setiap piksel pada citra yang keluarannya dipengaruhi oleh piksel tersebut dan piksel lainnya dalam suatu daerah tertentu. Salah satu contoh dari operasi berbasis lokal

adalah operasi konvolusi untuk mendeteksi tepi (*edge detection*) dan pelembutan citra (*image smoothing*). Operasi global merupakan operasi yang dilakukan terhadap setiap piksel pada citra yang keluarannya ditentukan oleh keseluruhan piksel yang membentuk citra. Contoh operasi global adalah operasi penyetaraan histogram untuk meningkatkan kualitas citra.

### 2.1.3 *Preprocessing*

*Preprocessing* merupakan proses pengolahan data asli sebelum tahapan pengolahan data. Proses ini bertujuan agar data siap untuk diproses ke tahap ekstraksi fitur. *Preprocessing* memiliki tujuan untuk menghilangkan derau, memperjelas fitur, mengubah ukuran citra, dan konversi data. Contoh dari *preprocessing* yaitu pengabuan citra (*grayscale*), binerisasi citra, *cropping* citra, dan *resize* citra [6].

#### 2.1.3.1 Pengabuan Citra

Citra *grayscale* adalah suatu citra yang hanya memiliki warna berupa tingkat keabuan. Citra *grayscale* digunakan karena membutuhkan sedikit informasi yang diberikan pada tiap piksel dibandingkan dengan citra berwarna sehingga dalam *grayscale* image hanya membutuhkan nilai intensitas tunggal dibandingkan dengan citra berwarna membutuhkan tiga intensitas untuk tiap pikselnya. Intensitas dari citra *grayscale* disimpan dalam 8 bit integer yang memberikan 256 kemungkinan yang mana dimulai dari level 0 sampai dengan 255 (0 untuk hitam dan 255 untuk putih) [6]. Gambar 2.3 merupakan contoh dari citra *grayscale*.



Gambar 2.3 Contoh Citra *Grayscale* [6]

Persamaan pengabuan citra dapat dilihat pada persamaan 2 . 1 dengan  $R$  melambangkan intensitas warna merah,  $G$  untuk intensitas warna hijau, dan  $B$  untuk intensitas warna biru.

$$Grayvalue = 0.299R + 0.587G + 0.114B \quad (2 . 1)$$

Persamaan 2 . 1 menyimpulkan bahwa persentase warna hijau yang paling besar karena manusia cenderung lebih sensitif terhadap perubahan warna hijau yang memiliki panjang gelombang sekitar 500-570 nm, merah, lalu biru, dan merupakan rekomendasi dari *International Telecommunication Union Radiocommunication Sector*.

### 2.1.3.2 Citra Biner

Citra biner (*binary image*) adalah citra yang hanya mempunyai dua nilai derajat keabuan yaitu hitam dan putih. Citra biner bernilai 1 untuk objek dan 0 untuk latar belakang. Citra biner sering kali digunakan karena mempercepat waktu proses dan memperkecil penggunaan memori[7]. Meskipun komputer saat ini dapat memproses citra hitam-putih (*grayscale*) maupun citra berwarna, namun citra biner masih tetap dipertahankan keberadaannya.



Gambar 2.4 Contoh Citra Biner [6].

Sama seperti citra *grayscale*, citra biner juga merupakan citra yang hanya memiliki satu kanal warna. Citra biner memiliki kedalaman bit sebesar 1-bit. *Thresholding* adalah metode untuk mengubah citra *grayscale* menjadi citra biner sehingga objek dapat dipisahkan dari *background*. Untuk mendapat nilai *threshold* yang adaptif menggunakan metode Otsu. Nilai intensitas warna pada setiap piksel citra biner dibagi menjadi  $2^{\wedge}1 = 2$  warna yaitu warna hitam yang dinyatakan oleh nilai 0 dan warna putih yang dinyatakan oleh nilai 1. Persamaan

yang digunakan untuk mengkonversi nilai piksel citra *grayscale* menjadi biner pada metode *thresholding* adalah:

$$g(x,y) = \begin{cases} 1, & \text{jika } f(x,y) \geq T \\ 0, & \text{jika } f(x,y) < T \end{cases} \quad (2.2)$$

---

Keterangan:

$g(x,y)$  : citra biner

$f(x,y)$  : citra *grayscale*

$T$  : *threshold*

---

Citra biner digunakan saat proses penentuan ROI. ROI yang berupa citra biner digunakan untuk menandai lokasi dimana mobil akan melaju. ROI yang digunakan berupa template dari dataset yang diperoleh.

### 2.1.3.3 Otsu Tresholding

Tujuan dari metode Otsu adalah membagi histogram citra gray level kedalam dua daerah yang berbeda secara otomatis tanpa membutuhkan bantuan pengguna untuk memasukkan nilai ambang. Pendekatan yang dilakukan oleh metode otsu adalah dengan melakukan analisis diskriminan yaitu menentukan suatu variabel (nilai ambang atau *threshold*) yang dapat membedakan antara dua atau lebih kelompok yang muncul secara alami. Analisis Diskriminan akan memaksimumkan nilai ambang agar dapat membagi objek latar depan (*foreground*) dan latar belakang (*background*).

Langkah awal yang harus dilakukan adalah membuat histogram. Dari histogram dapat diketahui jumlah piksel untuk setiap tingkat keabuan. Tingkat keabuan citra dinyatakan dengan  $i$  sampai dengan  $L$ . Level ke  $i$  dimulai dari 1, yaitu piksel 0. Untuk  $L$ , maksimal level adalah 256 dengan piksel bernilai 255. Nilai ambang yang akan dicari dari suatu citra *grayscale* dinyatakan dengan  $k$ . Nilai  $k$  berkisar antara 0 sampai dengan  $L - 1$ , dengan nilai  $L = 256$  (simbol histogram adalah  $P_i$ ) [8]. Jadi probabilitas setip piksel pada level ke  $i$  dinyatakan dengan persamaan berikut.

$$P_i = \frac{n_i}{N} \quad (2.3)$$

Keterangan:

- $P_i$  : probabilitas piksel ke-i
  - $n_i$  : jumlah piksel dengan tingkat keabuan i
  - $N$  : total jumlah piksel pada citra
- 

Langkah selanjutnya mencari nilai jumlah kumulatif, rerata kumulatif dan intensitas global. mencari nilai tersebut dapat melihat persamaan (2 . 4), persamaan (2 . 5), dan persamaan (2 . 6).

$$\omega(k) = \sum_{i=0}^k p_i \quad (2 . 4)$$

$$\mu(k) = \sum_{i=0}^k i.p_i \quad (2 . 5)$$

$$\mu_T(k) = \sum_{i=0}^{L-1} i.p_i \quad (2 . 6)$$

Keterangan:

- $k$  : tingkat level keabuan dimana setiap rentang piksel akan dihitung
  - $\omega(k)$  : jumlah kumulatif
  - $\mu(k)$  : rerata Kumulatif
  - $\mu_T(k)$  : rerata Intensitas Global
- 

Langkah selanjutnya adalah menentukan varian antar kelas (*between class variance*).

$$\sigma_B^2(k) = \frac{[\mu_T \omega(k) - \mu(k)]^2}{\omega(k)[1 - \omega(k)]} \quad (2 . 7)$$

Keterangan:

- $\sigma_B^2$  : nilai ambang
  - $k$  : tingkat level keabuan dimana setiap rentang piksel akan dihitung
  - $\omega(k)$  : jumlah kumulatif
  - $\mu(k)$  : rerata Kumulatif
  - $\mu_T(k)$  : rerata Intensitas Global
-

Hasil dari perhitungan *between class variance* dicari nilai maksimal. Nilai yang paling besar digunakan sebagai *threshold* atau nilai ambang ( $k$ ), dengan persamaan (2 . 8).

$$\sigma_B^2(k*) = \max_{1 \leq x \leq L} \sigma_B^2 \quad (2 . 8)$$

---

Keterangan:

$\sigma_B^2$	: nilai ambang maksimal
$\omega(k)$	: jumlah kumulatif
$\mu(k)$	: rerata Kumulatif
$\mu_T(k)$	: rerata Intensitas Global
$\sigma_B^2$	: nilai ambang
$k$	: tingkat level keabuan dimana setiap rentang piksel akan dihitung

---

*Between class variance* bertujuan untuk mencari nilai ambang dari sebuah citra *grayscale*, nilai ambang atau *threshold* digunakan sebagai nilai acuan untuk mengubah citra *grayscale* ke citra biner. Setiap citra memiliki nilai ambang yang berbeda-beda.

#### 2.1.3.4 Derau

Derau (*Noise*) merupakan piksel yang mengganggu kualitas citra. Derau dapat disebabkan oleh gangguan fisis (optik) pada alat akuisisi maupun secara disengaja akibat proses pengolahan yang tidak sesuai [6]. Beberapa gangguan mungkin saja terjadi saat pengambilan citra, seperti kamera tidak fokus atau munculnya bintik-bintik yang bisa terjadi karena proses pengambilan gambar yang tidak sempurna. Setiap gangguan pada citra dinamakan derau, yang tidak hanya terjadi karena ketidak sempurnaan dalam proses pengambilan citra, tetapi dapat disebabkan juga oleh noda kotoran yang terjadi pada citra setelah pengambilan citra. Contohnya adalah bintik hitam atau putih yang muncul secara acak yang tidak diinginkan di dalam citra. Bintik acak ini disebut dengan derau *salt and pepper*. *Salt and Pepper* merupakan *noise* yang terkadang muncul pada citra. *Noise* ini dapat terjadi karena adanya gangguan pada citra, misalnya temperatur. *Salt and Pepper* pada citra berupa piksel hitam putih yang tersebar seperti pada gambar 2.5.



**Gambar 2.5** Contoh Derau *Salt and Pepper* [6].

#### 2.1.3.5 Segmentasi Citra

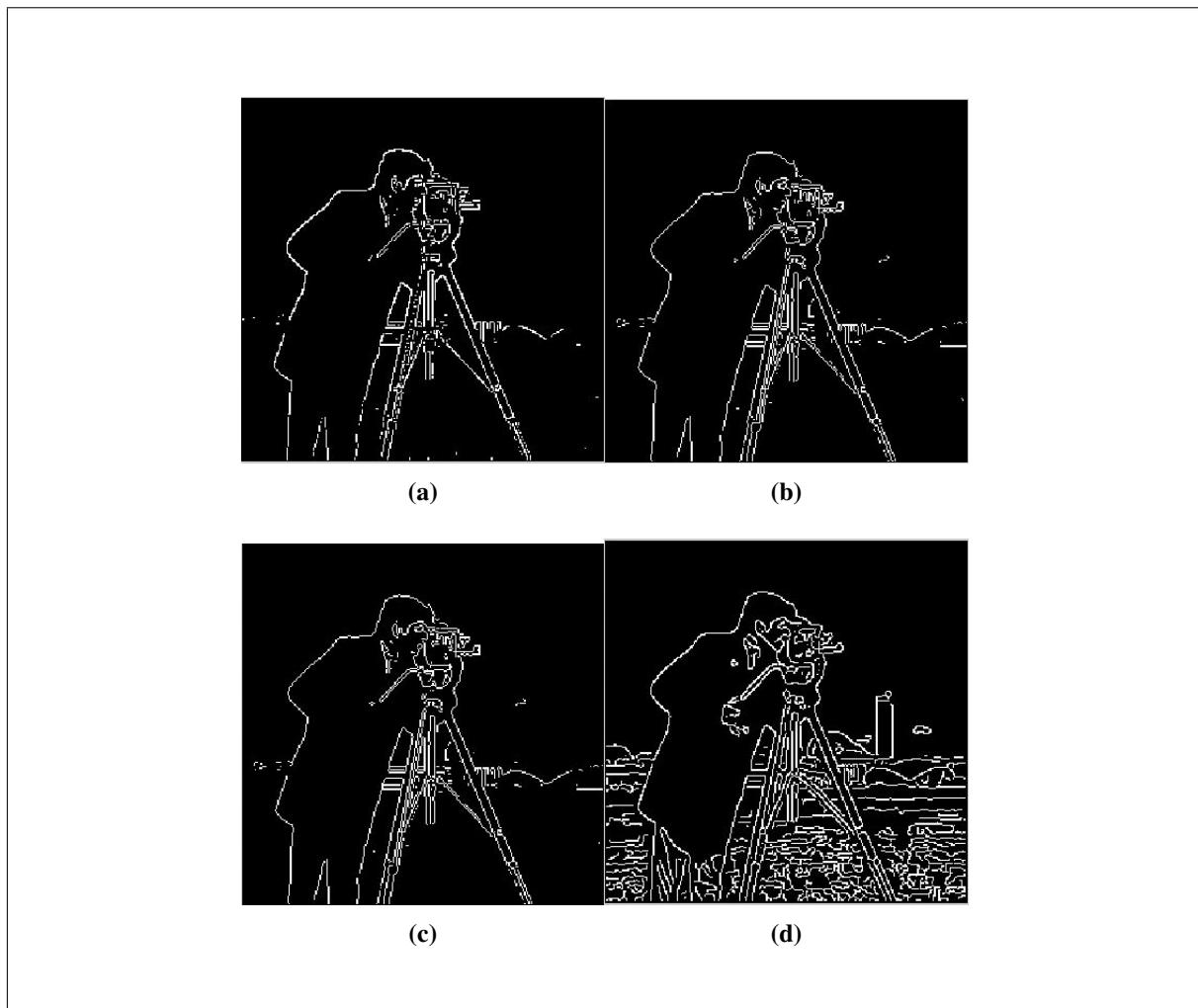
Segmentasi citra adalah membagi suatu citra menjadi wilayah - wilayah yang homogen [6]. Tahapan ini bertujuan untuk mempartisi citra menjadi bagian-bagian pokok yang mengandung informasi penting. Misalnya, memisahkan objek dan latar belakang. Segmentasi terdiri dari *downsampling*, penipisan dan deteksi tepian.

Tahap *downsampling* merupakan proses untuk menurunkan jumlah piksel dan menghilangkan sebagian informasi dari citra. Dengan resolusi citra yang tetap, *downsampling* menghasilkan ukuran citra yang lebih kecil. *Thinning* (penipisan) adalah proses mengurangi suatu obyek didalam citra digital menjadi ukuran yang minimum (objek (*region*) direduksi menjadi rangka (*skeleton*)). *Thinning* hanya digunakan pada citra biner dan menghasilkan citra biner lain sebagai outputnya . *Thinning* bertujuan untuk mengurangi bagian yang tidak perlu (*redundant*) sehingga hanya dihasilkan informasi yang penting saja. Pola hasil penipisan harus tetap menyerupai bentuk pola asal. Penentuan tepian suatu objek dalam citra merupakan salah satu wilayah pengolahan citra digital yang paling awal dan paling banyak diteliti. Proses ini seringkali ditempatkan sebagai langkah pertama dalam aplikasi segmentasi citra, yang bertujuan untuk mengenali objek-objek yang terdapat dalam citra ataupun konteks citra secara keseluruhan. Deteksi tepi berfungsi untuk mengidentifikasi garis batas (*boundary*) dari suatu objek yang terdapat pada citra.

Algoritme segmentasi didasarkan pada 2 buah karakteristik nilai derajat kecerahan citra, yaitu: *discontinuity* dan *similarity*. Pada *discontinuity*, citra dipisahkan/dibagi atas dasar perubahan yang mencolok dari derajat kecerahannya. Aplikasi yang umum adalah untuk deteksi titik, garis, area, dan sisi citra. Pada *similarity* didasarkan atas *thresholding*, *region growing*, dan *region splitting and merging*.

### 2.1.3.6 Deteksi Tepi

Deteksi tepi (*Edge detection*) adalah operasi yang dijalankan untuk mendeteksi garis tepi (*edges*) yang membatasi dua wilayah citra homogen yang memiliki tingkat kecerahan yang berbeda [6]. Tujuannya adalah untuk mengubah citra 2D menjadi bentuk kurva. Metode yang banyak digunakan untuk proses deteksi tepi adalah metode Robert, Prewitt, Sobel, dan Canny. Hasil dari metode - metode deteksi tepi dapat dilihat pada gambar 2.6.



**Gambar 2.6** Contoh Deteksi Tepi [6]. (a) Robert (b) Prewitt (c) Sobel (d) Canny

Metode pada deteksi tepi terletak pada penggunaan kernel yang berbeda. Metode Sobel merupakan pengembangan metode robert dengan menggunakan filter HPF yang diberi satu angka nol penyanga. Metode ini mengambil prinsip dari fungsi laplacian dan gaussian yang dikenal sebagai fungsi untuk membangkitkan HPF. Kelebihan dari metode sobel ini adalah kemampuan untuk mengurangi noise sebelum melakukan perhitungan deteksi tepi. Kernel filter yang digunakan dalam metode Sobel ini adalah:

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

Koordinat x didefinisikan di sini sebagai peningkatan pada arah horizontal, dan koordinat y didefinisikan sebagai peningkatan pada arah vertikal. Pada setiap titik dalam gambar, perkiraan gradien yang dihasilkan dapat dikombinasikan untuk memberikan besarnya gradien menggunakan rumus:

$$|G| = \sqrt{G_x^2 + G_y^2} \quad (2 . 9)$$

---

Keterangan:

- |       |                |
|-------|----------------|
| $G$   | : gradien      |
| $G_x$ | : arah sumbu X |
| $G_y$ | : arah sumbu Y |
- 

#### 2.1.3.7 *Region of Interest (ROI)*

*Region of Interest* adalah suatu bagian dari citra yang dipilih untuk kemudian diproses. Daerah tersebut dibedakan dengan menggunakan klasifikasi dan *masking*. Jika piksel pada *mask* tidak nol, maka pemrosesan citra dilakukan. Sebaliknya jika piksel pada *mask* sama dengan nol, proses tidak dijalankan. Setelah daerah yang diinginkan ditemukan, daerah tersebut ditandai dengan kotak untuk membatasi daerah yang akan dikenali. Dalam *Region of Interest*, citra dapat didefinisikan lebih dari satu region (bagian). *Region of Interest* sangat membantu untuk segmentasi dalam pemrosesan citra karena dengan menggunakan teknik ini citra atau obyek dapat lebih mudah dikenali. Karena obyek sudah akan dibagi dalam *region - region* tertentu sesuai dengan citra obyeknya. *Region of Interest* membantu dalam mengurangi penggunaan memori.

#### 2.1.3.8 *Sliding Window*

Pada bidang *computer vision*, *sliding window* merupakan sebuah daerah persegi atau persegi panjang yang memiliki ukuran tertentu dan akan bergecer dengan jarak tertentu secara berurutan ke seluruh daerah citra. Tahap ini dilakukan untuk memproses citra lokal secara bergantian dan pada umumnya digunakan untuk prooses pencarian dari suatu citra. Ukuran dari *window* dan jarak perpindahan antar *window* yang digunakan tergantung dari masalah yang akan diselesaikan maka biasanya ukuran tersebut disesuaikan dengan ukuran objek pada citra. Ukuran *window* yang optimal adalah *window* yang dapat mencakup keseluruhan objek, tidak terlalu besar atau kecil, oleh karena itu, perlu dilakukan analisis objek citra agar mendapatkan

keseluruhan objek. Sama halnya dengan jarak perpindahan antar *window*, bila terlalu besar akan terdapat bagian citra yang terlewat, bila terlalu kecil akan menyebabkan waktu komputasi yang lebih lama.

### 2.1.4 Ekstraksi Fitur

Ekstraksi fitur merupakan suatu pengambilan ciri (fitur) dari suatu bentuk yang nantinya nilai yang didapatkan akan dianalisis untuk proses selanjutnya [6]. Ekstraksi fitur bertujuan untuk mencari daerah fitur yang signifikan pada gambar. Ekstraksi fitur dilakukan dengan cara menghitung jumlah titik atau piksel yang ditemui dalam setiap pengecekan, dimana pengecekan dilakukan dalam berbagai arah pengecekan pada koordinat kartesian dari citra digital yang dianalisis, yaitu vertikal, horizontal. Ciri yang telah diekstrak kemudian digunakan sebagai parameter / nilai masukan untuk membedakan antara objek satu dengan lainnya pada tahapan identifikasi/ klasifikasi. Ekstraksi fitur terbagi menjadi tiga macam yaitu ekstraksi fitur bentuk, ekstraksi fitur tekstur, ekstraksi fitur warna.

Bentuk dari suatu objek adalah karakter permukaan yang diwakili oleh garis dan kontur. Fitur bentuk dikategorikan bergantung pada teknik yang digunakan berdasarkan daerah (*region-based*) dan berdasarkan batas (*boundary-based*). Teknik berdasarkan daerah (*region-based*) menggambarkan bentuk wilayah dengan menggunakan karakteristik internal, contohnya adalah piksel yang berada dalam suatu wilayah. Sedangkan teknik berdasarkan batas (*boundary-based*) menggambarkan bentuk daerah dengan menggunakan karakteristik eksternal, contohnya adalah piksel sepanjang batas objek.

Pada ekstraksi fitur tekstur, fitur pembeda adalah tekstur yang merupakan karakteristik penentu pada citra. Teknik statistik yang terkenal untuk ekstraksi fitur adalah matriks gray level co-occurrence. Teknik tersebut dilakukan dengan melakukan pemindaian untuk mencari jejak derajat keabuan setiap dua buah piksel yang dipisahkan dengan jarak  $d$  dan sudut  $\theta$  yang tetap. Biasanya sudut yang digunakan adalah 0, 45, 90, dan 135. Sedangkan pada ekstraksi fitur warna, ciri pembeda adalah warna. Biasanya ekstraksi fitur ini digunakan pada citra berwarna yang memiliki komposisi warna RGB (red, green, blue).

#### 2.1.4.1 *Histogram of Oriented Gradients*

*Histogram of Oriented Gradients* (HOG) merupakan metode ekstraksi fitur bentuk berupa garis dengan memperhatikan distribusi gradien intensitas lokal atau arah tepi. Dalam praktiknya ini dilakukan dengan membagi jendela gambar menjadi wilayah spasial kecil ("sel"). Untuk setiap sel mengumpulkan histogram 1-D lokal dari arah gradien atau orientasi tepi atas piksel sel. Sel dikumpulkan menjadi blok untuk dinormalisasi. Blok deskriptor hasil normalisasi disebut deskriptor HOG [9]. Pertama akan dihitung gradien untuk setiap piksel pada citra dari sumbu x dan y dengan menggunakan :

$$G_x(x,y) = I(x+1,y) - I(x-1,y) \quad (2 . 10)$$

$$G_y(x,y) = I(x,y+1) - I(x,y-1) \quad (2 . 11)$$

---

Keterangan:

$G_x(x,y)$  : gradien sumbu x

$G_y(x,y)$  : gradien sumbu y

$I(x,y)$  : nilai piksel citra dari baris x dan kolom y

---

Setelah mendapat gradien dari sumbu x dan y dari setiap piksel, besar nilai dan arah gradien dihitung menggunakan:

$$G(x,y) = \sqrt{G_x(x,y)^2 + G_y(x,y)^2} \quad (2 . 12)$$

$$\theta(x,y) = \arctan \frac{G_y(x,y)}{G_x(x,y)} \quad (2 . 13)$$

---

Keterangan:

$G(x,y)$  : besar nilai gradien sumbu x dan y

$\theta(x,y)$  : arah nilai gradien sumbu x dan y

---

Setiap piksel pada citra kemudian dibagi ke dalam sel, yang kemudian dihitung persebaran HOGnya menggunakan vote. Pertama, proses vote pada HOG dilakukan dengan membagi jumlah sudut gradien ke dalam jumlah *orientation bin* untuk menentukan nilai-nilai dari *bin*. Untuk setiap arah sudut gradien dari setiap piksel dalam sel dimasukkan ke dalam rentang *orientation bin* yang sudah ditentukan. Besar nilai gradien kemudian dibagi dengan *orientation bin* yang berhubungan. HOG dibuat untuk setiap sel.

Selanjutnya dilakukan normalisasi terhadap vote pada setiap *bin* dalam sel. Normalisasi dilakukan dalam 1 blok dengan ukuran m x n sel. Metode untuk normalisasi terdapat sebanyak 4 buah meliputi: *L2-Norm*, *L2-Hys*, *L1-sqrt*, dan *L1-norm*. Pada penelitian ini, akan digunakan normalisasi dengan metode *L2-Norm* dengan persamaan:

$$f = \sqrt{\frac{v}{\sum_{n=1}^N v}} \quad (2 . 14)$$

Keterangan:

- $v$  : bobot vektor hasil  $L1-Sqrt$  yang mewakili setiap *bin*
- $i$  : nilai counter i sampai dengan N
- $N$  : total nilai *bin* untuk normalisasi

Untuk algoritme rumus normalisasi L2-Hys merupakan algoritma mengikuti dari L2-Norm, namun dengan membatasi nilai maksimal hasil normalisasi sebesar 0,2.

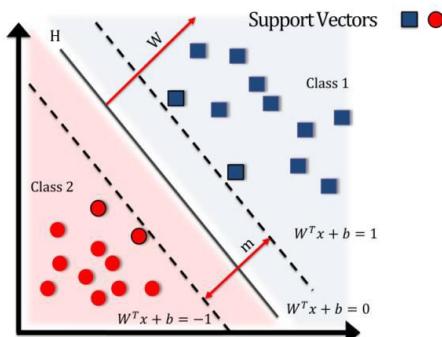
Proses normalisasi blok dilakukan dengan sliding window yang melakukan proses dengan pergeseran sebesar 1x ukuran sel secara vertikal dan horizontal. Proses ini akan bersifat overlapping untuk beberapa sel yang dinormalisasi sehingga menimbulkan informasi yang redundan, namun akurasi yang dihasilkan justru semakin meningkat [10].

### 2.1.5 Klasifikasi

Klasifikasi adalah sebuah proses untuk menemukan sebuah model yang menjelaskan dan membedakan konsep atau kelas data dengan tujuan memperkirakan kelas dari suatu objek yang kelasnya tidak diketahui [11].

#### 2.1.5.1 Support Vector Machine

*Support Vector Machines* (SVM) merupakan metode klasifikasi. SVM mengelompokkan fitur menjadi beberapa kelas menggunakan *hyperplane* pada suatu ruang yang disebut *feature space*.



Gambar 2.7 Contoh *Hyperplane* pada SVM [1].

Berdasarkan contoh pada gambar 2.7, *hyperplane* merupakan sebuah indikator pemisah dimana ditentukan berdasarkan *training* dengan margin tertinggi [11]. *Hyperplane* yang optimal harus memenuhi persamaan berikut:

$$w^T \cdot x + b = 0 \quad (2 . 15)$$

---

Keterangan:

- $w$  : vektor berat
  - $\cdot$  : perkalian vektor
  - $x$  : vektor input
  - $b$  : nilai bias
- 

Proses pemetaan dalam SVM menggunakan kernel, dan kernel yang biasanya digunakan adalah RBF (*Radian Basis Function*). Berikut rumus kernel RBF:

$$K(x, z) = e^{-((x-z)^2/(2\sigma^2)} \quad (2 . 16)$$

---

Keterangan:

- $xdanz$  : pasangan dua data training
  - $\sigma$  : konstanta
- 

Klasifikasi non-linier dilakukan menggunakan persamaan:

$$f(x) = sign(\sum_{i=1}^m \alpha_i y_i K(x, x_i) + b) \quad (2 . 17)$$

---

Keterangan:

- $\alpha_i$  : alpha
  - $y_i$  : kelas
  - $K(x, x_i)$  : kernel matriks
  - $b$  : bias
-

Jika nilai  $f(x)$  adalah 1, maka data tersebut akan masuk ke kelas positif. Sedangkan jika  $f(x)$  menunjukkan -1, maka data tersebut menunjukkan ke nilai kelas yang negatif.

### 2.1.6 Confusion Matrix

*Confusion Matrix* merupakan sebuah tabel yang digunakan untuk mengukur performa dari suatu *classifier* [12]. Berikut ini merupakan gambar untuk menjelaskan *confusion matrix*.

		Classifier Prediction	
		Positive	Negative
Actual Value	Positive	True Positive	False Negative
	Negative	False Positive	True Negative

Gambar 2.8 *Confusion Matrix*

Contoh di atas merupakan confusion matrix untuk klasifikasi 2 kelas. Tabel memiliki 4 istilah yang akan dijelaskan sebagai berikut:

1. True Positive (TP) : kondisi dimana data aktual bernilai positif dan prediksi dari klasifier bernilai positif.
2. True Negative (TN): kondisi dimana data aktual bernilai negatif dan prediksi dari klasifier bernilai negatif.
3. False Positive (FP): kondisi dimana data aktual bernilai negatif dan prediksi dari klasifier bernilai positif.
4. False Negatif (FN): kondisi dimana data aktual bernilai positif dan prediksi dari klasifier bernilai negatif.

Dari perhitungan di atas terdapat persamaan turunan lain yang bisa digunakan untuk menghitung performa dari *classifier*.

1. *Accuracy*: menghitung nilai prediksi yang benar oleh *classifier*

$$\text{Accuracy} = (\text{TruePositives} + \text{TrueNegatives}) / \text{TotalData} \quad (2 . 18)$$

## II. LANDASAN TEORI

---

2. *Misclassification Rate*: menghitung nilai kesalahan klasifikasi oleh *classifier*.

$$\text{Misclassification Rate} = (\text{FalsePositives} + \text{FalseNegatives}) / \text{TotalData} \quad (2 . 19)$$

3. *True Positive Rate*: menghitung nilai prediksi bernilai positif ketika data aktual bernilai positif.

$$\text{True Positive Rate (Recall)} = \text{TruePositives} / \text{ActualYes} \quad (2 . 20)$$

4. *False Positive Rate*: menghitung nilai prediksi positif ketika data aktual bernilai negatif.

$$\text{False Positive Rate} = \text{FalsePositive} / \text{ActualNo} \quad (2 . 21)$$

5. *Specificity*: menghitung nilai prediksi negatif ketika data aktual bernilai negatif.

$$\text{Specificity} = \text{TrueNegatives} / \text{ActualNo} \quad (2 . 22)$$

6. *Precision*: menghitung nilai prediksi positif yang benar.

$$\text{Precision} = \text{TruePositives} / \text{PredictedYes} \quad (2 . 23)$$

7. *Prevalence*: menghitung seberapa sering data aktual bernilai positif muncul.

$$\text{Prevalence} = \text{ActualYes} / \text{TotalData} \quad (2 . 24)$$

### 2.1.7 Penggunaan *Library*

Berikut adalah penjelasan dari *library* yang digunakan di dalam penelitian.

#### 2.1.7.1 OpenCV

*Library* yang digunakan adalah OpenCV untuk proses *pre-processing* citra. OpenCV merupakan *library open-source* yang banyak digunakan untuk penelitian terkait proses pengolahan citra dan *computer vision*.

**Tabel 2.1** Tabel fungsi *Library* OpenCV

No	Function	Deskripsi
1	Imgcodecs.imread(String filename)	Mengambil citra dari <i>path</i> yang diisikan ke parameter.
2	Imgproc.cvtColor(Mat src, Mat dst, int code)	Mengubah jenis warna pada citra sesuai yang diinginkan. Parameter fungsi ini terdiri dari Mat asal, Mat tujuan, dan <i>code</i> . <i>Code</i> digunakan untuk memilih tipe konversi citra tersebut, misal <i>grayscale</i> .
3	Imgproc.Canny(Mat image,Mat edges,double threshold1, double threshold2)	Fungsi ini digunakan untuk mendeteksi tepian pada citra menggunakan Canny.
4	Imgproc.GaussianBlur(Mat src, Mat dst, Size ksize, double sigmaX)	Melakukan <i>Gaussian Filter</i> terhadap citra yang dimasukkan ke dalam parameter dengan ukuran <i>kernel</i> dan nilai <i>sigma</i> yang diberikan.
5	Imgproc.threshold(Mat src, Mat dst, double thresh, double maxval, int type)	Melakukan <i>thresholding</i> terhadap seluruh nilai piksel dari citra yang dijadikan masukkan dengan nilai <i>threshold</i> , nilai maksimum, serta jenis metode <i>thresholding</i> yang digunakan, misalnya metode <i>thresholding Otsu</i> .
6	Imgproc.findContours(Mat image, List<MatOfPoint> contours, Mat hierarchy, int mode, int method)	Melakukan pencarian kontur terhadap citra yang dijadikan masukkan.
7	Imgproc.contourArea(Mat contour)	Melakukan perhitungan luas area dari kontur yang diberikan.
8	Imgproc.imwrite(String filename, Mat img)	Menyimpan citra yang diisikan ke parameter ke <i>path</i> yang dijadikan tujuan penyimpanan.

## 2.2 Tinjauan Studi

Terdapat beberapa metode yang dapat digunakan untuk melakukan deteksi mobil pada citra digital. Tabel 2.1 akan menjelaskan tentang metode-metode tersebut beserta hasil dari penerapannya.

**Tabel 2.2** Tabel Tinjauan Studi

No	Judul	Masalah	Metode	Hasil
1	Adhi Prahara, Murinto "Car Detection Based on Road Direction on Traffic Surveillance Image"	Mendeteksi mobil dari semua sudut pandang kamera pengawas.	HOG, SVM	Area jalan diekstraksi untuk menentukan area deteksi dan arah jalanan digunakan untuk menentukan detektor mobil yang akan digunakan oleh Linear-Support Vector Machine (Linear-SVM).
2	Daniel Neumann, Tobias Langner, Fritz Ulbrich, Dorothee Spitta and Daniel Goehring "Online Vehicle Detection using Haar-like, LBP and HOG Feature basedImage Classifiers with Stereo Vision Preselection"	Membandingkan metode ekstraksi fitur.	Haar-like, LBP, dan HOG	Memberikan perbandingan penggunaan metode ekstraksi fitur antara Haar-like, LBP, dan HOG.
3	A.Shakin Banu dan P. Vasuki "Video Based Vehicle Detection Using Morphological Operation and HOG Feature Extraction"	Penggunaan proses morfologi dan HOG untuk mendeteksi objek.	Morphological Operation, HOG	Hasil analisis menerangkan bahwa deteksi kendaraan menggunakan metode ini mencapai tingkat kesuksesan dengan akurasi sekitar 83 persen.

No	Judul	Masalah	Metode	Hasil
4	G. Adhika dan R.R.W. Ken "Penerapan Histogram of Oriented Gradients, Principal Component Analysis, dan AdaBoost untuk Sistem Pengenalan Wajah"	Penggunaan proses HOG, PCA, dan AdaBoost untuk pengenalan wajah.	HOG, PCA, AdaBoost	Hasil analisis menerangkan bahwa pengenalan wajah menggunakan metode PCA dapat meningkatkan akurasi.

Pada penelitian yang dilakukan Adhi Prahara et al., mengusulkan framework dengan metode HOG dan SVM untuk mendeteksi mobil. Pertama, dataset dibagi menjadi data latih dan data uji. Data dikelompokkan berdasarkan sudut pandang (depan atau belakang, kiri atas atau kanan bawah, kanan atas atau kiri bawah, kiri atau kanan). Selanjutnya, untuk menentukan sudut pandang, dilakukan deteksi arah jalan. Terakhir, metode HOG dan SVM digunakan untuk deteksi mobil.

Pada penelitian yang dilakukan Daniel Neumann et al., menerapkan detektor DPM (DeformableParts Model), di mana bagian-bagian dari pola gambar dilatih dengan peningkatan resolusi oleh filter bagian. Untuk dapat mengidentifikasi kendaraan digunakan klasifikasi gambar dan melatihnya pada tampilan belakang kendaraan. Kemudian menerapkan berbagai algoritma *computer vision* yang efisien.

Pada penelitian yang dilakukan A. Shakin Banu et al., mengusulkan framework dengan metode morphological operations dan HOG untuk mendeteksi mobil. Pertama, akan dilakukan pemilihan ROI (*Region of Interest*) untuk mengurangi penggunaan memori. Berdasarkan hasil pre-prosesing, citra akan diterapkan Sobel edge Detection sehingga didapatkan fitur gradien fungsi intensitas dari frame yang kemudian digunakan untuk perhitungan gradien. Terakhir, metode morphological operation akan dilakukan untuk menghapus objek yang tidak seharusnya terdeteksi, sehingga mengurangi piksel dalam frame. HOG kemudian diterapkan untuk ekstraksi fitur dan SVM untuk menentukan apakah mobil atau bukan.

Pada penelitian yang dilakukan G. Adhika et al., menjelaskan bahwa terdapat banyak faktor yang mempengaruhi hasil pengenalan seperti kualitas citra, ekspresi wajah dan kemiripan bentuk dari komponen wajah manusia dimana hal tersebut mempengaruhi fitur HOG yang dihasilkan. Penggunaan metode PCA meningkatkan akurasi dari HOG dan AdaBoost dari 86% menjadi 96% dengan ukuran sel sebesar 8, blok sebesar 16, dan bin sebesar 16.

## 2.3 Tinjauan Objek

Pada bagian ini akan dijelaskan mengenai objek terkait yang akan digunakan dalam penelitian ini.

### 2.3.1 Deteksi Mobil

Mobil adalah kendaraan darat yang digerakkan oleh tenaga mesin, beroda empat atau lebih (selalu genap), biasanya menggunakan bahan bakar minyak (bensin atau solar) untuk menghidupkan mesinnya. Jenis mobil yang terdeteksi terdiri 3 golongan yaitu kendaraan kecil, kendaraan sedang, dan kendaraan besar. Hasil yang diperoleh dapat dimanfaatkan untuk mengetahui lokasi kendaraan yang melewati jalan tersebut dalam suatu interval waktu tertentu. Citra mobil yang digunakan merupakan hasil citra dari sudut horizontal yang diambil dari dasbor mobil. Contoh citra mobil dan hasil visualisasi fitur HOG dapat dilihat pada gambar 2.9.



**Gambar 2.9** Citra Mobil nampak dari belakang. (a) Citra masukkan (b) Visualisasi fitur HOG

## **BAB III**

### **ANALISIS DAN PERANCANGAN SISTEM**

Bab ini memaparkan analisis masalah yang diatasi berserta pendekatan dan alur kerja dari perangkat lunak yang dikembangkan, mengimplementasikan metode yang digunakan dan hasil yang akan ditampilkan.

#### **3.1 Analisis Masalah**

Pada bab 1 telah dijelaskan bahwa penelitian sistem pengenalan mobil masih berkembang dan implementasinya memegang peranan penting dalam bidang transportasi. Pada penelitian ini, digunakan metode *Histogram of Oriented Gradient* untuk mengekstraksi fitur yang sudah melalui tahap *preprocessing*, kemudian dilakukan klasifikasi dengan menggunakan metode *Support Vector Machine*. Klasifikasi yang dilakukan mengelompokkan objek menjadi 2 kelas yaitu mobil dan bukan mobil.

Dataset yang digunakan terdiri dari 3 jenis dataset yang diambil dari sumber yang berbeda. Penelitian ini menggunakan *dataset UIUC Car Image Database*, *GTI Database*, dan *KITTI Database*. Kemudian citra akan diubah menjadi format pgm. Citra yang digunakan ialah citra *grayscale* dengan *depth bit* 8. Ukuran citra mobil yang digunakan berukuran 64 piksel.

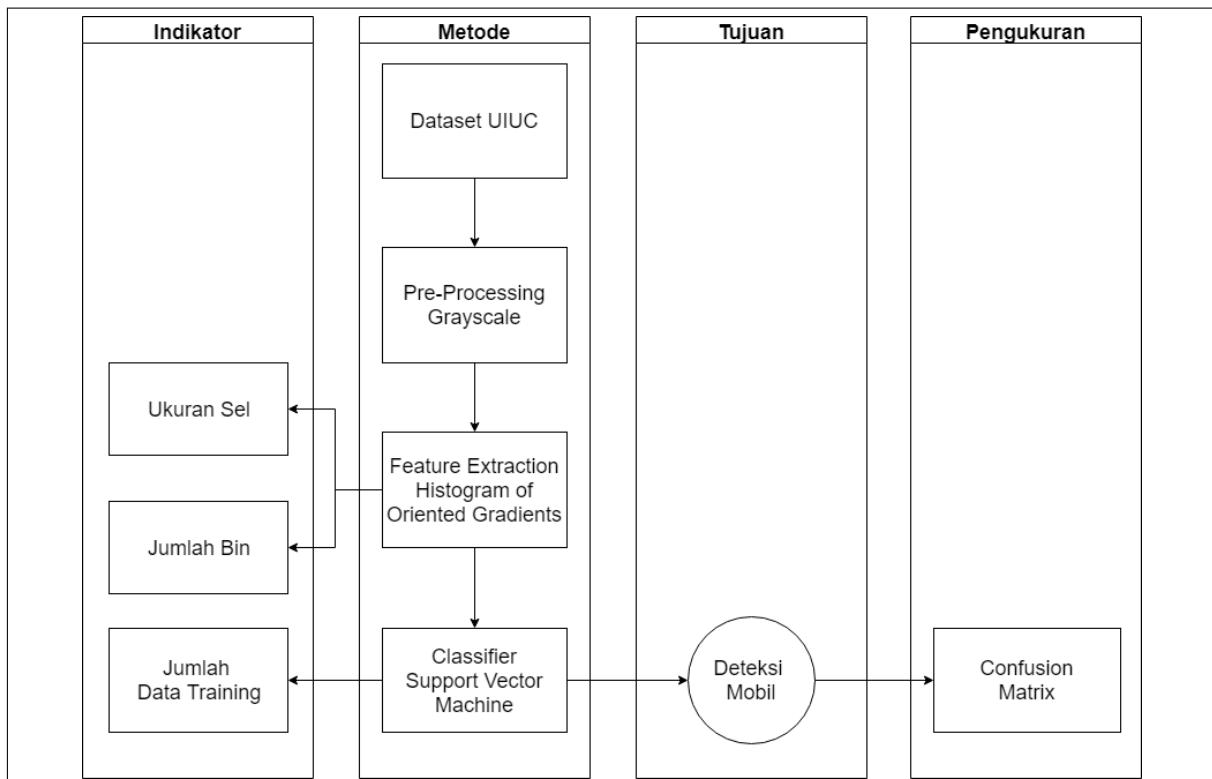
Penelitian diawali dengan melakukan *preprocessing dataset* menjadi citra *grayscale* untuk mempermudah proses ekstraksi fitur. Setelah melalui tahap *preprocessing*, selanjutnya adalah tahap pelatihan. Pada tahap ini, proses latih akan menggunakan metode HOG untuk mendapatkan fitur berupa gradien sudut yang direpresentasikan dalam bentuk fitur vektor. Kemudian hasil ekstraksi fitur tersebut akan dimasukkan ke dalam SVM untuk melakukan klasifikasi mobil dan bukan mobil. Hasil dari proses latih ini berupa matriks fitur yang akan digunakan untuk proses pegujian.

Tahap selanjutnya adalah melakukan proses pengujian. Proses pengujian akan melalui tahap yang sama dengan proses latih, namun pada inisialisasi fitur menggunakan hasil dari fitur yang telah didapatkan sebelumnya. Pada tahap pengujian akan diperhatikan dampak antara penggunaan ROI dan tidak menggunakan ROI pada proses sliding windows. Hasil deteksi mobil adalah penandaan area dimana mobil terdeteksi. Hasil deteksi ini akan dibandingkan dengan hasil deteksi yang dibuat manual oleh manusia.

Arsitektur yang digunakan secara garis besar masih sama dengan arsitektur asli yang dibuat oleh Adhi Prahara [2], namun terdapat beberapa modifikasi untuk keperluan analisis parameter yang akan digunakan seperti orientasi, *HOG channel*, ukuran sel dan ukuran blok.

### 3.2 Kerangka Pemikiran

Berikut ini adalah kerangka pemikiran untuk melakukan deteksi mobil.

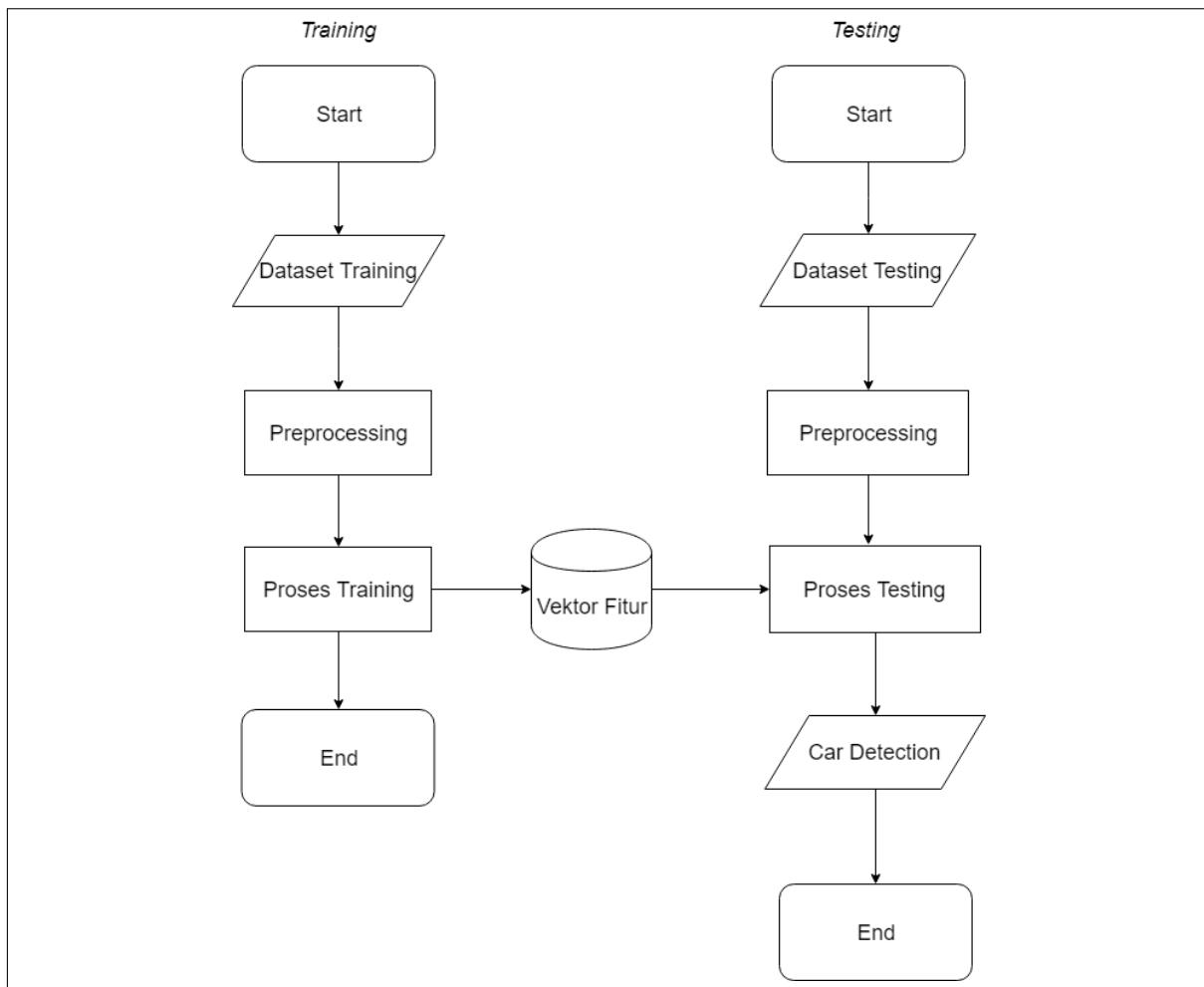


**Gambar 3.1** Kerangka Pemikiran

Berdasarkan gambar 3.1, terdapat beberapa variabel indikator yang memengaruhi hasil dan perlu dilakukan penyesuaian meliputi ukuran sel, jumlah *bin* yang menentukan batasan sudut yang digunakan, dan jumlah data *training* untuk *classifier Support Vector Machine*. Ukuran sel akan mempengaruhi jumlah fitur. Semakin kecil ukuran sel, jumlah fitur akan bertambah. Penelitian ini bertujuan untuk melihat hasil akurasi dari deteksi mobil menggunakan *confusion matrix*.

### 3.3 Urutan Proses Global

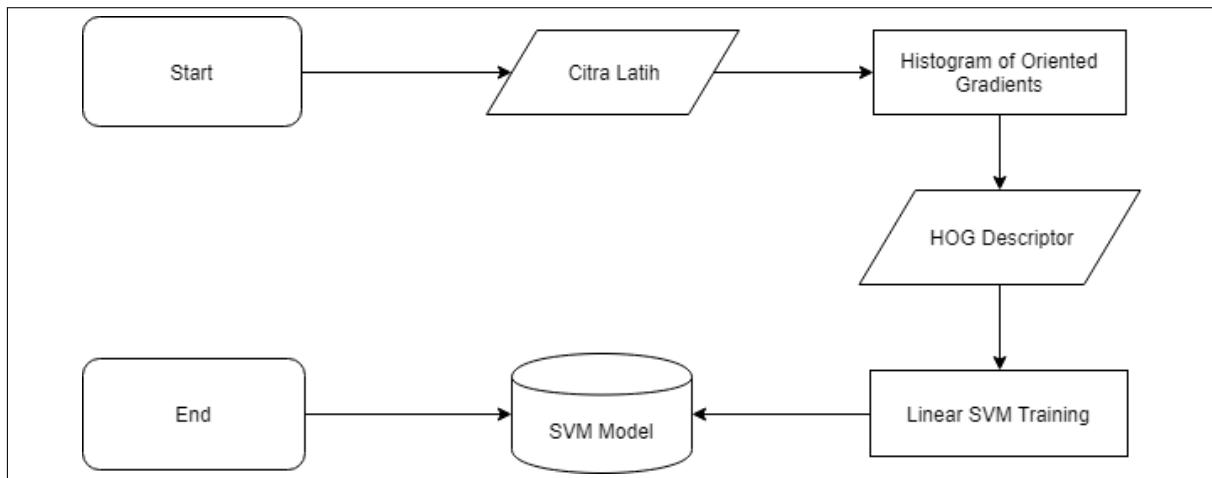
Dalam sistem pengenalan mobil terbagi atas dua proses yaitu proses *training* dan proses *testing*. Proses *training* dilakukan untuk mendapatkan kelas dari objek yang akan dikenali. Proses *testing* dilakukan untuk menghitung hasil yang berupa akurasi dari pengenalan mobil.



**Gambar 3.2 Flowchart Global Sistem Pengenalan Mobil**

Pada gambar 3.2, dataset yang digunakan akan melalui tahap preprocessing untuk mempermudah proses ekstraksi fitur. Citra tersebut kemudian akan dibagi menjadi data latih dan data uji. Proses pelatihan akan menggunakan data latih untuk memperoleh fitur. Sedangkan proses pengujian akan menggunakan data uji dan fitur yang diperoleh dari tahap pelatihan. Hasil dari tahap pengujian adalah hasil deteksi mobil yang dibuat oleh sistem.

### 3.3.1 Proses Training

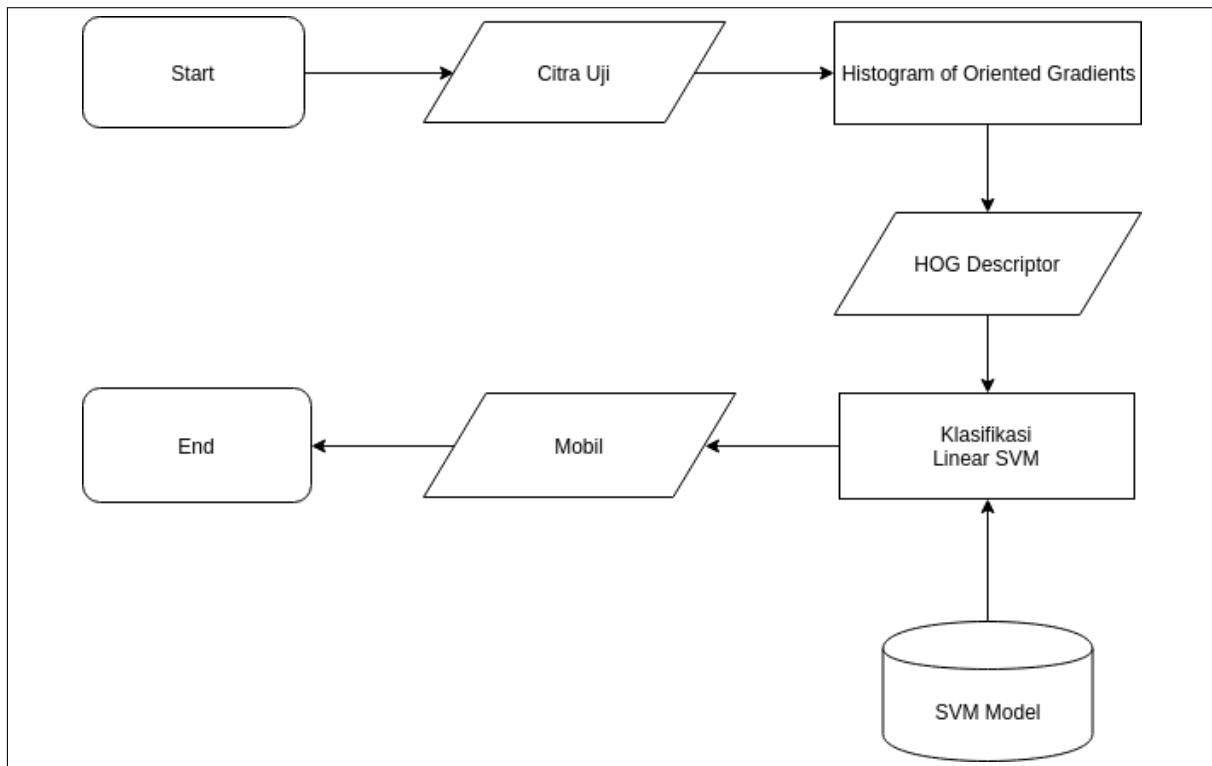


**Gambar 3.3 Flowchart Training Sistem Pengenalan Mobil**

Berikut ini adalah uraian dari *flowchart* pada gambar 3.3 yang dilakukan dalam penelitian ini:

1. Citra yang menjadi masukan yaitu dari data latih yang berisi kumpulan mobil. Citra mobil memiliki ukuran beragam dengan rentang ukuran lebar 200 - 300 piksel dan tinggi 100 - 200 piksel. Citra berupa *grayscale* dengan arah pengambilan citra terdiri dari belakang.
2. *Histogram of Oriented Gradient* berfungsi untuk mengambil fitur dari citra masukan. Hasil dari ekstraksi fitur menggunakan HOG adalah *HOG descriptor*. *HOG descriptor* mendeskripsikan distribusi dari gradien berarah pada suatu area citra.
3. Berdasarkan penelitian yang dilakukan oleh Dalal dan Triggs [9], *HOG* menggunakan ukuran sel  $8 \times 8$  piksel dan  $16 \times 16$  piksel untuk ukuran blok kemudian *bin* yang digunakan pada tahap pembuatan *histogram* adalah 9 (dimulai dari 0 derajat hingga 180 derajat).
4. *Support Vector Machine* (SVM) digunakan untuk klasifikasi fitur ke dalam 2 kelas (mobil dan bukan mobil) berdasar fitur yang sudah didapatkan. Hasil dari klasifikasi ini nantinya akan disimpan ke dalam bentuk berkas.

### 3.3.2 Proses *Testing*



**Gambar 3.4 Flowchart Testing Sistem Pendeksi Mobil**

Pada gambar 3.4 terlihat alur proses *testing*. Pada proses *testing* terdapat beberapa proses yang sama seperti pada proses *training*. Berikut adalah uraian dari *flowchart* pada gambar 3.4 yang dilakukan dalam penelitian ini:

1. Citra pengujian yang digunakan didapatkan dari *dataset* dan [www.youtube.com](http://www.youtube.com), penggunaan dari dataset ini sesuai dengan perizinan dari institusi yang bersangkutan.
2. Citra yang akan menjadi input dari *HOG* adalah citra hasil dari grayscale citra mobil yang sudah dilakukan pada tahapan *preprocessing*.
3. Ukuran dari sel dan blok yang digunakan untuk proses ekstraksi fitur dengan menggunakan *HOG* adalah sama dengan yang digunakan ketika pada tahap *training*.
4. Pada tahap *testing*, model SVM yang digunakan adalah berkas hasil keluaran dari SVM pada tahap *training*.
5. Hasil keluaran akan berupa penandaan mobil yang berhasil dikenali oleh sistem.

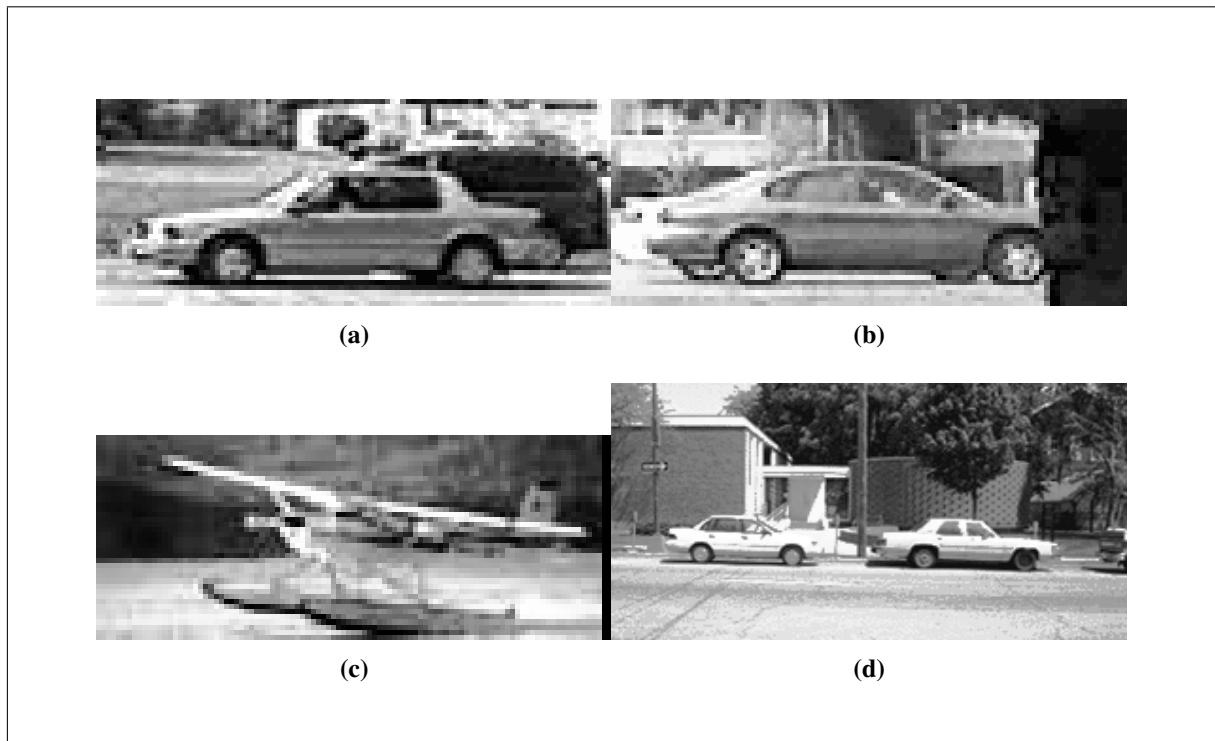
### 3.4 Analisis Manual

Bagian ini melakukan analisis tahapan proses dengan melakukan perhitungan manual. Analisis untuk proses pelatihan dan pembelajaran pada penelitian ini berjumlah yaitu mobil dan bukan mobil.

### 3.4.1 Dataset

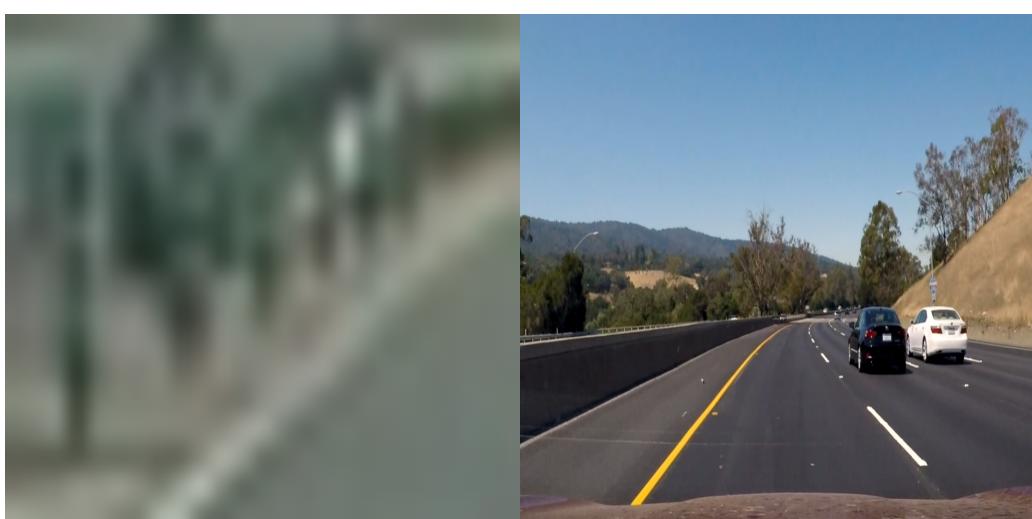
Dataset yang digunakan terdiri dari 3 jenis data latih yang diambil dari sumber yang berbeda. Penelitian ini menggunakan *dataset UIUC Car Image Database*, *GTI Database*, dan *KITTI Database*. Arah pengambilan kamera untuk *dataset* dilakukan dari posisi *horizontal*.

Dataset *UIUC Car Image Database* diperoleh dari <http://cogcomp.org/Data/Cars/>. Citra yang diambil dari *dataset* merupakan citra *grayscale* dengan 8 channel warna. Pada UIUC terdapat 1328 citra mobil. Dari total *dataset* yang digunakan, terdiri dari 1050 citra latih dan 278 citra uji. Citra latih dibagi menjadi 2 macam, yaitu 550 citra mobil dan 500 bukan mobil. Citra mobil yang digunakan untuk citra latih diambil dari 2 arah yaitu kiri dan kanan. Citra uji terdiri dari beragam citra mobil yang terdiri dari 2 jenis yaitu 170 citra *single-scale* dan 108 citra *multi-scale* dengan ukuran panjang berkisar 100 piksel sampai 300 piksel dan lebar 75 piksel sampai 180 piksel. Pembagian data untuk pelatihan dan pengujian sudah dilakukan dari sumbernya.



**Gambar 3.5** Contoh *Dataset UIUC*. (a) Data Latih Mobil (Kiri) (b) Data Latih Mobil (Kanan) (c) Data Latih Bukan Mobil (d) Data Uji

*Dataset GTI Database* diperoleh dari [http://www.gti.ssr.upm.es/data/Vehicle\\_database.html](http://www.gti.ssr.upm.es/data/Vehicle_database.html). Citra yang diambil dari *dataset* merupakan citra RGB dengan 24 channel warna. Untuk proses pelatihan dan pengujian, ada total 2826 data citra. Data latih yang digunakan diambil dari belakang mobil dengan 3 sudut pengambilan yaitu belakang lurus, belakang kiri, dan belakang kanan. Pengambilan citra dari belakang dilakukan dengan jarak dekat dan jarak jauh. Pembagian data untuk pelatihan dan pengujian dilakukan manual oleh peneliti.



**Gambar 3.6** Contoh *Dataset GTI*. (a) Data Latih Mobil (kiri) (b) Data Latih Mobil (kanan) (c) Data Latih Mobil (jauh) (d) Data Latih Mobil (dekat) (e) Data Latih Bukan Mobil (f) Data Uji

*Dataset KITTI Database* diperoleh dari <http://www.cvlibs.net/datasets/kitti/>. *Dataset KITTI* merupakan data latih yang berupa mobil yang sama dengan GTI, namun dengan kondisi pencahayaan dan arah yang berbeda. Citra yang diambil dari *dataset* merupakan citra RGB dengan 24 channel warna. Untuk proses pelatihan, ada total 5966 data citra yang diambil dari *dataset*.

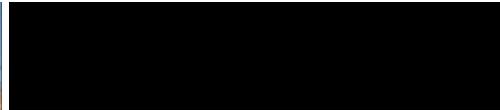


**Gambar 3.7** Contoh *Dataset KITTI*, data data latih mobil

Ukuran setiap data latih akan diubah menjadi berukuran 64 x 64 piksel. Format citra dari *dataset* ini akan diubah *Portable Gray Map* (PGM) dengan 8 bit kedalaman *depth*. Data latih dibagi menjadi 2 jenis yaitu mobil (kondisi utuh, tidak ada bagian mobil yang terpotong) dan bukan mobil (jalan, motor, rumah, pejalan kaki, dan sebagainya). Pembagian data latih terdiri dari 4 jenis pengambilan citra, yaitu jauh, dekat, kiri, dan kanan.

Untuk citra uji, terdiri dari mobil yang berbeda - beda. Citra data uji diambil dari [www.youtube.com](http://www.youtube.com) yang akan diproses secara citra digital. Dari video akan diambil beberapa frame untuk dideteksi keberadaan mobil. Ukuran untuk setiap citra uji beragam dengan panjang antara 110 piksel sampai 301 piksel dan lebar 75 piksel sampai 179 piksel dengan *depth bit* 24.

Pada dataset juga terdapat *template* untuk *Region of Interest* dimana berupa citra biner yang berfungsi untuk menandai lokasi dimana mobil akan melaju. ROI digunakan pada saat proses *testing*. Penggunaan ROI ditandai dengan nilai 1 untuk *foreground* dan 0 untuk *background*.



**Gambar 3.8** Contoh ROI

Analisis selanjutnya akan dilakukan untuk menangani kasus dimana bagian dari mobil yang terdapat dalam citra uji hanya sebagian, terdapat lebih dari 1 mobil dalam 1 citra uji, dan ukuran objek berdasar jarak kamera dengan objek penelitian (mobil). Mobil yang akan terdeteksi memiliki ukuran terkecil  $64 \times 64$  piksel.

#### 3.4.2 Tahap Pendektsian Lokasi Mobil

Seperti dijelaskan sebelumnya, sistem pendektsian mobil terdiri dari beberapa tahap. Citra input pertama - tama dibuat menjadi *grayscale*, kemudian melalui tahap ekstraksi fitur dengan *Histogram of Oriented Gradients*, dan terakhir fitur yang diperoleh kemudian diklasifikasi dengan *Support Vector Machines*. Berikut adalah skema alur dari tahap pendektsian lokasi mobil.

##### 3.4.2.1 *Grayscale*

Proses pertama yaitu mengubah citra masukan menjadi citra *grayscale*. Tujuan dari *grayscaling* citra adalah untuk menghilangkan informasi warna dari setiap piksel citra. Dalam proses deteksi mobil dengan HOG, input warna tidak diperlukan karena warna tidak diperlukan oleh metode HOG. Di bawah merupakan contoh matriks citra asli dengan 3 *channel* warna yaitu *Red*, *Green*, dan *Blue* berukuran  $5 \times 5$  piksel yang diambil dengan menggunakan *image tools* dari aplikasi *MatLab*.

### III. ANALISIS DAN PERANCANGAN SISTEM

R: 22 G: 26 B: 29	R: 24 G: 27 B: 32	R: 30 G: 33 B: 38	R: 32 G: 35 B: 42	R: 34 G: 39 B: 45
R: 25 G: 28 B: 33	R: 29 G: 32 B: 39	R: 32 G: 36 B: 45	R: 23 G: 27 B: 38	R: 23 G: 27 B: 39
R: 36 G: 30 B: 33	R: 29 G: 32 B: 37	R: 96 G: 103 B: 109	R: 124 G: 131 B: 139	R: 104 G: 109 B: 115
R: 26 G: 27 B: 31	R: 58 G: 59 B: 64	R: 103 G: 108 B: 112	R: 133 G: 138 B: 144	R: 155 G: 164 B: 169
R: 27 G: 28 B: 30	R: 41 G: 42 B: 46	R: 119 G: 124 B: 127	R: 108 G: 113 B: 119	R: 144 G: 153 B: 158

Gambar 3.9 Matriks Citra Asal berukuran  $5 \times 5$

Dari contoh nilai citra RGB di atas, akan diubah menjadi nilai *grayscale* menggunakan rumus (2 . 1) dengan perhitungan sebagai berikut:

$$\begin{aligned} Matriks[3,3] &= (0.299 * 133) + (0.587 * 138) + (0.114 * 144) \\ &= 137.189 \approx 137 \end{aligned}$$

Perhitungan di atas diterapkan pada seluruh piksel pada citra asal yang kemudian menghasilkan matriks citra berukuran  $5 \times 5$  dengan satu nilai derajat keabuan.

25	27	33	35	38
28	32	36	27	27
29	32	102	130	108
27	59	107	137	162
28	42	123	112	151

Gambar 3.10 Matriks Citra Hasil Grayscale

#### 3.4.3 Histogram of Oriented Gradient

Hasil citra *grayscale* dari *preprocessing* kemudian digunakan untuk input HOG. HOG bertujuan untuk mengambil fitur dari citra masukkan. Hasil dari proses ini adalah vektor fitur yang berbentuk matriks fitur dimana ukurannya ditentukan berdasar jumlah *bin*, ukuran sel dan blok. Ukuran sel, blok, dan jumlah *bin* akan dianalisis pada bab selanjutnya.

*Pseudocode:*

1. **Masukan:** Citra *grayscale* hasil resize.
2. Menentukan jumlah *bin*, ukuran sel, dan ukuran blok.
3. Menghitung nilai gradien dan arah gradien untuk setiap piksel dari citra masukan dengan persamaan 2 . 12 dan 2 . 13.
4. Menghitung nilai magnitude gradien dan arah gradien untuk setiap piksel dari citra masukan menggunakan hasil dari tahap 2 sebelumnya.
5. Membagi citra masukan ke dalam ukuran sel yang sudah ditentukan.
6. Untuk setiap sel, lakukan proses vote untuk setiap *c bin* terhadap sudutnya dengan menggunakan magnitude gradien dan arah gradien dari tahap 4.
7. Untuk setiap blok (*a x b* sel), gabungkan histogram *bin* ke dalam satu matriks baris, sehingga didapat ukuran [*a x b x c*] x 1 matriks untuk proses normalisasi.
8. Menggunakan rumus algoritme *L2 Norm* dengan menggunakan persamaan 2 . 14 dalam proses normalisasi kemudian lakukan proses normalisasi berupa *sliding window* dengan melakukan pergeseran sebesar ukuran 1 sel ke arah vertikal dan horizontal dari hasil tahap 6.
9. Untuk setiap hasil normalisasi, gabungkan seluruh matriks baris sehingga membentuk sebuah fitur yang besar dengan ukuran (jumlah pergeseran vertikal) x (jumlah pergeseran horizontal) x (*a x b x c*).
10. **Keluaran:** Vektor Fitur dari tahap normalisasi.

Berdasarkan *pseudocode* di atas, akan dilakukan langkah - langkah untuk menghitung matriks fitur vektor. Berdasarkan penelitian [9], ukuran sel yang digunakan adalah  $8 \times 8$  piksel. Ukuran sel akan mempengaruhi jumlah fitur. Semakin kecil jumlah sel, maka jumlah fitur akan bertambah. Pada proses *Histogram of Oriented Gradients*, contoh masukan untuk proses ini berupa citra *grayscale* yang sudah diproses melalui *resize* citra menjadi ukuran  $4 \times 4$  piksel.

89	92	88	92
90	88	90	86
91	90	90	94
91	122	91	122

**Gambar 3.11** Contoh sebagian matriks citra hasil *preprocessing* dalam ukuran  $4 \times 4$  piksel

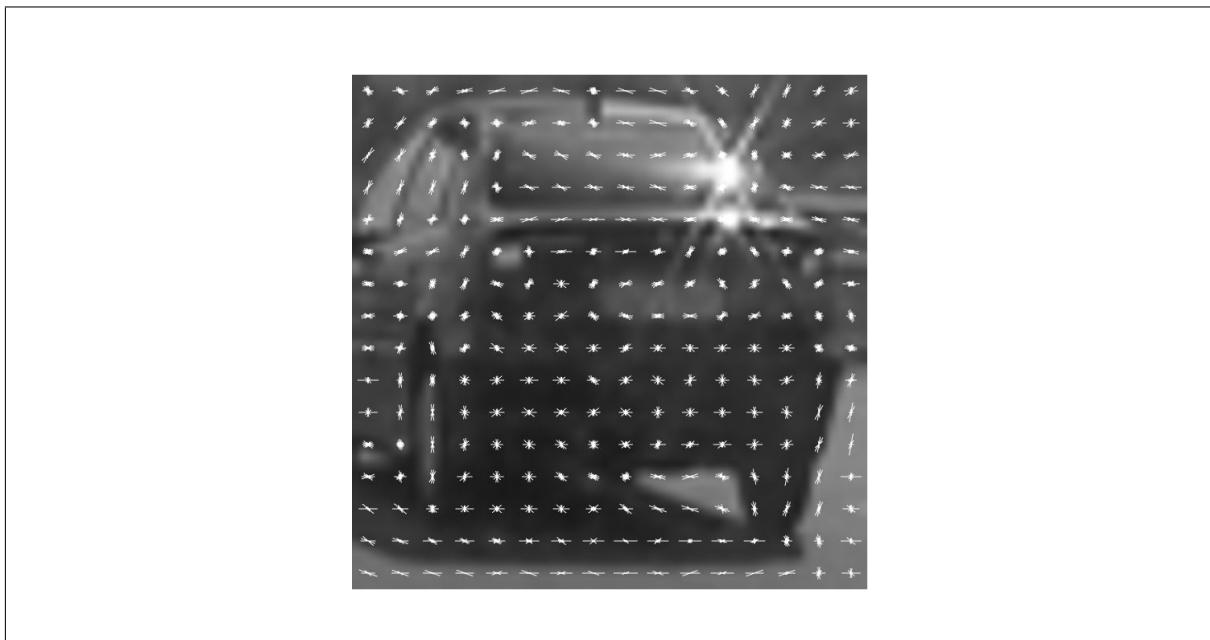
Langkah pertama adalah menghitung nilai gradien dari posisi vertikal dan horizontal untuk setiap piksel menggunakan persamaan (2 . 10) dan (2 . 11).

$$G_x(2,5) = 89 - 89 = 0$$

$$G_y(2,5) = 85 - 122 = -37$$

Langkah pertama diterapkan untuk seluruh piksel pada citra. Berikut contoh dari

visualisasi fitur HOG.



**Gambar 3.12** Visualisasi Fitur HOG dengan ukuran sel 4 x 4 piksel

Untuk setiap piksel, hitung *magnitude* gradien menggunakan persamaan (2 . 12) dan arah gradien menggunakan persamaan (2 . 13).

$$M(2,5) = \sqrt{0^2 + (-37)^2} = 37$$

$$\theta(2,5) = \arctan \frac{-37}{0} = 90$$

Setelah itu, tentukan ukuran sel, ukuran blok dan jumlah *bins*. Pada awalnya, ukuran sel akan dipilih sebesar  $2 \times 2$  piksel dan ukuran blok sebesar 2 kali lipat daripada ukuran sel yaitu  $4 \times 4$  piksel. Lakukan perhitungan *Histogram of Oriented Gradient* untuk semua sel dengan melakukan *voting* arah gradien berdasarkan *magnitude* gradien, dimana arah gradien akan menjadi jumlah sudut *bin*, dan *magnitude* gradien akan menjadi bobot nilai yang kemudian dimasukkan berdasarkan persentase. Untuk contoh perhitungan analisis kali ini jumlah *bins* yang dipakai sebanyak 9 buah, sehingga didapat nilai sudut setiap *histogram bin* yaitu  $180 / 4 = 45$ . Berikut merupakan contoh proses *voting* untuk piksel dengan koordinat (2,5).

$$M(2,5) = 37$$

$$\theta(2,5) = 90$$

Sehingga untuk *bin* dengan sudut 90 akan mendapat nilai bobot sebesar 37 yang didapatkan dari nilai gradien *magnitude*-nya. Lakukan proses tersebut untuk setiap sel sampai setiap sel akan mempunyai *Histogram of Oriented Gradient*. Kemudian untuk setiap blok akan

dilakukan normalisasi dengan menggabungkan hasil histogram dari setiap sel dalam bloknya. Adapun proses normalisasi dapat menggunakan 4 algoritme yaitu, *L1-Norm*, *L1-Sqrt*, *L2-Norm*, dan *L2-Hys*. Penelitian ini akan menggunakan algoritme normalisasi *L2-Norm* karena berdasarkan penelitian sebelumnya, hasil yang didapat lebih baik dari algoritme lainnya [2]. Di bawah adalah contoh perhitungan normalisasi untuk blok pertama:

1	0	4	0
0	0.76	36.24	0
0	0	40	0
0	1.65	36.35	0

**Gambar 3.13** Matriks hasil Perhitungan Histogram untuk seluruh sel

Berdasarkan matriks pada gambar 3.13. Elemen matriks yang akan kita gunakan dalam perhitungan normalisasi ini adalah seluruh elemen baris pertama dan baris kedua.

$$L2_{Norm} = \sqrt{1^2 + 0^2 + 4^2 + 0^2 + \dots + 36.5^2 + 0^2} = 51,360978379$$

Setiap nilai pada histogram dari sel dalam blok tersebut akan dibagi dengan nilai hasil normalisasinya. Berikut merupakan contoh matriks hasil perhitungan histogram baris pertama kolom 1-4.

$$\begin{bmatrix} 0,019470034 & 0 & 0,077880136 & 0 \end{bmatrix}$$

Lakukan proses normalisasi untuk setiap blok dengan menggeser secara horizontal sejauh 1 kali ukuran sel (setengah ukuran blok) dan kemudian secara vertikal sejauh 1 kali ukuran sel sampai blok tersebut sudah berada pada akhir dari bagian citra. Kemudian hasil dari proses normalisasi akan disusun menjadi matriks dengan kolom sebesar *jumlah bin* × *lebar blok dalam satuan sel* × *jumlah pergeseran horizontal* dan jumlah baris sebesar *jumlah pergeseran vertikal* × *tinggi blok dalam satuan sel*, dengan perhitungan tersebut, dalam analisa saat ini didapatkan ukuran matriks sebesar 6 × 8. Dalam analisa ini, hasil keluaran dari metode *Histogram of Oriented Gradient* ada sebanyak 48 fitur. Matriks inilah yang akan dijadikan sebagai masukan bagi metode *Machine Learning* yang akan digunakan dalam penelitian ini.

#### 3.4.4 Support Vector Machine

*HOG descriptor* yang dihasilkan dari perhitungan metode *Histogram of Oriented Gradient* akan digunakan sebagai bahan masukan *Support Vector Machine*. Masukkan untuk metode SVM ini berupa matriks fitur berukuran jumlah data x jumlah data. *Support Vector Machine* yang akan digunakan dalam penelitian ini akan menggunakan *library* dari Weka

SVM. *Support Vector Machine* termasuk dalam algoritme *supervised learning*. Konsep dasar dari metode ini adalah untuk menemukan sebuah *separating hyperplane* (bidang) yang dapat memisahkan dua kelas sebagai keputusan klasifikasi. Dalam penelitian ini mobil yang akan dikenali adalah semua jenis mobil dari arah kiri dan kanan.

**Tabel 3.1** Tabel Contoh Data Latih

Data	f1	f2	f3	f4	Class
A1	5.1	3.5	1.4	0.2	1
A2	4.9	3.0	1.4	0.2	1
A3	7.0	3.2	4.7	1.4	-1
A4	6.4	3.2	4.5	1.5	-1

Pada matriks fitur di atas, 1 menandakan kelas mobil dan -1 menandakan kelas bukan mobil. Berdasarkan matriks fitur tersebut, akan dibuat matriks RBF dengan ukuran sebanyak data. Misal terdapat 4 data yang digunakan maka ukuran matriks RBF adalah  $4 \times 4$ . Matriks RBF digunakan untuk menghitung *dot product* dari masing-masing data. Pada perhitungan ini nilai  $\sigma$  yang digunakan adalah 1.

$$K(A1, A1) = \exp\left(-\frac{|A1-A1|^2}{2\sigma^2}\right) = \exp\left(-\frac{|5.1-5.1|^2+|3.5-3.5|^2+|1.4-1.4|^2+|0.2-0.2|^2}{2(1)^2}\right) \\ = 1$$

$$K(A1, A2) = \exp\left(-\frac{|A1-A2|^2}{2\sigma^2}\right) = \exp\left(-\frac{|5.1-4.9|^2+|3.5-3.0|^2+|1.4-1.4|^2+|0.2-0.2|^2}{2(1)^2}\right) \\ = 0.8650222931107414$$

$$K(A1, A3) = \exp\left(-\frac{|A1-A3|^2}{2\sigma^2}\right) = \exp\left(-\frac{|5.1-7.0|^2+|3.5-3.2|^2+|1.4-4.7|^2+|0.2-1.4|^2}{2(1)^2}\right) \\ = 3.3046824003738314E^{-4}$$

$$K(A1, A4) = \exp\left(-\frac{|A1-A4|^2}{2\sigma^2}\right) = \exp\left(-\frac{|5.1-6.4|^2+|3.5-3.2|^2+|1.4-4.5|^2+|0.2-1.5|^2}{2(1)^2}\right) \\ = 0.001444488499020542$$

Dari perhitungan *dot product* di atas maka akan terbentuk matriks RBF seperti pada tabel 3.4.

**Tabel 3.2** Tabel Perhitungan Matriks RBF

	A1	A2	A3	A4
A1	1	0.865022293	3.30E-4	1.44E-03
A2	0.865022293	1	2.27E-4	1.11E-03
A3	3.30E-4	2.27E-4	1	0.814647316
A4	1.44E-03	1.11E-03	0.814647316	1

Setelah mendapatkan matriks RBF, maka dapat membuat persamaan linear untuk mendapatkan nilai  $\alpha$  dan  $b$  (bias) berdasarkan persamaan *hyperplane* SVM yaitu persamaan 2 . 17.

Contoh perhitungan:

$$\begin{aligned}
 (1)\alpha_1 + (1)\alpha_2 + (-1)\alpha_3 + (-1)\alpha_4 + (0)b &= 0 \\
 (1)\alpha_1 + (0.865022293)\alpha_2 + (-3.30E^{-4})\alpha_3 + (-1.44E^{-3})\alpha_4 + b &= 1 \\
 (0.865022293)\alpha_1 + (1)\alpha_2 + (-2.27E^{-4})\alpha_3 + (-1.11E^{-3})\alpha_4 + b &= 1 \\
 (3.30E^{-4})\alpha_1 + (2.27E^{-4})\alpha_2 + (-1)\alpha_3 + (-0.814647316)\alpha_4 + b &= -1 \\
 (1.44E^{-3})\alpha_1 + (1.11E^{-3})\alpha_2 + (-0.814647316)\alpha_3 + (-1)\alpha_4 + b &= -1
 \end{aligned}$$

Untuk mendapatkan solusi  $(\alpha_1, \alpha_2, \alpha_3, \alpha_4, b)$  dari sistem persamaan linear di atas dapat menggunakan *library* JAMA. Berikut adalah solusi yang diperoleh dari sistem persamaan di atas.

**Tabel 3.3** Tabel nilai  $\alpha$  dan  $b$

$\alpha_1$	$\alpha_2$	$\alpha_3$	$\alpha_4$	$b$
0.544855	0.543123	0.541044	0.546933	-0.013700

Nilai  $\alpha$  dan  $b$  yang telah diperoleh kemudian disimpan untuk keperluan proses klasifikasi. Misal terdapat data uji sebagai berikut:

**Tabel 3.4** Tabel Contoh Data Uji

Data	f1	f2	f3	f4	Class
T1	4.6	3.1	1.5	0.2	?

Dalam proses pengujian, langkah yang ditempuh mirip dengan proses pelatihan. Pertama adalah membentuk matriks RBF untuk pengujian dengan menggunakan data latih pada tabel 3.3. Contoh perhitungan adalah sebagai berikut:

$$\begin{aligned}
 K(T1, A1) &= \exp\left(-\frac{|T1-A1|^2}{2\sigma^2}\right) = \exp\left(-\frac{|4.6-5.1|^2+|3.1-3.5|^2+|1.5-1.4|^2+|0.2-0.2|^2}{2(1)^2}\right) \\
 &= 0.8105842459701871
 \end{aligned}$$

$$\begin{aligned}
 K(T1, A2) &= \exp\left(-\frac{|T1-A2|^2}{2\sigma^2}\right) = \exp\left(-\frac{|4.6-4.9|^2+|3.1-3.0|^2+|1.5-1.4|^2+|0.2-0.2|^2}{2(1)^2}\right) \\
 &= 0.9464851479534836
 \end{aligned}$$

$$\begin{aligned}
 K(T1, A3) &= \exp\left(-\frac{|T1-A3|^2}{2\sigma^2}\right) = \exp\left(-\frac{|4.6-7.0|^2+|3.1-3.2|^2+|1.5-4.7|^2+|0.2-1.4|^2}{2(1)^2}\right) \\
 &= 1.6247279265951668E^{-4}
 \end{aligned}$$

$$\begin{aligned}
 K(T1, A4) &= \exp\left(-\frac{|T1-A4|^2}{2\sigma^2}\right) = \exp\left(-\frac{|4.6-6.4|^2+|3.1-3.2|^2+|1.5-4.5|^2+|0.2-1.5|^2}{2(1)^2}\right) \\
 &= 9.396529058360945E^{-4}
 \end{aligned}$$

Dari perhitungan *dot product* maka akan terbentuk matriks RBF uji seperti pada tabel 3.7 di bawah ini.

### III. ANALISIS DAN PERANCANGAN SISTEM

---

**Tabel 3.5** Tabel Perhitungan Matriks RBF Uji

	<b>A1</b>	<b>A2</b>	<b>A3</b>	<b>A4</b>
<b>T1</b>	0.810584245	0.946485147	1.62E-4	9.39E-4

Setelah mendapatkan matriks RBF uji maka proses klasifikasi dapat dilakukan dengan menggunakan persamaan 2.15. Perhitungan klasifikasi adalah sebagai berikut:

$$\begin{aligned}f(x) &= \text{sign}[(0.810584 * 0.810584245 * 1.0) + (0.946485 * 0.946485147 * 1.0) + (1.624727 * \\&\quad 1.62E^{-4} * (-1.0)) + (9.396529 * 9.396529058E^{-4} * (-1.0)) + (-0.013700)] \\&= 1\end{aligned}$$

Setelah dilakukannya perhitungan manual antara data uji terhadap data latih, nilai yang diperoleh adalah 1 yang artinya data uji masuk ke dalam kelas 1 (positif). Jika hasil perhitungan bernilai -1 maka data uji masuk ke dalam kelas -1 (negatif). Misal dalam pengujian di atas kelas 1 adalah mobil dan kelas -1 adalah bukan mobil, maka data uji tergolong kelas mobil.

## **BAB IV**

### **IMPLEMENTASI DAN PENGUJIAN**

Pada bab ini akan menjelaskan mengenai proses implementasi dan pengujian terhadap sistem yang telah dibangun berdasarkan penjelasan pada bab sebelumnya.

#### **4.1 Lingkungan Implementasi**

Pada lingkungan implementasi, akan dijelaskan mengenai perangkat yang digunakan dalam proses pembangunan sistem baik dari perangkat keras maupun perangkat lunak yang digunakan.

##### **4.1.1 Spesifikasi Perangkat Keras**

Spesifikasi komputer yang digunakan dalam pembuatan dan pengujian aplikasi deteksi bangunan pada citra udara dengan *Convolutional Neural Network* adalah sebagai berikut:

1. Komputer dengan *processor* Intel Core i5-7400U CPU @ 2.70GHz
2. *Harddisk* dengan kapasitas 1 TB
3. RAM 16 GB
4. VGA NVDIA GeForce GT 1050M

##### **4.1.2 Lingkungan Perangkat Lunak**

Spesifikasi perangkat lunak yang digunakan dalam pembangunan aplikasi deteksi bangunan pada citra udara dengan *Convolutional Neural Network* adalah sebagai berikut: 7

1. *Virtual Machine*: VMware Workstation 15
2. Sistem Operasi: Ubuntu 18.04
3. IDE: Geany 1.34.1
4. Development Tool: Python 3.6.8 64-bit

#### **4.2 Daftar Kelas dan Metode**

Pada bagian ini, akan dijelaskan mengenai *class* dan *method* yang digunakan dalam penelitian ini.

#### 4.2.1 Daftar Class dan Method hog\_util\_functions

Berikut adalah tabel berisi *method* pada *class* *hog\_util\_functions*. *Class* *hog\_util\_functions* digunakan untuk mengolah citra menggunakan metode *Histogram of Oriented Gradients*.

**Tabel 4.1** Daftar Metode pada Kelas *hog\_util\_functions*

No	Metode	<i>Input</i>		<i>Output</i>	Keterangan
		Tipe	Variabel		
1	convert_rgb_color	string string	img conv	image	Untuk mengubah <i>color space</i> .
2	get_hog_features	string int int int boolean boolean	img orient pix_per_cell cell_per_block vis feature_vec	int[]	Untuk mengambil fitur menggunakan metode HOG.
3	single_image_features	string string int[] int int int int int boolean boolean boolean	img color_space spatial_size hist_bins orient pix_per_cell cell_per_block hog_channel spatial_feat hist_feat hog_feat	int[]	Untuk mengekstrak fitur dari satu jendela gambar.

#### 4.2.2 Daftar Class dan Method find\_cars

Berikut adalah tabel berisi *method* pada *class* *find\_cars*. *Class* *find\_cars* digunakan untuk mencari objek berupa mobil pada frame masukan.

**Tabel 4.2** Daftar Metode pada Kelas *find\_cars*

No	Metode	<i>Input</i>		<i>Output</i>	Keterangan
		Tipe	Variabel		
1	load_classifier	string	pickle_file	int[]	Untuk memuat parameter HOG dan SVM dari <i>pickle file</i> .
2	draw_labeled_bboxes	string string int[][][] int	img labels color thick	image	Untuk membuat kotak penanda lokasi mobil yang terdeteksi.

3	find_cars	string int int int string int string[] int	img ystart ystop scale svc X_scaler params cells_per_step	int[]	Untuk mengekstrak fitur menggunakan sub-sampling HOG dan membuat prediksi.
---	-----------	---	--	-------	--

#### 4.2.3 Daftar *Class* dan *Method* train\_hog\_classifier

Berikut adalah tabel berisi *method* pada *class* train\_hog\_classifier. *Class* train\_hog\_classifier digunakan untuk proses pelatihan dari dataset.

**Tabel 4.3** Daftar Metode pada Kelas *train\_hog\_classifier*

No	Metode	<i>Input</i>		<i>Output</i>	Keterangan
		Tipe	Variabel		
1	load_data_sets			string[] string[]	Untuk memuat data dari <i>dataset</i> .
2	save_data_sets	string image[] image[]	pickle_file cars notcars		Untuk menyimpan daftar gambar mobil dan bukan mobil yang sudah dimuat.
3	load_cars_norcars	string	data_file	image[][] image[][]	Untuk memuat <i>dataset</i> dari <i>pickle file</i> .
4	save_classifier_data	string int int int int	pickle_file X_train Y_train X_test Y_test		Untuk menyimpan dataset yang sudah dibagi menjadi data latih dan data uji.
5	save_classifier	string string int string[]	pickle_file svc X_scaler params		Untuk menyimpan parameter HOG.
6	extract_features	image string[]	imgs params		Untuk mengekstrak fitur dari citra.

#### 4.3 Implementasi Perangkat Lunak

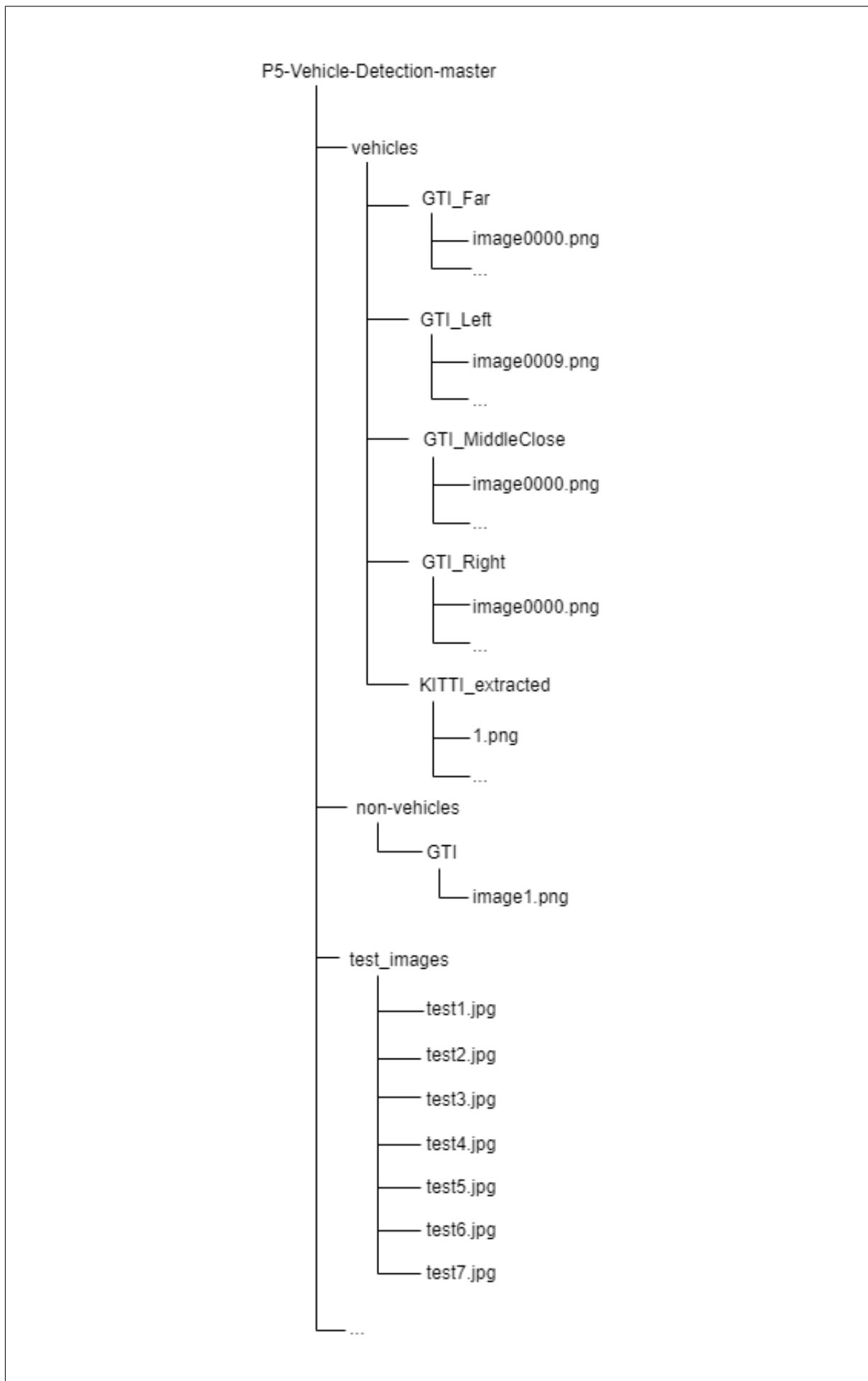
Pada implementasi perangkat lunak dilakukan menurut analisis yang telah disusun pada BAB III.

#### 4.3.1 Implementasi Pengambilan Dataset

*Dataset* GTI diperoleh dari [http://www.gti.ssr.upm.es/data/Vehicle\\_database.html](http://www.gti.ssr.upm.es/data/Vehicle_database.html) dan *Dataset* KITTI dari <http://www.cvlibs.net/datasets/kitti/>. Gambar 4.1 menunjukkan daftar *folder* penyimpanan *Dataset*.

#### IV. IMPLEMENTASI DAN PENGUJIAN

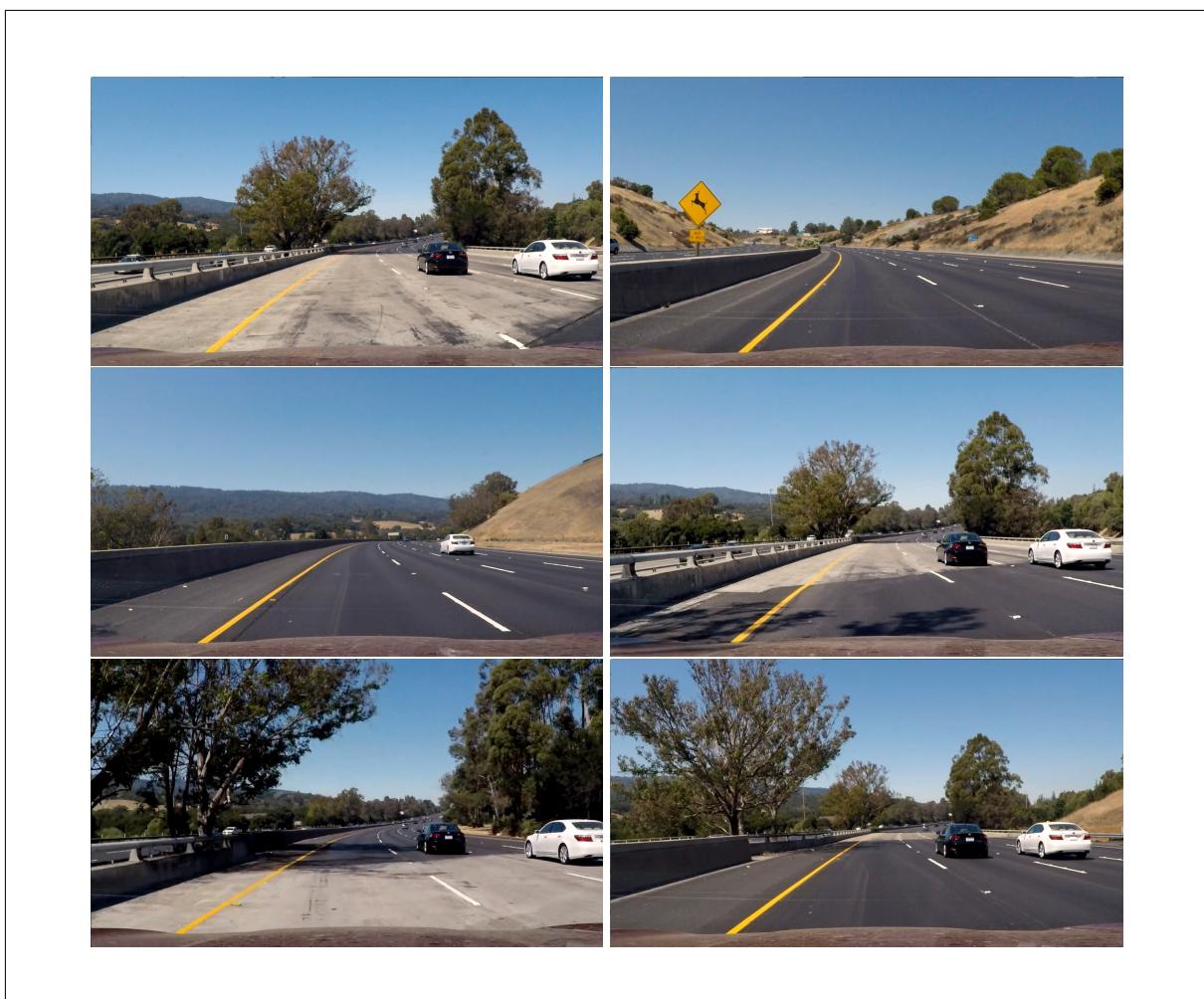
---



Gambar 4.1 Daftar folder *Dataset*

Citra untuk proses pengenalan kendaraan terbagi menjadi 2 yaitu *vehicles* dan *non-vehicles*. Data citra *vehicles* dibagi menjadi 5 macam yaitu *GTI\_Far*, *GTI\_Left*, *GTI\_MiddleClose*, *GTI\_Right*, dan *KITTI\_extracted*. *GTI\_Far* berisi citra yang diambil dari jauh. *GTI\_Left* berisi citra yang diambil dari arah kiri kendaraan. *GTI\_MiddleClose* berisi citra yang diambil dari dekat. *GTI\_Right* berisi citra yang diambil dari arah kanan kendaraan. *KITTI\_extracted* berisi citra yang terdiri dari variasi ukuran pengambilan citra. Hasil dari pengumpulan data akan disimpan dalam *pickle file*. Untuk proses klasifikasi, citra dibagi 70% untuk *training* dan 30% untuk *testing*. Pembagian citra dilakukan secara acak.

Setelah proses pengenalan kendaraan selesai, hasil pengenalan kemudian digunakan untuk proses deteksi pada *dataset* yang diambil dari <http://www.youtube.com>. Dari *dataset* yang berupa video, sudah terdapat 6 buah citra yang mewakili setiap kondisi pada folder *test\_images*. Gambar 4.2 menunjukkan citra untuk proses deteksi kendaraan.



**Gambar 4.2** Citra untuk deteksi kendaraan

##### 4.3.2 Implementasi Proses Latih

Pelatihan yang dilakukan merupakan proses klasifikasi dimana membedakan objek mobil dan bukan mobil. Berikut ini adalah uraian dari proses pelatihan yang dilakukan dalam penelitian ini:

1. *Dataset* dimuat dan dikelompokkan menjadi mobil dan bukan mobil.
2. Menentukan semua parameter yang dibutuhkan.
3. Melakukan ekstraksi fitur kepada seluruh *dataset*.
4. Menyatukan dan melakukan normalisasi terhadap fitur.
5. Fitur dari *dataset* kemudian diacak dan dibagi menjadi 70% untuk pelatihan, dan 30% untuk pengujian.
6. Melakukan pelatihan proses klasifikasi SVM menggunakan fitur dari data latih.
7. Menyimpan hasil klasifikasi HOG dalam *pickle file*.

### 4.3.3 Implementasi Proses Uji

Pada bagian ini, akan dilakukan berbagai skenario pengujian dengan beragam parameter dari metode HOG. Tujuan dari penelitian ini adalah untuk menerapkan metode *HOG* pada proses ekstraksi fitur, oleh karena itu perlu diketahui beberapa parameter masukkan yaitu ukuran sel, ukuran blok, dan jumlah *bin* untuk menghasilkan fitur dan akurasi pengenalan karakter yang terbaik. Pengujian ini akan dilakukan dengan data latih sebanyak 6 buah citra yang diambil dari video. 6 buah citra ini didapat dari dataset dan sudah meliputi seluruh keadaan kondisi lokasi penelitian.

### 4.3.4 Implementasi Aplikasi

Bagian ini menjelaskan tampilan *Graphical User Interface* (GUI) yang terdapat dalam aplikasi deteksi mobil dari citra kamera yang diambil dari *dashboard* mobil. Tampilan GUI berguna untuk memudahkan user dalam melakukan proses uji. Tampilan dibuat berbasis *desktop* menggunakan bahasa program python. Berikut adalah tampilan dari aplikasi deteksi mobil:

### 4.3.5 Skenario Pengujian

Pada tahap ini, akan dijelaskan mengenai pengujian yang dilakukan dengan aplikasi yang telah dibuat. Hasil dari pengujian ini adalah untuk mendapatkan hasil klasifikasi untuk deteksi mobil menggunakan metode HOG.

#### 4.3.5.1 Pengujian Kombinasi Parameter

Bagian ini akan menjelaskan pengujian dengan beragam kombinasi parameter metode HOG dalam proses ekstraksi fitur. Fitur dari metode *HOG* akan digunakan pada proses

klasifikasi dan deteksi dengan nilai dari setiap parameter yang akan digunakan untuk kombinasi meliputi:

1. Ukuran sel yang akan digunakan : 2, 4, 8, 16
2. Ukuran blok yang akan digunakan : 4, 8, 16, 32
3. Jumlah *bin* yang akan digunakan : 4, 6, 9, 18

Untuk nilai sigma pada metode *SVM* yang digunakan dalam kombinasi adalah 0.01, 0.1, dan 1. Hasil kemudian akan diukur akurasinya menggunakan *Confusion Matrix*.

#### 4.3.5.2 Pengujian dengan Ukuran Sel 2

Bagian ini akan menjelaskan pengujian terhadap akurasi dengan menggunakan ukuran sel  $2 \times 2$  piksel dan ukuran blok  $2 \times 2$  sel ( $4 \times 4$  piksel). Dalam penelitian ini, jumlah bin yang akan digunakan adalah 4, 6, 9, dan 18 dan untuk nilai sigma pada metode *SVM* yang akan digunakan adalah 0.01, 0.1, dan 1. Berikut adalah hasil pengujian untuk setiap kombinasi parameter tersebut:

**Tabel 4.4** Hasil Pengujian dengan ukuran sel  $2 \times 2$  piksel

Parameter (CellSize, NumBins, Sigma)	CRR	OVR
(2, 4, 0.01)	58.52%	10.34%
(2, 4, 0.1)	26.13%	0%
(2, 4, 1)	18.18%	0%
(2, 6, 0.01)	50%	6.89%
(2, 6, 0.1)	25.56%	0%
(2, 6, 1)	18.18%	0%
(2, 9, 0.01)	48.29%	3.44%
(2, 9, 0.1)	25%	0%
(2, 9, 1)	18.18%	0%
(2, 18, 0.01)	44.88%	3.44%
(2, 18, 0.1)	25%	0%
(2, 18, 1)	18.18%	0%

Berdasarkan tabel di atas, dapat disimpulkan bahwa akurasi pengenalan karakter maksimal yang didapatkan apabila menggunakan ukuran sel  $2 \times 2$  piksel adalah 58.52%. Kombinasi parameter yang digunakan untuk mencapai hasil tersebut adalah ukuran sel  $2 \times 2$  piksel, jumlah *bin* sebanyak 4 sehingga besar setiap *bin* adalah 45 derajat, kemudian nilai *sigma* yang digunakan untuk metode *SVM* adalah 0.01. Dengan citra karakter inputan berukuran  $32 \times 32$  piksel. Maka panjang vektor fitur dari *HOG descriptor* yang dihasilkan

adalah 3600 fitur.

Kolom di bagian kiri menunjukkan kombinasi parameter yang digunakan pada saat pengujian. Kolom CRR merupakan akronim dari *Character Recognition Rate* yang memiliki rumus jumlah karakter yang dikenali dibagi dengan keseluruhan karakter yang terdeteksi. Dari 176 karakter yang terdeteksi, sebanyak 103 di antaranya dapat diklasifikasikan dengan baik. Kolom OVR pada bagian kanan menunjukkan performa keseluruhan yang memiliki rumus jumlah plat nomor yang terdeteksi dan dikenali dengan benar dibagi dengan keseluruhan jumlah plat nomor yang ada. Dari 29 plat nomor yang terdeteksi, hanya 3 plat nomor yang dapat dikenali dengan baik. Dapat disimpulkan bahwa penggunaan ukuran sel  $2 \times 2$  piksel kurang tepat untuk kasus ini.

#### 4.3.5.3 Pengujian dengan Ukuran Sel 4

Pada bagian ini, pengujian akan dilakukan dengan menggunakan ukuran sel berukuran  $4 \times 4$  piksel dan ukuran blok  $2 \times 2$  sel ( $8 \times 8$  piksel). Jumlah bin yang akan digunakan adalah 4, 6, 9, dan 18. Sedangkan untuk nilai sigma pada metode *SVM* yang akan digunakan adalah 0.01, 0.1, dan 1. Berikut adalah hasil pengujian untuk setiap kombinasi parameter tersebut:

**Tabel 4.5** Hasil Pengujian dengan ukuran sel  $4 \times 4$  piksel

Parameter (CellSize, NumBins, Sigma)	CRR	OVR
(4, 4, 0.01)	86.36%	27.58%
(4, 4, 0.1)	47.72%	0%
(4, 4, 1)	26.13%	0%
(4, 6, 0.01)	85.22%	31.03%
(4, 6, 0.1)	40.90%	0%
(4, 6, 1)	25%	0%
(4, 9, 0.01)	89.77%	37.93%
(4, 9, 0.1)	40.90%	0%
(4, 9, 1)	23.86%	0%
(4, 18, 0.01)	84.09%	34.48%
(4, 18, 0.1)	40.90%	0%
(4, 18, 1)	24.43%	0%

Berdasarkan tabel di atas, dapat disimpulkan bahwa akurasi pengenalan karakter maksimal yang didapatkan apabila menggunakan ukuran sel  $4 \times 4$  piksel adalah 89.77%. Kombinasi parameter yang digunakan untuk mencapai hasil tersebut adalah ukuran sel  $4 \times 4$  piksel, jumlah *bin* sebanyak 9 sehingga besar setiap *bin* adalah 20 derajat, kemudian nilai *sigma* yang digunakan untuk metode *SVM* adalah 0.01. Dengan citra karakter inputan berukuran  $32 \times 32$  piksel. Maka panjang vektor fitur dari *HOG descriptor* yang dihasilkan

adalah 1764 fitur. Jika dibandingkan dengan pengujian sebelumnya, jumlah fitur yang lebih sedikit justru mampu mendapatkan akurasi pengenalan karakter yang lebih baik.

Sama seperti bagian sebelumnya, kolom di bagian kiri menunjukkan kombinasi parameter yang digunakan pada saat pengujian. Dari hasil pada kolom CRR, dari 176 karakter yang terdeteksi, sebanyak 158 di antaranya dapat diklasifikasikan dengan baik. Dari hasil pada kolom OVR menunjukkan, dari 29 plat nomor yang terdeteksi, 11 plat nomor dapat dikenali dengan baik, hal ini merupakan peningkatan apabila dibandingkan dengan hasil pengujian sebelumnya, namun masih cukup banyak plat yang tidak dapat dikenali dengan baik. Dari pengujian ini dapat disimpulkan bahwa penggunaan ukuran sel  $4 \times 4$  piksel sudah dapat meningkatkan akurasi pengenalan karakter namun masih belum optimal.

#### 4.3.5.4 Pengujian dengan Ukuran Sel 8

Pada bagian ini, pengujian akan dilakukan dengan menggunakan ukuran sel berukuran  $8 \times 8$  piksel dan ukuran blok  $2 \times 2$  sel ( $16 \times 16$  piksel). Jumlah bin yang akan digunakan adalah 4, 6, 9, dan 18. Sedangkan untuk nilai sigma pada metode *SVM* yang akan digunakan adalah 0.01, 0.1, dan 1. Berikut adalah hasil pengujian untuk setiap kombinasi parameter tersebut:

**Tabel 4.6** Hasil Pengujian dengan ukuran sel  $8 \times 8$  piksel

Parameter (CellSize, NumBins, Sigma)	CRR	OVR
(8, 4, 0.01)	15.90%	0%
(8, 4, 0.1)	93.18%	51.72%
(8, 4, 1)	46.02%	0%
(8, 6, 0.01)	21.02%	0%
(8, 6, 0.1)	93.18%	51.72%
(8, 6, 1)	42.61%	0%
(8, 9, 0.01)	28.40%	0%
(8, 9, 0.1)	93.75%	51.72%
(8, 9, 1)	39.20%	0%
(8, 18, 0.01)	21.59%	0%
(8, 18, 0.1)	94.88%	58.62%
(8, 18, 1)	40.34%	0%

Berdasarkan tabel di atas, dapat disimpulkan bahwa akurasi pengenalan karakter maksimal yang didapatkan apabila menggunakan ukuran sel  $8 \times 8$  piksel adalah 94.88%. Kombinasi parameter yang digunakan untuk mencapai hasil tersebut adalah ukuran sel  $8 \times 8$  piksel, jumlah *bin* sebanyak 18 sehingga besar setiap *bin* adalah 10 derajat, kemudian nilai *sigma* yang digunakan untuk metode *SVM* adalah 0.1. Dengan citra karakter inputan berukuran  $32 \times 32$  piksel. Maka panjang vektor fitur dari *HOG descriptor* yang dihasilkan

adalah 648 fitur. Sama seperti pengujian sebelumnya, jika dibandingkan dengan pengujian sebelumnya, jumlah fitur yang lebih sedikit justru mampu mendapatkan akurasi pengenalan karakter yang lebih baik.

Sama seperti bagian sebelumnya, kolom di bagian kiri menunjukkan kombinasi parameter yang digunakan pada saat pengujian. Dari hasil pada kolom CRR, dari 176 karakter yang terdeteksi, sebanyak 167 di antaranya dapat diklasifikasikan dengan baik. Dari hasil pada kolom OVR menunjukkan, dari 29 plat nomor yang terdeteksi, 17 plat nomor dapat dikenali dengan baik, hal ini merupakan peningkatan apabila dibandingkan dengan hasil pengujian sebelumnya, dengan memperhatikan akurasi pengenalan karakter yang tinggi namun hasil performa keseluruhan yang masih di bawah 70% maka dapat disimpulkan bahwa kemungkinan ada faktor lain yang cukup menghambat dalam proses pengenalan karakter pada keseluruhan plat nomor. Dari pengujian ini dapat disimpulkan bahwa penggunaan ukuran sel  $8 \times 8$  piksel sudah dapat menghasilkan akurasi pengenalan karakter yang baik.

#### 4.3.5.5 Pengujian dengan Ukuran Sel 16

Pada bagian ini, pengujian akan dilakukan dengan menggunakan ukuran sel berukuran  $16 \times 16$  piksel dan ukuran blok  $2 \times 2$  sel ( $32 \times 32$  piksel). Jumlah bin yang akan digunakan adalah 4, 6, 9, dan 18. Sedangkan untuk nilai sigma pada metode *SVM* yang akan digunakan adalah 0.01, 0.1, dan 1. Berikut adalah hasil pengujian untuk setiap kombinasi parameter tersebut:

**Tabel 4.7** Hasil Pengujian dengan ukuran sel  $16 \times 16$  piksel

Parameter (CellSize, NumBins, Sigma)	CRR	OVR
(16, 4, 0.01)	13.06%	0%
(16, 4, 0.1)	18.75%	0%
(16, 4, 1)	84.09%	17.24%
(16, 6, 0.01)	13.06%	0%
(16, 6, 0.1)	22.72%	0%
(16, 6, 1)	84.09%	13.79%
(16, 9, 0.01)	13.06%	0%
(16, 9, 0.1)	28.97%	0%
(16, 9, 1)	86.36%	27.58%
(16, 18, 0.01)	13.06%	0%
(16, 18, 0.1)	23.29%	0%
(16, 18, 1)	85.79%	24.13%

Berdasarkan tabel di atas, dapat disimpulkan bahwa akurasi pengenalan karakter maksimal yang didapatkan apabila menggunakan ukuran sel  $16 \times 16$  piksel adalah 86.36%.

Kombinasi parameter yang digunakan untuk mencapai hasil tersebut adalah ukuran sel  $16 \times 16$  piksel, jumlah *bin* sebanyak 9 sehingga besar setiap *bin* adalah 20 derajat, kemudian nilai *sigma* yang digunakan untuk metode *SVM* adalah 1. Dengan citra karakter inputan berukuran  $32 \times 32$  piksel. Maka panjang vektor fitur dari *HOG descriptor* yang dihasilkan adalah 9 fitur. Berbeda dengan pengujian sebelumnya, kali ini jumlah fitur yang terlalu sedikit justru akan mengurangi akurasi dari proses pengenalan karakter yang sebelumnya sudah mencapai 94.88%, dari keseluruhan pengujian yang sudah dilakukan terhadap jumlah sel, dapat disimpulkan bahwa ukuran sel yang terlampaui besar ataupun terlampaui kecil pada penggunaan metode *HOG* dapat mengurangi kualitas fitur yang dihasilkan sehingga akan berefek terhadap hasil klasifikasi.

Sama seperti bagian sebelumnya, kolom di bagian kiri menunjukkan kombinasi parameter yang digunakan pada saat pengujian. Dari hasil pada kolom CRR, dari 176 karakter yang terdeteksi, sebanyak 152 di antaranya dapat diklasifikasikan dengan baik. Dari hasil pada kolom OVR menunjukkan, dari 29 plat nomor yang terdeteksi, 8 plat nomor dapat dikenali dengan baik, hal ini merupakan dampak dari penurunan akurasi pengenalan karakter. Dari pengujian ini dapat disimpulkan bahwa penggunaan ukuran sel  $16 \times 16$  piksel dapat menghasilkan akurasi pengenalan karakter yang baik namun belum optimal.

## **BAB V**

### **PENUTUP**

**5.1      Kesimpulan**

**5.2      Saran**

## DAFTAR REFERENSI

- [1] S. Bougharriou, F. Hamdaoui, and A. Mtibaa, "*Linear SVM classifier based HOG car detection*", International Conference on Sciences and Techniques of Automatic Control and Computer Engineering (STA), 2017.
- [2] Adhi Prahara and Murinto, "*Car Detection Based on Road Direction on Traffic Surveillance Image*", International Conference on Science in Information Technology (ICSITech), 2016.
- [3] Daniel Neumann, Tobias Langner, Fritz Ulbrich, Dorothee Spitta1and Daniel Goehring, "*Online Vehicle Detection using Haar-like, LBP and HOG Feature basedImage Classifiers with Stereo Vision Preselection*", IEEE Intelligent Vehicles Symposium (IV), 2017.
- [4] Md. Shamim Reza Sajib, and Dr. Saifuddin Md. Tareeq, "*A Feature Based Method for Real Time Vehicle Detection and Classification From Om-Road Videos*", International Conference of Computer an Informtion Technology (ICCIT), 2017.
- [5] A. Shakin Banu and P. Vasuki, "*Video Based Vehicle Detection using Morphological Operation and HOG Feature Extraction*", ARPN Journal of Engineering and Applied Sciences, 2015.
- [6] Gonzalez and Rafael C., "*Digital Image Processing*", Prentice-Hall, Inc., 2002.
- [7] Rinaldi Munir, "*Pengolahan Citra Digital*", Informatika, 2004.
- [8] Otsu, N., "*A Threshold Selection Method from Gray-Level Histogram*", Institute of Electrical and Electronics Engineers (IEEE), 1979
- [9] Navneet Dalal and Bill Triggs, "*Histograms of Oriented Gradients for Human Detection*", Institute of Electrical and Electronics Engineers (IEEE), 2005
- [10] "*Histogram of Oriented Gradients*". [Online]. Available : <https://www.learnopencv.com/histogram-of-oriented-gradients/>.
- [11] Ingo Steinwart and Andreas Christmann , "*Support Vector Machines*", Information Science and Statistics. Springer, New York, 2008.
- [12] K. Markham, "Simple guide to confusion matrix terminology," *Data School*, 08-Jun-2016. [Online]. Available: <https://www.dataschool.io/simple-guide-to-confusion-matrix-terminology/>. [Accessed: 19-Sep-2018].
- [13] G. Adhika dan R.R.W. Ken, "*Penerapan Histogram of Oriented Gradients, Principal Component Analysis, dan AdaBoost untuk Sistem Pengenalan Wajah*", Institut Teknologi Harapan Bangsa (ITHB), 2018