PENERAPAN METODE HOUGH TRANSFORM UNTUK DETEKSI PLAT NOMOR KENDARAAN DAN HISTOGRAM OF ORIENTED GRADIENT UNTUK PENGENALAN KARAKTER PLAT NOMOR KENDARAAN

TUGAS AKHIR

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

Disusun Oleh:
Jeffry Saputra
1115040

INSTITUT TEKNOLOGI HARAPAN BANGSA

Veritas vos liberabit

PROGRAM STUDI INFORMATIKA
INSTITUT TEKNOLOGI HARAPAN BANGSA
BANDUNG
2019

BAB I

PENDAHULUAN

1.1 Latar Belakang

Computer Vision atau disebut juga Penglihatan Komputer adalah sebuah cabang ilmu komputer yang mempelajari bagaimana komputer dapat memiliki kemampuan untuk dapat menginterpretasikan suatu kondisi melalui sebuah citra dan dapat bekerja selayaknya seperti penglihatan manusia. Terdapat beberapa tahapan dalam *computer vision* yang digunakan untuk persepsi visual, seperti akuisisi citra, pengolahan citra, formasi citra, ekstraksi dan pencocokan fitur, segmentasi, deteksi dan pengenalan objek, dan lain sebagainya. Deteksi objek adalah metode untuk mendeteksi suatu objek dan digunakan untuk mencari objek-objek dari suatu citra. Dalam aplikasi sistem kecerdasan untuk transportasi, objek dapat berupa mobil, bagian dari mobil (logo, plat nomor kendaraan), ataupun rambu-rambu lalu lintas.

Sistem kecerdasan untuk transportasi merupakan bidang yang saat ini sedang berkembang dengan pesat dalam ranah *computer vision* [1]. Penerapannya pun semakin nyata dalam kehidupan manusia sehari-hari. Sistem navigasi satelit, sistem pengenalan rambu lalu lintas, sistem parkir otomatis, pengenalan plat kendaraan, dan keamanan kendaraan merupakan contoh dari aplikasi sistem kecerdasan untuk transportasi. Pengenalan plat nomor kendaraan memegang beberapa peranan penting dalam bidang transportasi, diantaranya untuk sistem pembayaran elektronik, dan penegakan hukum [2].

Walaupun sistem pengenalan plat nomor kendaraan sudah memiliki sejarah penelitian yang panjang, hal ini tetap saja memiliki tantangan. Hal ini disebabkan banyak faktor yang mempengaruhi hasil akhir dari pengenalan plat nomor, contohnya adalah kondisi pencahayaan yang tidak merata, kondisi tulisan karakter pada plat nomor yang kurang jelas, dan lain sebagainya [2].

Secara umum, sistem pengenalan plat nomor kendaraan dibagi kedalam tiga bagian utama: deteksi area plat nomor kendaraan, segmentasi karakter, dan pengenalan karakter. Gou et al. menerapkan ketiga hal tersebut dengan menggunakan metode *Extremal Region* untuk mendeteksi lokasi plat nomor kendaraan sekaligus mendapatkan area dari karakter plat nomor kendaraan tersebut kemudian

melakukan pengenalan karakter menggunakan *Restricted Boltzmann Machines* [3].

Penelitian lain menggunakan *Maximally Stable Extremal Region* untuk mendeteksi area karakter dari plat nomor kendaraan kemudian dilanjutkan dengan metode *Histogram of Oriented Gradient* untuk mendapatkan fitur dari masing-masing karakter dan menggunakan metode *Extreme Learning Machine* untuk melakukan proses pengenalan karakter [2].

Penelitian lain menggunakan metode morfologi citra untuk deteksi plat kendaraan dan menggunakan *K-Nearest Neighbors* untuk melakukan klasifikasi terhadap karakter dan *Support Vector Machine* untuk melakukan klasifikasi terhadap karakter yang memiliki kemiripan (e.g: pasangan B dengan 8, 5 dengan S, 4 dengan A, dsb) [1].

Penelitian lain menggunakan metode *Hough Transform* untuk mendeteksi lokasi plat kendaraan kemudian dilanjutkan dengan metode *Template Matching* untuk mengenali karakter dari plat nomor tersebut [4].

Penelitian ini menggunakan metode *Hough Transform* untuk mendeteksi plat nomor kendaraan, kemudian karakter-karakter pada plat kendaraan akan disegmentasi dengan menghitung grafik horizontal pita, kemudian dilanjutkan dengan menggunakan metode *Histogram of Oriented Gradient* untuk mengambil fitur dari karakter-karakter dari citra hasil segmentasi dan terakhir akan diklasifikasikan dengan menggunakan metode *Support Vector Machine*.

Pada tahapan pengujian akan dilakukan dua macam pengujian akurasi, yaitu akurasi deteksi plat kendaraan dan akurasi pengenalan karakter plat nomor kendaraan. Perhitungan akurasi ini masing-masing akan menggunakan *confusion matrix*.

1.2 Rumusan Masalah

Berdasarkan latar belakang di atas penulis merumuskan masalah sebagai berikut:

1. Berapa akurasi pendeteksian plat nomor kendaraan dengan menggunakan metode *Hough Transform* ?
2. Berapa akurasi pengenalan plat nomor kendaraan jika menggunakan metode *Support Vector Machine* yang ekstraksi fiturnya menggunakan metode *Histogram of Oriented Gradient* ?

1.3 Tujuan Penelitian

Berdasarkan rumusan masalah di atas, tujuan penelitian Tugas Akhir ini adalah sebagai berikut:

1. Mengimplementasi metode untuk pendeteksian plat nomor kendaraan dan implementasi metode untuk ekstraksi fitur pada pengenalan karakter plat nomor kendaraan dalam bentuk program.

1.4 Batasan Masalah

Dalam penelitian ini, peneliti akan membatasi masalah yang akan diteliti antara lain:

1. Plat kendaraan yang akan dideteksi adalah plat nomor kendaraan Indonesia.
2. Plat kendaraan Indonesia yang akan dideteksi adalah plat nomor kendaraan pribadi (plat hitam dengan tulisan putih), plat nomor kendaraan umum (plat kuning dengan tulisan hitam), dan plat nomor kendaraan dinas negara (plat merah dengan tulisan putih).

1.5 Kontribusi Penelitian

Kontribusi yang diberikan dari penelitian ini adalah:

1. Membuat penerapan metode *Hough Transform* untuk pendeteksian plat nomor kendaraan.
2. Membuat penerapan metode *Histogram of Oriented Gradient* untuk proses ekstraksi fitur pada proses pengenalan karakter pada plat nomor kendaraan.

1.6 Metode Penelitian

Metode penelitian yang dilakukan dalam penelitian ini adalah sebagai berikut:

1. Studi Literatur

   Penulisan ini dimulai dengan studi kepustakaan yaitu mengumpulkan bahan-bahan referensi baik dari buku, artikel, *paper*, jurnal, makalah mengenai sistem pengenalan plat nomor kendaraan.
2. Data sampling

   Data sampling yang akan digunakan berupa citra kendaraan yang berasal dari Universitas Telkom yang bernama *Tel-U Vehicle License Plate Data-set V1.0*. Dataset ini merupakan dataset yang disusun oleh akademisi Universitas Telkom untuk keperluan penelitian mengenai plat nomor kendaraan.
3. Analisis Masalah

   Pada tahap ini dilakukan analisis permasalahan yang ada, batasan yang dimiliki

dan kebutuhan yang diperlukan.

4. Perancangan dan Implementasi Algoritme

   Pada tahap ini dilakukan pendefinisian beberapa aturan dalam teknik *preprocessing* citra, serta perancangan pada algoritme yang akan dipakai untuk menyelesaikan masalah berdasarkan metode yang telah dipilih.

5. Pengujian

   Pada tahap ini dilakukan pengujian terhadap aplikasi yang telah dibangun.

6. Dokumentasi

   Pada tahap ini dilakukan pendokumentasian hasil analisis dan implementasi secara tertulis dalam bentuk laporan skripsi.

1.7 Sistematika Pembahasan

Pada penelitian ini peneliti menyusun berdasarkan sistematika penulisan sebagai berikut:

BAB I PENDAHULUAN

Pendahuluan yang berisi latar belakang, rumusan masalah, tujuan penelitian, batasan masalah, kontribusi penelitian, serta metode penelitian.

BAB II LANDASAN TEORI

Landasan Teori yang berisi penjelasan dasar teori yang mendukung penelitian ini.

BAB III ANALISIS DAN PERANCANGAN

Analisis dan Perancangan yang berisi analisis berupa algoritme yang digunakan.

BAB IV IMPLEMENTASI DAN PENGUJIAN

Implementasi dan Pengujian yang berisi implementasi pengujian dengan berbagai data testing beserta hasilnya.

BAB V KESIMPULAN DAN SARAN

Penutup yang berisi kesimpulan dari penelitian dan saran untuk penelitian lebih lanjut di masa mendatang.

BAB II

LANDASAN TEORI

Bab ini menjelaskan teori-teori yang berkaitan mengenai teori penunjang dan jurnal terkait yang digunakan dalam proses penelitian tugas akhir ini.

2.1 Tinjauan Pustaka

Penelitian ini menggunakan beberapa teori terkait yang diperlukan dalam pengerjaan yang dilakukan. Penjelasan mengenai teori-teori tersebut akan dijelaskan sebagai berikut.

2.1.1 Citra Digital

Citra digital merupakan sebuah fungsi dua dimensi *f(x,y)*, di mana *x* dan *y* adalah koordinat, dan nilai *f* menyatakan intensitas atau tingkat keabuan yang dimiliki citra pada titik atau *pixel* (*picture element*) tersebut. Nilai *f* merupakan nilai berhingga dan bersifat diskrit [5].

Jenis citra digital bergantung pada jenis perangkat keras yang digunakan dan dapat dikelompokkan ke dalam beberapa jenis model warna, yang paling umum digunakan adalah model RGB. Citra model RGB merupakan citra yang menggunakan 3 kombinasi warna, yaitu merah, hijau, dan biru. Pada citra RGB 24-bit, setiap warna mempunyai nilai *f* antara 0 hingga 255 sehingga perpaduan dari ketiga warna tersebut akan menghasilkan 256^3 jenis warna [5].

2.1.2 Pengolahan Citra

Pengolahan citra dapat didefinisikan sebagai suatu bidang yang menggunakan citra sebagai masukan, lalu masukan tersebut diolah sehingga menghasilkan citra kembali. Berdasarkan definisi tersebut, perhitungan rata-rata dari suatu citra yang menghasilkan sebuah angka tidak termasuk pengolahan citra [5].

Terdapat satu paradigma yang mengategorikan 3 jenis proses komputasi dalam pengolahan citra, yaitu tingkat rendah, tingkat sedang, dan tingkat tinggi. Proses tingkat rendah mencakup operasi yang sangat sederhana seperti *image preprocessing* untuk mengurangi *noise*, meningkatkan kontras, dan mempertajam citra. Proses tingkat sedang meliputi segmentasi untuk membagi daerah citra menjadi *region* atau objek. Sedangkan proses tingkat tinggi memampukan komputer untuk mengerti seperti pengenalan objek dan analisis citra [5].

2.1.3 Pengabuan Citra

Citra RGB yaitu citra berwarna memiliki ukuran yang lebih besar dibandingkan dengan citra *grayscale*. Untuk mempercepat proses komputasi pada citra, maka citra RGB perlu diubah menjadi citra *grayscale* dengan skala keabuan 256. Persamaan pengabuan citra dapat dilihat pada persamaan 2 . 1.

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

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 [10]. Persamaan 2 . 1 merupakan rekomendasi dari *International Telecommunication Union Radiocommunication Sector*.

2.1.4 Deteksi Tepi

Tepian memiliki arti yaitu terjadinya perubahan intensitas secara signifikan pada sebuah citra. Deteksi tepi yang digunakan pada penelitian ini adalah dengan menggunakan operator *Canny* untuk mendapatkan tepian citra sebesar 1 piksel. Proses deteksi tepi Canny memiliki tahapan sebagai berikut:

1. Penghalusan
   Pada tahapan pertama citra dihaluskan dengan *Gaussian filter* untuk mengurangi derau yang dapat menghasilkan detail yang mengganggu.

2. Menghitung Gradien
   Untuk menghitung gradien yang dapat menghasilkan tepian yang masih tebal dengan beberapa operator, diantaranya adalah Sobel, Prewitt, atau Robert.

3. *Non-maxima Supression* Tahapan ini digunakan untuk menipiskan tepian tebal yang diperoleh dari operasi sebelumnya dengan cara mencari nilai maksimum di tepian.

4. *Double Thresholding*
   Dari hasil *non-maxima supression* bisa ditemukan tepian yang belum sempurna, sehingga perlu dilakukan *thresholding* untuk menghilangkan derau yang tidak diinginkan. Caranya adalah dengan menetapkan 2 nilai *threshold* yaitu *high threshold* dan *low threshold* untuk menentukan apakah

piksel tersebut akan masuk dalam *threshold* untuk dijadikan tepian. Piksel yang nilainya berada di atas *high threshold* akan menjadi tepian kuat, sebaliknya jika di bawah *low threshold* akan dijadikan sebagai *background*.

5. *Edge Tracking* Tahapan terakhir yaitu *Edge Tracking* atau *Edge Linking* digunakan untuk menghubungkan tepian kuat dan tepian lemah yang nilai pikselnya berada diantara *high threshold* dan *low threshold*. Ketika tepian lemah yang tidak terhubung dengan tepian kuat maka piksel tersebut akan dianggap sebagai *background*. Hasil akhir dari deteksi tepi Canny adalah tepian halus yang memiliki lebar sebesar 1 piksel.

2.1.5 *Hough Transform*

Hough Transform adalah sebuah teknik untuk mengidentifikasi bentuk spesifik dalam sebuah citra. *Hough Transform* mengkonversikan semua titik dalam sebuah kurva ke dalam sebuah lokasi tunggal dalam ruang parametrik (ruang akumulator) lain dengan transformasi koordinat. Metode ini bertujuan untuk memetakan fitur global ke fitur lokal. Konsep ini juga dapat diterapkan untuk mendeteksi garis lurus, lingkaran, elips atau bentuk geometrik lainnya. *Hough Transform* yang digunakan adalah untuk ekstraksi garis lurus. Berikut adalah rumus yang digunakan untuk mencari jarak antara titik *origin* dengan garis yang terbentuk [6]:

$$\rho = x\cos(\theta) + y\sin(\theta) \tag{2 . 2}$$

Dimana:
ρ = jarak antara titik *origin* dengan garis
x = koordinat titik x
y = koordinat titik y
θ = sudut derajat; $0^\circ \leq \theta \leq 180^\circ$

Metode *Hough Transform* menerapkan skema *voting*. Sebuah *array* akumulator diperlukan untuk menyimpan hasil *voting*. Rentang nilai θ (*theta*) yang digunakan adalah antara nilai 0 hingga 180 derajat. Sedangkan rentang nilai ρ (*rho*) yang digunakan dalam akumulator adalah [7]:

$$-D \leq \rho \leq D \tag{2 . 3}$$

Dimana:
D = jarak diagonal dari gambar

Karena citra yang digunakan berbentuk persegi panjang maka jarak diagonal memenuhi persamaan berikut:

$$D = \sqrt{N^2 + M^2} \quad (2.4)$$

Dimana:
D = jarak diagonal dari gambar
N = ukuran *width* dari gambar
M = ukuran *height* dari gambar

Sehingga rentang nilai *rho* yang digunakan dalam penelitian ini adalah:

$$-\sqrt{N^2 + M^2} \leq \rho \leq \sqrt{N^2 + M^2} \quad (2.5)$$

Dimana:
N = ukuran *width* dari gambar
M = ukuran *height* dari gambar

Setelah proses perhitungan *voting* dalam *accumulator space* selesai, yang dilakukan selanjutnya adalah memilih *peak* terbaik yang terdapat dalam *accumulator space*. Nilai *peak* yang tinggi memberikan indikasi yang baik dari garis [8]. Berdasarkan R. Varun, *et al*. (2015), diketahui *local maxima* dari *accumulator space* dipertimbangkan sebagai *peak* yang menonjol. Jumlah *peak* merupakan hal yang krusial. Jumlah *peak* yang terlalu sedikit atau terlalu banyak dapat mempengaruhi kinerja sistem.

Algoritma pencarian *peak* menerapkan nilai *threshold* dan pencarian lokal yang berdasarkan dari ukuran *neighbourhood* (NS). Nilai *threshold* untuk membatasi nilai *voting* untuk mempertimbangkan *peak* yang menonjol. Nilai NS bernilai 1 atau lebih. Algoritma pencarian *peak* tersebut dapat menghindari hasil ganda untuk sebuah garis [8].

Serangkaian *peaks* yang diekstrak oleh metode *Hough Transform* akan menghasilkan beragam nilai *theta*. Intensitas kemunculan dari setiap nilai *theta* akan dihitung dan dijadikan fitur yang mewakili sebuah citra plat nomor.

Dari penjelasan di atas maka *output* dari proses ekstraksi fitur dengan metode *Hough Transform* adalah intensitas kemunculan dari setiap nilai *theta* yang didapat dari keseluruhan *peak* yang terpilih. Karena rentang nilai *theta* adalah 0 - 180 derajat maka ukuran fitur yang diekstrak adalah 181.

2.1.6 Fitur pada Citra

Dalam sistem pengenalan objek, fitur merupakan hal yang penting. Fitur merupakan atribut yang menonjol atau karakteristik yang dapat membedakan antara satu objek dengan objek lainnya. Fitur pada sebuah citra dapat digunakan untuk proses segmentasi dan klasifikasi. Sebuah objek dapat dibedakan berdasarkan fitur internal dan fitur eksternal. Fitur internal didapatkan berdasarkan komposisi piksel yang membentuk suatu wilayah (*region*), sedangkan fitur eksternal membahas mengenai batas wilayah (*region boundary*) dari sebuah objek. Contoh fitur internal adalah fitur tekstur, fitur dasar geometri, momen, histogram, dan *Euler Number*. Fitur eksternal adalah *Chain codes*, *signatures*, dan *Fourier descriptors* untuk menggambarkan bentuk dari objek (*shape descriptor*) [5].

2.1.7 *Histogram of Oriented Gradient*

Histogram of Oriented Gradients merupakan salah satu teknik pengambilan fitur yang bertujuan untuk mengambil informasi penting dari sebuah citra. Cara kerja metode ini yaitu dengan mengevaluasi histogram lokal yang sudah ternormalisasi secara baik dari distribusi gradien citra dalam *grid* yang padat. Teknik mengekstrak fitur untuk metode ini yaitu dari distribusi lokal dari intensitas gradient tiap piksel yang terdapat pada sebuah objek citra. Dalam metode *Histogram of Oriented Gradient*, ukuran sel berupa kumpulan atau gabungan piksel dan blok berupa kumpulan atau gabungan sel beserta jumlah *orientation bin* yang merupakan tempat unutk menampung hasil arah dan besar gradien akan mempengaruhi hasil keluaran fitur vektor yang dihasilkan dan juga akurasi yang didapat. Pertama untuk setiap piksel dari citra akan dihitung gradiennya dari sumbu x dan y dengan menggunakan persamaan :

$$G_x(x,y) = I(x+1,y) - I(x-1,y) \quad (2.6)$$

$$G_y(x,y) = I(x,y+1) - I(x,y-1) \quad (2.7)$$

Dimana:
$G_x(x,y)$ = nilai gradient untuk sumbu x
$G_y(x,y)$ = nilai gradient untuk sumbu y
$I(x,y)$ = nilai piksel citra dari baris x dan kolom y

Setelah didapat nilai gradient dari sumbu x dan y untuk setiap pikselnya, proses selanjutnya adalah menghitung besar nilai dan arah gradiennya dengan menggunakan rumus:

$$M(x,y) = \sqrt{G_x(x,y)^2 + G_y(x,y)^2} \quad (2.8)$$

$$\theta(x,y) = arctan\frac{G_y(x,y)}{G_x(x,y)} \quad (2.9)$$

Dimana:
$M(x,y)$ = besar nilai gradient dari sumbu x dan y
$\theta(x,y)$ = arah nilai gradient dari sumbu x dan y

Kemudian, setiap piksel citra akan dibagi ke dalam beberapa sel yang dari setiap sel, akan dihitung persebaran *Histogram of Oriented Gradient*-nya melalui proses *voting*. Proses *voting* dalam *Histogram of Oriented Gradient* pertama akan menentukan nilai-nilai dari *bin* dengan membagi total jumlah sudut gradien ke dalam jumlah *orientation bin*. Kemudian untuk setiap arah sudut gradien dari setiap piksel dalam sel akan dimasukkan ke dalam rentang *orientation bin* yang sudah ditentukan pada pertama kali, kemudian membagi besar nilai gradiennya dengan *orientation bin* yang terkait.

Setelah *Histogram of Oriented Gradient* sudah dibuat untuk setiap sel, proses selanjutnya adalah melakukan normalisasi terhadap hasil *vote* pada setiap *bin* dalam sel. Normalisasi akan dilakukan dalam 1 blok, dengan ukuran blok

merupakan m $\times$ n sel. Terdapat 4 macam metode untuk normalisasi, yaitu: *L2-Norm*, *L2-Hys*, *L1-sqrt*, dan *L1-Norm*. Persamaannya adalah sebagai berikut:

$$V_i = \frac{V_i}{\sum_{i=1}^{N} V_i} \quad (2.10)$$

Dimana:
V_i = bobot vektor hasil *L1-Norm* yang merepresentasikan nilai setiap *bin*
i = nilai *counter* dari 1 sampai N
N = jumlah total nilai total *bin* yang digunakan dalam proses normalisasi

$$V_i = \sqrt{\frac{V_i}{\sum_{i=1}^{N} V_i}} \quad (2.11)$$

Dimana:
V_i = bobot vektor hasil *L1-sqrt* yang merepresentasikan nilai setiap *bin*
i = nilai *counter* dari 1 sampai N
N = jumlah total nilai total *bin* yang digunakan dalam proses normalisasi

$$V_i = \frac{V_i}{\sqrt{\sum_{i=1}^{N} V_i^2}} \quad (2.12)$$

Dimana:
V_i = bobot vektor hasil *L2-Norm* yang merepresentasikan nilai setiap *bin*
i = nilai *counter* dari 1 sampai N
N = jumlah total nilai total *bin* yang digunakan dalam proses normalisasi

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

Adapun proses normalisasi blok akan dilakukan dalam *sliding window* yang akan bergerak melakukan proses dengan pergeseran sebesar 1$\times$ ukuran sel secara vertikal dan horizontal. Proses ini kemudian akan bersifat *overlapping* untuk beberapa sel yang dinormalisasi sehingga menimbulkan informasi yang redundan, namun akurasi yang dihasilkan justru semakin meningkat karenanya. Terakhir,

hasil dari normalisasi tiap blok akan digabungkan menjadi 1 fitur vektor besar.

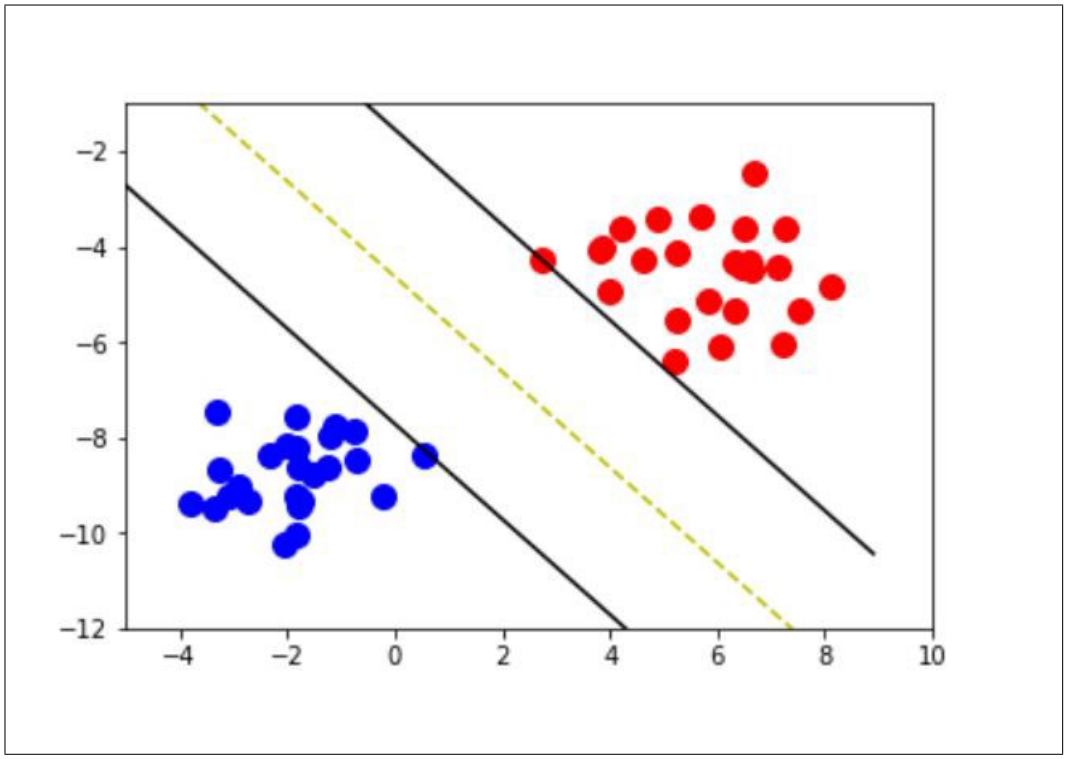
2.1.8 *Support Vector Machine*

Support Vector Machine merupakan salah satu algoritme *supervised learning* untuk melakukan klasifikasi serta regresi dengan menggunakan teori vektor. SVM dapat memetakan vektor input ke dalam sebuah ruang beukuran n-dimensional (n adalah jumlah fitur). Konsep dasar dari SVM adalah menemukan sebuah *separating hyperplane* (bidang) yang dapat memisahkan dua kelas dengan margin maksimal.

Garis putus-putus yang berada paling dekat dengan masing-masing kelas merupakan *hyperplane* paralel untuk memisahkan kedua kelas (lihat gambar 2.1). Asumsinya adalah semakin besar jarak atau margin antara 2 *hyperplane* pendukung ini maka semakin baik hasil klasifikasinya. *Hyperplane* yang optimal harus memenuhi persamaan 2 . 13 berikut:

$$w^T \cdot x + b = 0 \tag{2 . 13}$$

Dimana w^T adalah vektor berat dan x adalah vektor input dan b merupakan nilai bias. Tanda "." menggambarkan perkalian dot vektor.



Gambar 2.1 Contoh *Hyperplane* pada SVM

SVM pada mulanya digunakan untuk menangani klasifikasi yang terdiri dari 2 kelas saja. Namun seriring dengan perkembangan zaman masalah yang dihadapi semakin kompleks sehingga membutuhkan teknik untuk melakukan proses klasifikasi lebih dari 2 kelas. Untuk melakukan klasifikasi lebih dari 2 kelas, terdapat 2 pendekatan yang bisa digunakan yaitu *One-Versus-One* dan *One-Versus-Rest*. Pada pendekatan *One-Versus-One*, akan dibuat sebanyak $k(k-1)/2$ pasangan kelas untuk pengujian untuk klasifikasi dengan kelas sebanyak k. Untuk menentukan kelas mana yang menjadi klasifikasi untuk suatu kumpulan data caranya adalah sistem *voting*. Kelas dengan jumlah *voting* terbanyak akan menjadi *classifier* untuk data tersebut. Pada pendekatan *One-Versus-Rest* akan dibuat sebanyak k pasangan kelas untuk klasifikasi dengan kelas sebanyak k. Setiap kelas yang diuji akan dibandingkan dengan sisa kelas yang ada. Misal terdapat 3 kelas A,B, dan C, maka kelas A akan dibandingkan dengan kelas B dan C, kelas B dibandingkan dengan kelas A dan C, kelas C dibandingkan dengan kelas A dan B. Kekurangan dari pendekatan ini adalah jumlah *training set* yang tidak seimbang [9].

Untuk proses klasifikasi *non-linear* dapat dicari dengan persamaan 2.13 berikut:

$$f(x) = sign(\sum_{i=1}^{l} \alpha_i y_i K(x, x_i) + b) \quad (2.14)$$

Dimana:

l	=	Banyaknya kelas citra
α_i	=	Nilai alpha ke i
y_i	=	Nilai kelas citra ke i
$K(x, x_i)$	=	Fungsi kernel
b	=	Nilai bias

Nilai α dan b dapat dicari dengan persamaan linear yang membentuk *hyperplane* SVM. Persamaan 2 . 15 sampai 2 . 17 berikut merupakan persamaan *hyperplane* SVM:

$$\sum_{i=1}^{l} \alpha_i y_i K(x, x_i) + b) = 0 \quad (2.15)$$

$$\sum_{i=1}^{l} \alpha_i y_i K(x, x_i) + b) = 1 \quad (2.16)$$

$$\sum_{i=1}^{l} \alpha_i y_i K(x, x_i) + b) = -1 \quad (2.17)$$

Dimana:

l	=	banyaknya kelas citra
α_i	=	nilai alpha ke i
y_i	=	nilai kelas citra ke i
$K(x, x_i)$	=	fungsi kernel
b	=	nilai bias

Seringkali kasus yang ada dalam dunia nyata tidak selalu bisa dipisahkan secara linier (*linearly separable*) seperti pada contoh gambar di atas. Misalnya suatu kumpulan data memiliki fitur yang memiliki n-dimensi. Linear SVM tidak bisa diterapkan untuk kasus tersebut, sehingga diperlukan teknik agar membuat *hyperplane* yang bisa memisahkan antara 2 kelas dalam ruang multidimensi. Cara yang umum digunakan untuk menyelesaikan masalah tersebut adalah dengan menggunakan kernel. Kernel yang umum digunakan pada SVM yaitu *Radial Basis Function* seperti pada persamaan 2 . 18 berikut:

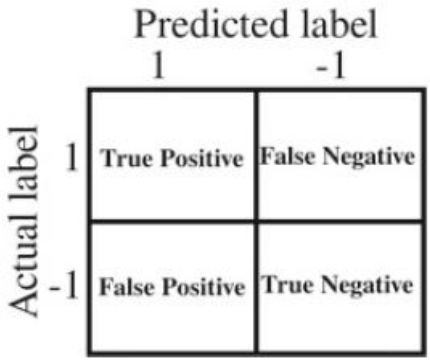
$$RBF = K(x_i, x_j) = \exp(-\frac{\|x_i - x_j\|}{2\sigma^2}) \quad (2.18)$$

Dimana:

K	=	nilai fungsi kernel RBF
x_i	=	vektor input 1
x_j	=	vektor input 2
σ	=	konstanta sigma

2.1.9 *Confusion Matrix*

Confusion Matrix merupakan metode pengukuran untuk mengevaluasi hasil klasifikasi. Dengan melakukan klasifikasi sebanyak C kelas, dihasilkan *confusion matrix* M berukuran $C \times C$, di mana elemen M_{ij} dalam matriks menunjukkan jumlah sampel yang salah diklasifikasikan, sementara M_{ii} adalah jumlah sampel yang hasil klasifikasinya adalah benar. *Confusion matrix* pada gambar 2.2 digunakan pada kasus klasifikasi dua buah kelas sehingga membentuk matriks berukuran 2×2 [**?**].



Gambar 2.2 *Confusion Matrix untuk Dua Kelas* [**?**]

Elemen M_{11} pada matriks menunjukkan jumlah sampel yang pada kenyataannya adalah kelas 1 dan diklasifikasikan sebagai kelas 1, sehingga disebut sampel *true-positive* (TP). Elemen M_{12} menunjukkan jumlah sampel yang pada kenyataannya adalah kelas 1 tetapi diklasifikasikan sebagai kelas -1, sehingga disebut sampel *false-negative* (FN). Elemen M_{21} menunjukkan jumlah sample yang pada kenyataannya adalah kelas -1 tetapi diklasifikasikan sebagai kelas 1, sehingga disebut sampel *false-positive* (FP). Dan elemen M_{22} menunjukkan jumlah sampel yang kenyataannya adalah kelas -1 dan diklasifikasikan sebagai kelas -1, sehingga disebut *true-negative* (TN). Maka untuk menghitung akurasi dapat

digunakan persamaan 2 . 19. Hasil akurasi yang semakin baik akan mendekati nilai 1, sebaliknya akurasi yang buruk mendekati nilai 0.

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \quad (2 . 19)$$

Lalu perhitungan *precision* yang merupakan perbandingan dari hasil positif dapat dihitung dengan persamaan 2 . 20.

$$Precision = \frac{TP}{TP+FP} \quad (2 . 20)$$

Dan perhitungan *recall* atau disebut juga sebagai sensitivitas dapat dihitung dengan persamaan 2 . 21.

$$Recall = \frac{TP}{TP+FN} \quad (2 . 21)$$

2.2 Tinjauan Studi

Pada bagian ini akan dijelaskan mengenai perbandingan dari berbagai penelitian terkait metode deteksi dan pengenalan plat nomor mobil.

2.2.1 *State of the Art*

Terdapat beberapa metode lain yang memiliki ruang lingkup yang mirip dengan penelitian ini khususnya mengenai deteksi dan pengenalan plat nomor mobil. Tabel 2.1 *State of the Art* akan menjelaskan perbedaan-perbedaan metode yang telah dipelajari oleh penulis dari jurnal.

Tabel 2.1 *State of the Art*

Jurnal	**Rumusan Masalah**	**Metode**
Gou, C., Wang, K., Yao, Y., Li, Z. (2016). Vehicle license plate recognition based on extremal regions and restricted Boltzmann machines. *IEEE Transactions on Intelligent Transportation Systems, 17*(4), 1096-1107.	Apakah dengan menerapkan pendeteksian plat nomor dengan *Extremal Region* dan pengenalan karakter plat nomor menggunakan *Hybrid Discriminative Restricted Boltzmann Machine* dapat meningkatkan akurasi pendeteksian dan pengenalan dalam berbagai kondisi cuaca dan *background* yang kompleks?	1. *Extremal Region* 2. *AdaBoost* 3. *Histogram of Oriented Gradient* 4. *Hybrid Discriminative Restricted Boltzmann Machine*
Gou, C., Wang, K., Yu, Z., Xie, H. (2014, October). License plate recognition using MSER and HOG based on ELM. In *Proceedings of 2014 IEEE International Conference on Service Operations and Logistics, and Informatics* (pp. 217-221). IEEE.	Apakah dengan menerapkan *Maximally Stable Extremal Region* untuk pendeteksian plat nomor dan *Extreme Learning Machine* untuk pengenalan karakter plat nomor dapat meningkatkan performa dan akurasi sistem?	1. *Maximally Stable Extremal Region* 2. *Histogram of Oriented Gradient* 3. *Extreme Learning Machine*

Tabel 2.1 *State of the Art* (Lanjutan)

Jurnal	**Rumusan Masalah**	**Metode**
Tabrizi, S. S., Cavus, N. (2016). A hybrid KNN-SVM model for Iranian license plate recognition. *Procedia Computer Science, 102*, pp. 588-594.	Apakah dengan menggabungkan metode klasifikasi *K-Nearest Neighbours* dan *Support Vector Machine* akan menghasilkan akurasi yang lebih baik dan mengurangi *cost* pada proses pengenalan karakter plat nomor kendaraan?	1. *Structural Feature* 2. *Horizontal and Vertical Crossing Count Histogram* 3. *Zoning Feature Extraction* 4. *K-Nearest Neigbhours* 5. *Support Vector Machine*
Rasheed, S., Naeem, A., Ishaq, O. (2012, October). Automated number plate recognition using hough lines and template matching. In *Proceedings of the World Congress on Engineering and Computer Science* (Vol. 1, pp. 24-26).	Apakah dengan menggunakan metode *Hough Transform* untuk mendeteksi plat kendaraan dan *Template Matching* untuk pengenalan karakter dapat menghasilkan akurasi yang baik untuk sistem pengenalan plat nomor kendaraan?	1. *Canny Detector* 2. *Hough Transform* 3. *Morphological Process* 4. *Template Matching*

2.2.2 Pembahasan Penelitian Terkait

Terdapat beberapa metode yang dapat khususnya untuk mendeteksi plat nomor dan mengenali karakter pada plat nomor. Pada referensi pertama [3] menggunakan metode *Extremal Region* sebagai proses untuk mendapatkan daerah-daerah karakter dari suatu plat nomor, kemudian *Extremal Region* yang didapat diseleksi dengan menggunakan *AdaBoost* sehingga bisa didapatkan daerah karakter plat nomor yang sesuai dengan kriteria yang diinginkan, dari daerah-daerah karakter yang didapatkan barulah kandidat plat nomor yang benar bisa didapatkan. Proses selanjutnya adalah pengambilan fitur karakter dengan menggunakan metode *Histogram of Oriented Gradient* sehingga didapatkan fitur vektor dari setiap

karakter pada plat nomor, yang nantinya akan menjadi masukkan bagi metode klasifikasi *Hybrid Discriminative Restricted Boltzmann Machine*.

Pada referensi kedua [2] menggunakan metode *Maximally Stable Extremal Region* untuk memilih kandidat daerah karakter yang nantinya akan menentukan lokasi dari plat nomor berdasarkan letak geometris dari kandidat-kandidat karakter tersebut. Setelah lokasi plat nomor didapatkan, *HOG Descriptor* dari setiap karakter diambil dengan menggunakan metode *Histogram of Oriented Gradient* dan setiap karakternya akan dikenali menggunakan metode *neural network* bernama *Extreme Learning Machine*.

Pada referensi ketiga [1] digabungkan metode klasifikasi *K-Nearest Neighbours* dengan metode klasifikasi *Support Vector Machine*. *K-Nearest Neighbours* digunakan karena sifatnya yang mudah dipelajari, bersifat tangguh terhadap data yang memiliki derau dan efektif jika jumlah yang dimiliki berjumlah banyak. Sedangkan metode *Support Vector Machine* digunakan untuk karakter-karakter yang memiliki kemiripan karakteristik, sehingga akurasi dari pengenalan karakter dapat meningkat.

Pada referensi keempat [4] menggunakan *Hough Transform* untuk mendeteksi lokasi plat kendaraan dan menghasilkan akurasi yang baik, yaitu sekitar 94% untuk plat nomor yang terdeteksi dan dengan menggunakan metode *Template Matching* menghasilkan akurasi pengenalan karakter plat nomor kendaraan sebesar 90%.

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

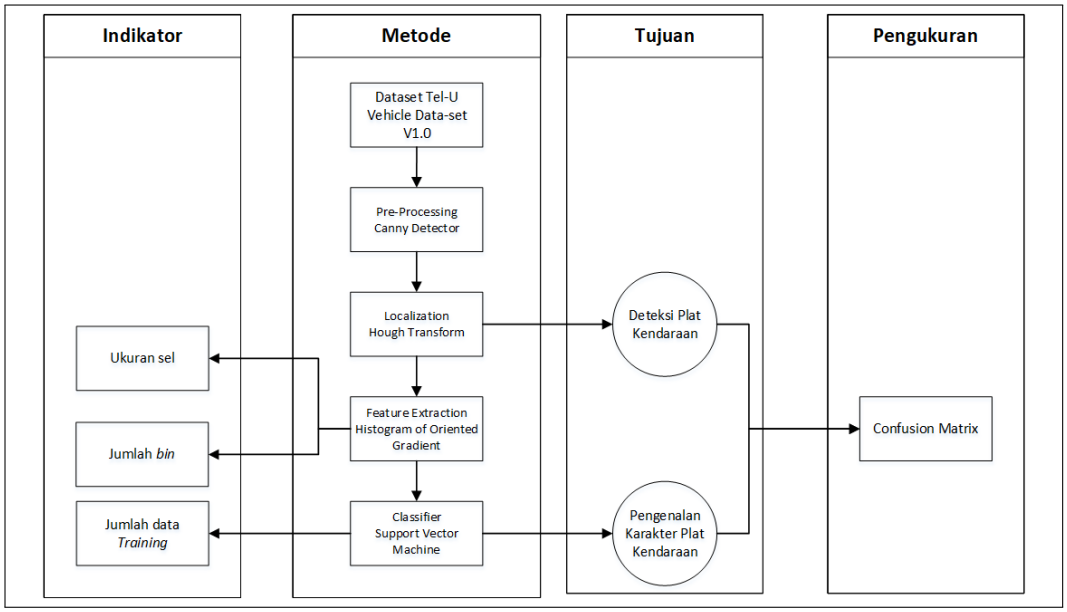
Pada bab 1 telah dijelaskan bahwa penelitian mengenai sistem pengenalan plat nomor kendaraan merupakan bidang yang masih berkembang dan implementasinya memegang peranan penting dalam bidang transportasi. Pada penelitian ini, penulis menggunakan metode *Canny Detector* dan *Hough Transform* untuk mendeteksi lokasi plat kendaraan, kemudian menggunakan metode *Histogram of Oriented Gradient* untuk mengekstraksi fitur dari citra karakter dari plat nomor yang sudah disegmentasi dengan menghitung grafik horizontal pita, kemudian dilakukan klasifikasi dengan menggunakan metode *Support Vector Machine*.

Masukan untuk sistem deteksi dan pengenalan plat nomor kendaraan ini adalah citra yang ditangkap oleh kamera DSLR Canon EOS 500 D dan Canon EOS 550 D beresolusi 15 dan 18 megapiksel. Citra tangkapan kemudian akan diubah resolusinya menjadi 1024×640 piksel. Setiap citra masukan berisi bagian depan dari kendaraan yang memiliki plat nomor kendaraan.

Keluaran atau hasil dari sistem akan berupa teks hasil dari pengenalan karakter pada citra plat nomor kendaraan masukkan.

3.2 Kerangka Pemikiran

Berikut ini adalah kerangka pemikiran dari metode yang diusulkan untuk melakukan deteksi plat nomor kendaraan dan melakukan pengenalan karakter pada citra karakter yang terdapat pada plat nomor.



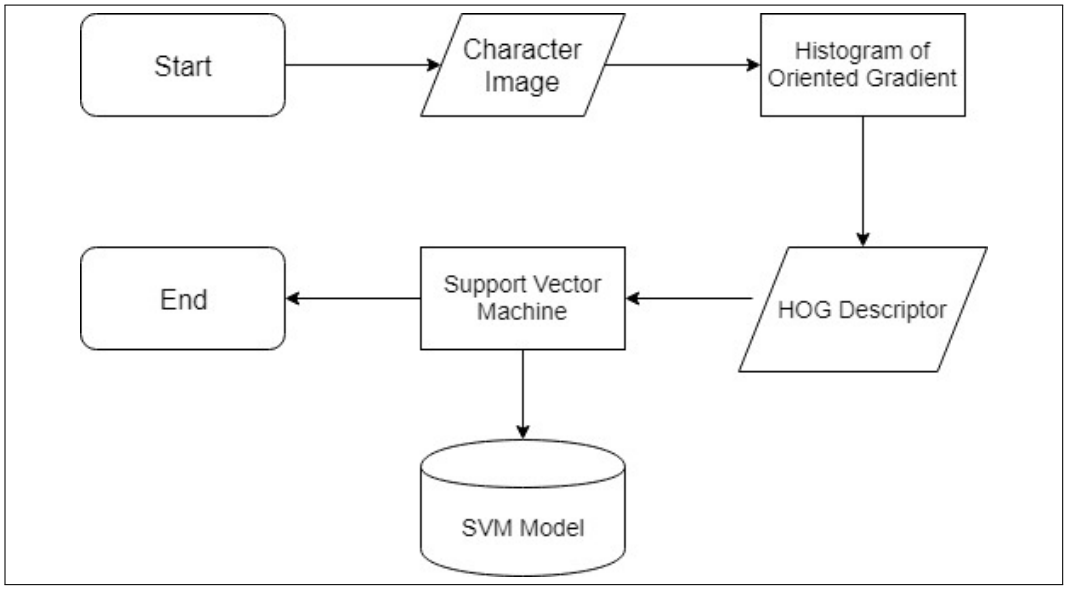
Gambar 3.1 Kerangka Pemikiran

Seperti pada gambar 3.1, terdapat beberapa variabel indikator yang memengaruhi hasil dan perlu dilakukan penyesuaian, seperti ukuran sel pada metode *Histogram of Oriented Gradient*, jumlah *bin* yang menentukan batasan sudut yang digunakan, dan jumlah data *training* untuk *classifier Support Vector Machine*. Penelitian ini memiliki dua tujuan, yaitu untuk melihat hasil akurasi deteksi plat nomor dan akurasi pengenalan karakter pada plat nomor dengan menggunakan *confusion matrix*.

3.3 Urutan Proses Global

Dalam sistem pengenalan plat nomor kendaraan terbagi atas dua proses yaitu proses *training* dan proses *testing*. Proses *training* dilakukan untuk mendapatkan kelas-kelas dari karakter-karakter yang akan dikenali. Proses *testing* dilakukan untuk menghitung hasil yang berupa akurasi dari pengenalan karakter pada plat nomor kendaraan.

3.3.1 Proses *Training*



Gambar 3.2 *Flowchart Training* Sistem Pengenalan Plat Nomor Kendaraan

Berikut ini adalah uraian dari *flowchart* pada gambar 3.2 yang dilakukan dalam penelitian ini:

1. Citra yang menjadi masukan yaitu dari *dataset English Font* yang berasal dari *The Chars 74K image dataset* yang berisi kumpulan jenis karakter dari komputer dengan 4 variasi, yaitu citra karakter miring (*Italic*), citra karakter tebal (**Bold**), dan citra karakter normal. Citra karakter memiliki ukuran 128 $\times$ 128 piksel. Citra karakter berwarna hitam dengan latar belakang berwarna putih. Kumpulan karakter yang digunakan adalah karakter angka dari 0 sampai dengan 9 dan karakter huruf kapital dari A sampai dengan Z, karakter huruf kecil tidak digunakan karena pada plat nomor kendaraan tidak menggunakan karakter huruf kecil.

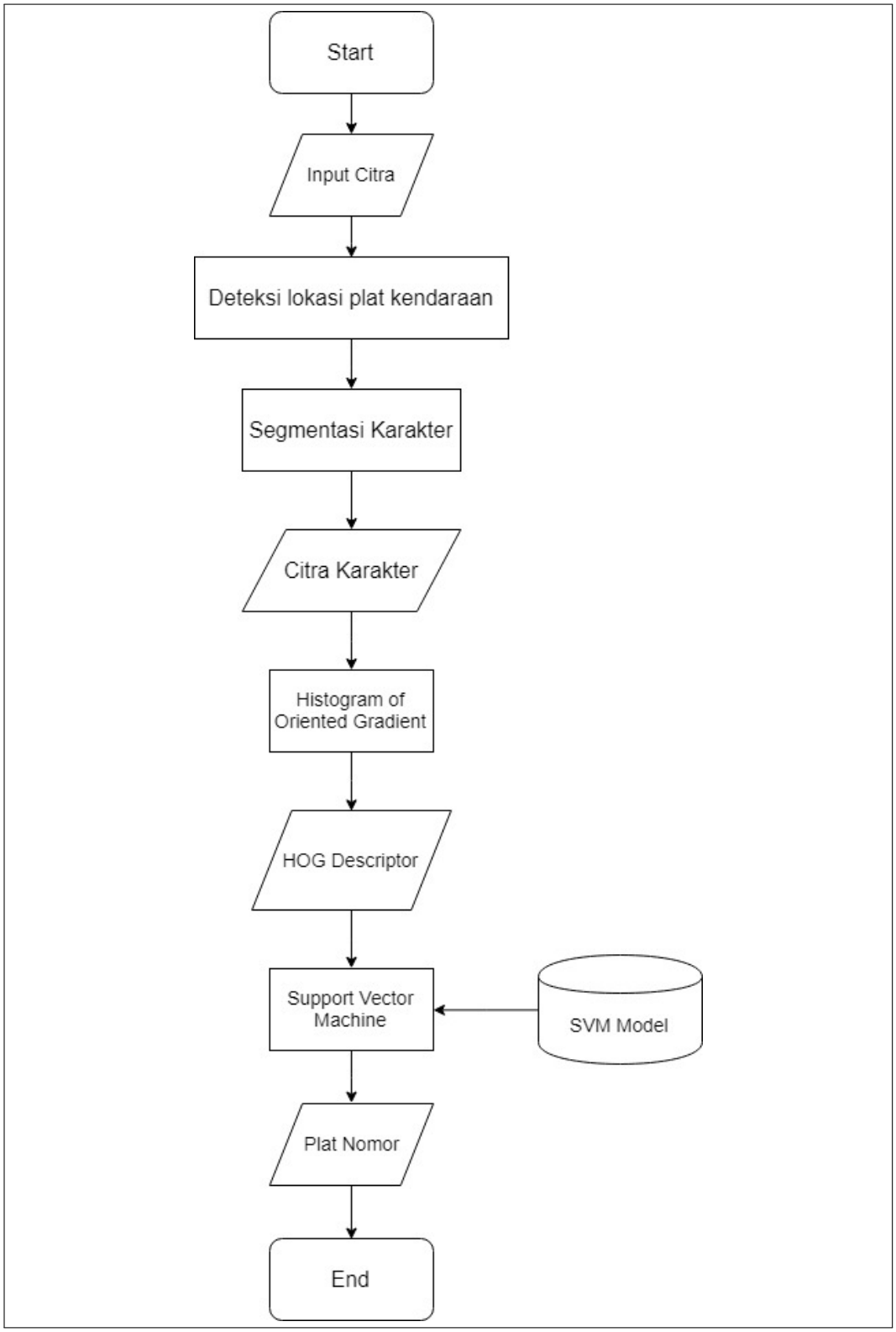
2. *Histogram of Oriented Gradient* berfungsi untuk mendapatkan fitur dari dari citra masukan. Hasil dari ekstraksi fitur dengan menggunakan HOG adalah *HOG descriptor*, yang mendeskripsikan distribusi dari gradien berarah pada suatu area citra.

3. Ukuran sel dan blok yang digunakan untuk proses ekstraksi fitur dengan menggunakan *HOG* adalah 8 $\times$ 8 piksel untuk ukuran sel dan 4 $\times$ 4 sel dan jumlah *bin* yang digunakan pada tahap pembuatan *histogram* adalah 9 dimulai dari 0 derajat hingga 180 derajat.

4. *Support Vector Machine* (SVM) digunakan untuk mengklasifikasikan fitur-fitur yang sudah didapatkan ke dalam kelas-kelas dari karakter yang

akan dikenali. Hasil dari klasifikasi ini nantinya akan disimpan ke dalam bentuk berkas.

3.3.2 Proses *Testing*



Gambar 3.3 *Flowchart Testing* Sistem Pendeteksi dan Pelacakan Manusia

Pada gambar 3.3 terlihat urutan proses *testing*. Pada proses *testing* terdapat beberapa proses yang sama seperti pada proses *training*. Berikut ini adalah uraian

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

1. Citra pengujian yang digunakan didapatkan dari *dataset* plat nomor kendaraan Universitas Telkom yang bernama *Tel-U Vehicle Data-set V1.0*, penggunaan dari dataset ini sesuai dengan perizinan dari institusi yang bersangkutan.
2. Citra yang akan menjadi input dari *HOG* adalah citra hasil dari segmentasi karakter pada citra plat kendaraan pada tahapan deteksi lokasi plat nomor kendaraan.
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 diambil dari berkas yang merupakan hasil keluaran dari SVM pada tahap *training*.
5. Hasil keluaran akan berupa sebuah *string* yang menunjukkan kumpulan karakter yang berhasil dikenali oleh sistem.

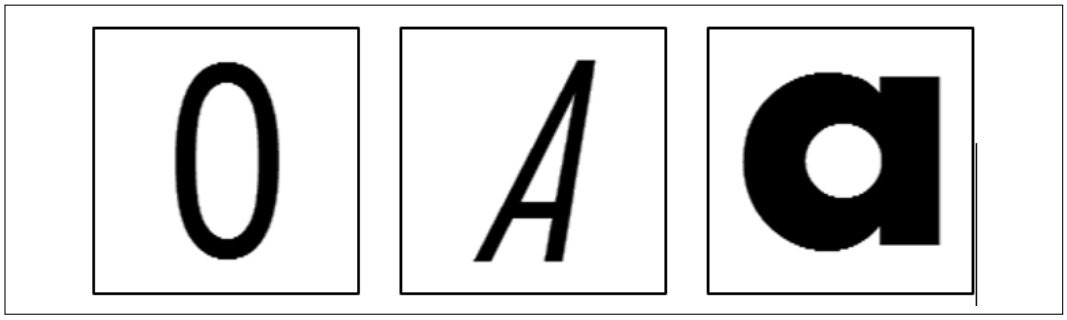
3.4 Analisis Manual

Pada bagian ini dilakukan analisis tahapan proses dengan melakukan perhitungan manual.

3.4.1 *Dataset*

Dari *The Chars 74K image dataset* untuk tahap *training* akan digunakan citra karakter angka dari 0 sampai dengan 9 dan karakter huruf kapital A sampai dengan Z. Terlihat contoh citra karakter yang ditunjukan pada gambar 3.4 merupakan contoh karakter angka dan huruf kapital yang akan dipakai. Sedangkan pada gambar 3.5, gambar tersebut menunjukkan contoh citra karakter yang tidak akan dipakai, yaitu citra karakter tipis, miring, dan karakter huruf kecil. Tidak digunakannya karakter-karakter tersebut dikarenakan pada plat nomor Indonesia, karakter yang digunakan hanyalah karakter huruf kapital dan angka dalam bentuk tegak dan tebal.

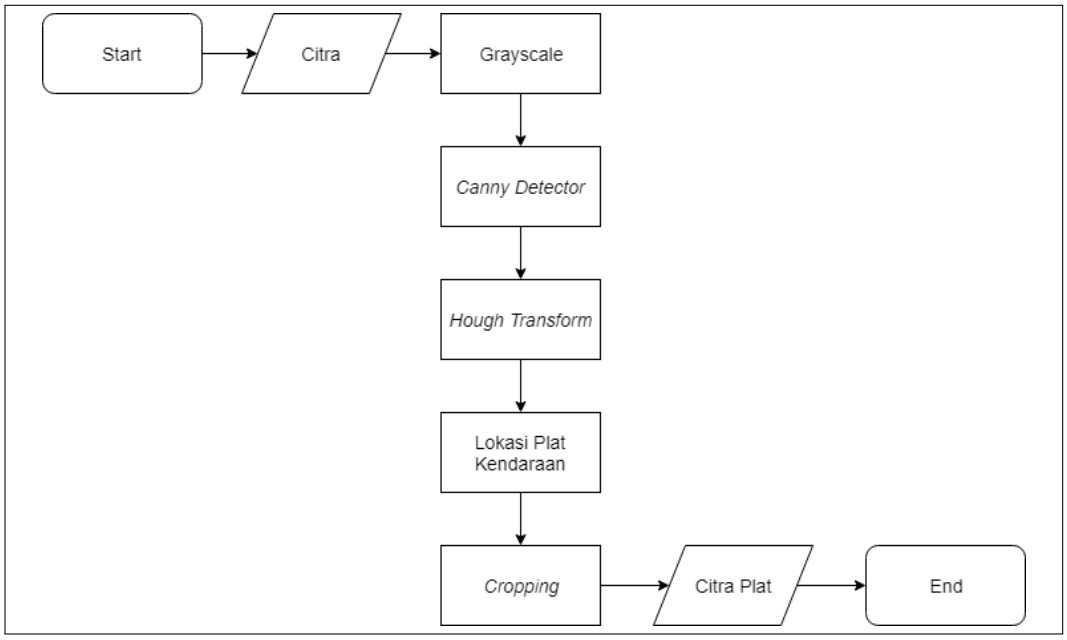
Gambar 3.4 Contoh citra karakter yang digunakan untuk tahap *training*



Gambar 3.5 Contoh citra karakter yang tidak digunakan

3.4.2 Tahap Pendeteksian Lokasi Plat Nomor

Skema alur dari tahap pendeteksian lokasi plat nomor adalah:



Gambar 3.6 Skema Alur Pendeteksian Plat

3.4.2.1 *Grayscale*

Proses pertama adalah mengubah citra masukan menjadi citra *grayscale*, tujuan dari *grayscaling* citra adalah untuk menghilangkan informasi warna dari setiap piksel citra. Untuk menghitung nilai derajat keabuan setiap piksel, diperoleh

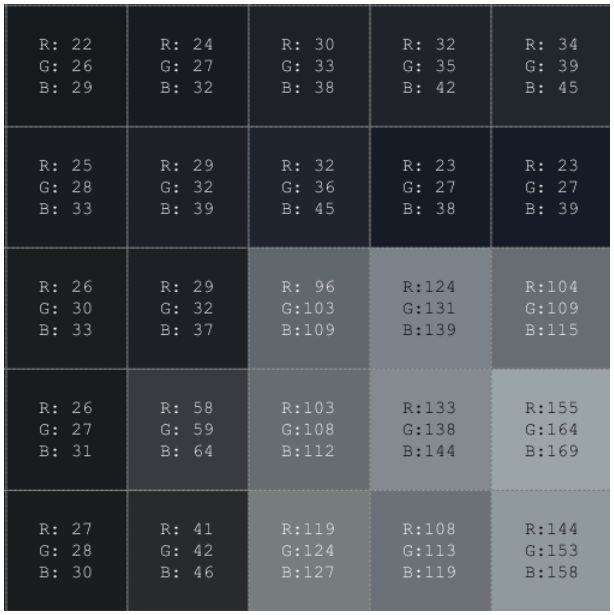
dengan rumus:

$$grayvalue = 0.299R + 0.587G + 0.114B$$

Gambar 3.7 Persamaan untuk mencari *grayscale value*

dimana R, G, B adalah nilai *red*, *green*, *blue* dari masing-masing piksel citra.

Di bawah merupakan contoh matriks citra asli dengan 3 *channel* warna yaitu *Red*, *Green*, dan *Blue* berukuran 5 × 5 piksel yang diambil dengan menggunakan *image tools* dari aplikasi *MatLab*

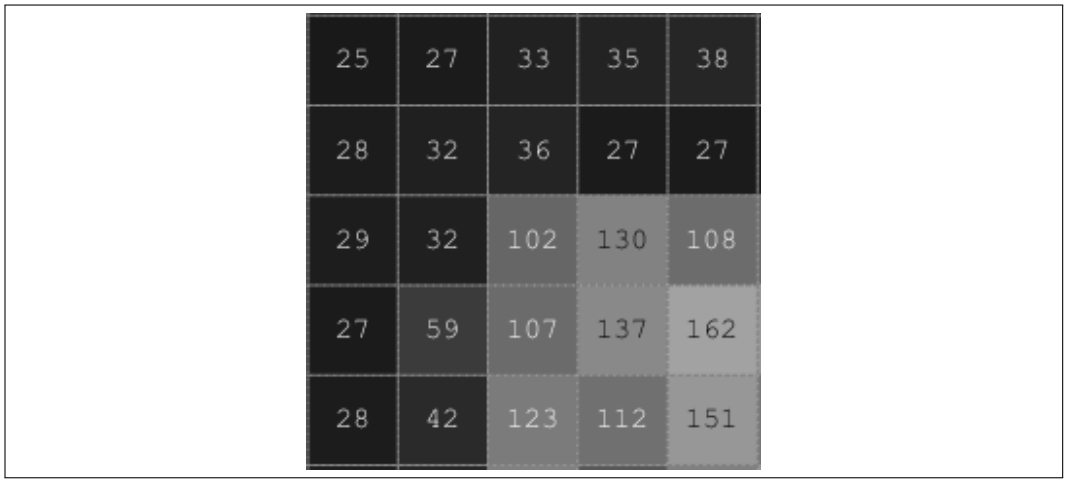


Gambar 3.8 Matriks Citra Asal berukuran 5 × 5

Dengan menggunakan persamaan 3.7, maka nilai matriks citra *grayscale* pada titik (3,3) akan menjadi sebagai berikut:

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

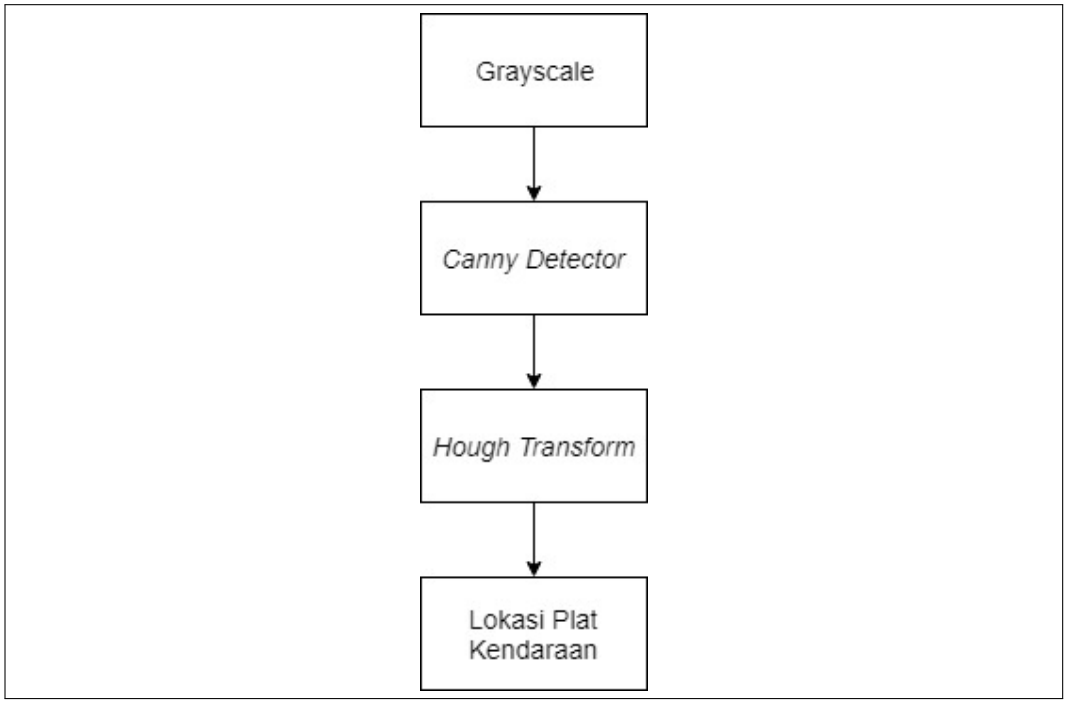
Perhitungan di atas dilakukan terhadap seluruh nilai matriks citra asal dan hasilnya adalah matriks citra berukuran 5 × 5 dengan satu nilai derajat keabuan.



Gambar 3.9 Matriks Citra Hasil Grayscale

3.4.2.2 Deteksi Tepi Canny

Pada bagian ini menjelaskan urutan langkah kerja pada proses Deteksi Tepi Canny. Gambar 3.10 adalah *flowchart* Deteksi Tepi Canny.



Gambar 3.10 Flowchart Deteksi Tepi Canny

Hal pertama yang dilakukan untuk deteksi tepi Canny adalah dengan melakukan *smoothing* pada citra dengan menggunakan turunan pertama pertama dari *Gauss* dengan besar *kernel* 3×3 dan nilai dari $\sigma = 1$ untuk menghilangkan *noise*.

$$Matriks\ Gauss\ Kernel = \begin{bmatrix} 0.077847 & 0.123317 & 0.077847 \\ 0.123317 & 0.195346 & 0.123317 \\ 0.077847 & 0.123317 & 0.077847 \end{bmatrix} \quad (3\ .\ 1)$$

Kernel seperti pada persamaan 3 . 1 dikonvolusikan ke dalam citra untuk mendapatkan sebuah nilai baru. Sebagai contoh apabila diambil area pada sebuah citra sebesar 5 $\times$ 5 piksel seperti pada gambar 3.11 yang kemudian dikonvolusikan menggunakan *kernel* pada persamaan 3 . 1.

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 3.11 Matriks citra ukuran 5 $\times$ 5

Maka pada elemen matriks (3,3) nilainya berubah menjadi:

$$\begin{aligned} Matriks[3,3] = {} & (0.077847 * 102) + (0.123317 * 130) + (0.077847 * 108) \\ & + (0.123317 * 107) + (0.195346 * 137) + (0.123317 * 162) \\ & + (0.077847 * 123) + (0.123317 * 112) + (0.077847 * 151) \end{aligned}$$

$$Matriks[3,3] = 127,45534 \approx 127$$

Sehingga elemen matriks (3,3) berubah nilainya menjadi 127, perhitungan tersebut berlaku untuk semua elemen matriks sisanya, kecuali pada elemen matriks di tepian citra, sehingga seluruh tepian citra sebesar 1 piksel tidak berubah.

Berdasarkan perhitungan di atas, didapatkan hasil matriks citra hasil konvolusi sebagai berikut:

25	27	33	35	38
28	36	48	55	27
29	49	77	97	108
27	59	97	127	162
28	42	123	112	151

Gambar 3.12 Matriks citra ukuran 5 $\times$ 5 hasil konvolusi

Tahap selanjutnya adalah perhitungan gradien dengan menggunakan operator gradien sobel. Sebagai contoh dengan area yang sama seperti yang digunakan pada *Gaussian Filter* pada titik(3,3), perhitungan gradien dilakukan sebagai berikut:

$$G_x = (-1*77)+(0*97)+(1*108)+(-2*97)+(0*127)+(2*162)+(-1*123)+(0*112)+(1*151) = 188$$
$$G_y = (-1*77)+(-2*97)+(-1*108)+(0*97)+(0*127)+(0*162)+(1*123)+(2*112)+(1*151) = 118$$
$$m = \sqrt{G_x^2+G_y^2} = \sqrt{188^2+118^2} = 221.716$$
$$\theta = arctan(\frac{118}{188}) = 32.122$$

Gambar 3.13 Perhitungan gradien dan arah

Dari hasil perhitungan di atas, didapatkan informasi bahwa titik (3,3) merupakan tepian yang memiliki arah 32.122° dengan *magnitude* sebesar 221.716. Jika *gradient magnitude* lebih dari 255 maka piksel akan menjadi 255 dan jika *gradient magnitude* kurang dari 0 maka piksel menjadi 0. Citra hasil setelah perhitungan gradien adalah sebagai berikut:

0	0	0	0	0
0	134.141	197.325	239.217	0
0	209.829	255	255	0
0	255	255	221.716	0
0	0	0	0	0

Gambar 3.14 Matriks citra hasil perhitungan gradien

Tahap setelah perhitungan gradien adalah *non-maximum suppression*, tahap ini membandingkan apakah gradien tersebut paling maksimum diantara tetangganya dan di atas nilai atas *threshold* yang sudah ditentukan. Jika nilai gradien di titik tersebut lebih besar dari *high threshold* dan merupakan nilai maksimum dari gradien-gradien tetangganya, maka nilai tersebut merupakan nilai tepi. Nilai tepi hasil *non-maximum suppression* digunakan untuk menghitung *Hysterysis thresholding*. Cara yang digunakan adalah dengan menentukan jika titik tersebut bukan tepi periksa titik selanjutnya, selain itu lihat apakah gradien lebih besar dari

batas *low threshold*. Cara ini diulang sampai tidak ada lagi perubahan.

3.4.2.3 *Hough Transform*

Metode *Hough Transform* yang digunakan adalah untuk identifikasi garis lurus. Dalam ekstraksi fitur *Hough Transform* perlu mespesifikasikan *accumulator space* untuk menyimpan nilai *voting*. Spesifikasi *accumulator space* ditentukan berdasarkan ukuran citra input. Dalam contoh ini digunakan citra input berukuran 5×6 piksel yang merupakan hasil proses deteksi tepi pada tahapan sebelumnya. Berikut spesifikasi matriks *accumulator space* yang digunakan sebagai contoh:

1. Rentang nilai *rho* dapat dicari dengan (*width* dan *height* masing-masing dikurang 1 karena koordinat x dan y dimulai dari 0,0):

$$D = \sqrt{(5-1)^2 + (6-1)^2} = 6,403$$

   Nilai D dibulatkan ke bawah, maka nilai D adalah 6. Karena nilai D yang didapat adalah 6 maka nilai *rho* yang digunakan adalah:

$$-6 \leq \rho \leq 6$$

2. Nilai *theta* yang digunakan adalah 0 hingga 180 derajat.

3. Inisialisasi nilai 0 untuk setiap sel dalam matriks *accumulator space*.

Dari spesifikasi di atas maka keadaan awal matriks *accumulator space* dapat dilihat pada tabel 3.1. Setelah membuat matriks *accumulator space*, selanjutnya menghitung voting dari setiap pasangan *rho* dan *theta*. Untuk menghitung nilai *rho* dapat dilakukan dengan menggunakan persamaan 2 . 2. Dalam metode *Hough Transform* hanya akan melakukan pengecekan terhadap objek berwarna putih atau yang memiliki nilai 255.

Tabel 3.1 Inisialisasi matriks *Accumulator Space*

6	0	0	0	0	0	0	0	
5	0	0	0	0	0	0	0	
4	0	0	0	0	0	0	0	
3	0	0	0	0	0	0	0	
2	0	0	0	0	0	0	0	
1	0	0	0	0	0	0	0	
0	0	0	0	0	0	0	0	...
-1	0	0	0	0	0	0	0	
-2	0	0	0	0	0	0	0	
-3	0	0	0	0	0	0	0	
-4	0	0	0	0	0	0	0	
-5	0	0	0	0	0	0	0	
-6	0	0	0	0	0	0	0	
	0	1	2	3	4	5	6	...

Pada tabel 3.1, sumbu X menggambarkan *theta* sedangkan sumbu Y menggambarkan *rho*.

Diasumsikan baris 1 dan kolom 1 adalah x = 0 dan y = 0. Misalkan dilakukan pengecekan untuk baris 1 kolom 1. Ternyata baris 1 kolom 1 memiliki nilai 255 (objek berwarna putih) karena itu dihitung nilai *rho* untuk sudut 0 hingga 180 derajat. Berikut adalah uraiannya:

1. Untuk sudut 0 derajat:
   $\rho = x * \cos\theta + y * \sin\theta$
   $\rho = 0 * \cos 0 + 0 * \sin 0$
   $\rho = 0 * 1 + 0 * 0$
   $\rho = 0$
   Tambahkan nilai *voting* untuk *rho* = 0 dengan *theta* = 0 sebesar 1.

2. Untuk sudut 1 derajat:
   $\rho = x * \cos\theta + y * \sin\theta$
   $\rho = 0 * \cos 1 + 0 * \sin 1$
   $\rho = 0 * 0,99984 + 0 * 0,01745$
   $\rho = 0$
   Tambahkan nilai *voting* untuk *rho* = 0 dengan *theta* = 1 sebesar 1.

3. Untuk sudut 2 derajat:
   $\rho = x * \cos\theta + y * \sin\theta$
   $\rho = 0 * \cos 2 + 0 * \sin 2$
   $\rho = 0 * 0,99939 + 0 * 0,03489$
   $\rho = 0$

Tambahkan nilai *voting* untuk *rho* = 0 dengan *theta* = 2 sebesar 1.

4. Untuk sudut 3 derajat:
   $\rho = x * \cos\theta + y * \sin\theta$
   $\rho = 0 * \cos 3 + 0 * \sin 3$
   $\rho = 0 * 0,99862 + 0 * 0,05233$
   $\rho = 0$
   Tambahkan nilai *voting* untuk *rho* = 0 dengan *theta* = 3 sebesar 1.

5. Untuk sudut 4 derajat:
   $\rho = x * \cos\theta + y * \sin\theta$
   $\rho = 0 * \cos 4 + 0 * \sin 4$
   $\rho = 0 * 0,99756 + 0 * 0,06975$
   $\rho = 0$
   Tambahkan nilai *voting* untuk *rho* = 0 dengan *theta* = 4 sebesar 1.

6. Untuk sudut 5 derajat:
   $\rho = x * \cos\theta + y * \sin\theta$
   $\rho = 0 * \cos 5 + 0 * \sin 5$
   $\rho = 0 * 0,99619 + 0 * 0,08715$
   $\rho = 0$
   Tambahkan nilai *voting* untuk *rho* = 0 dengan *theta* = 5 sebesar 1.

7. Untuk sudut 6 derajat:
   $\rho = x * \cos\theta + y * \sin\theta$
   $\rho = 0 * \cos 6 + 0 * \sin 6$
   $\rho = 0 * 0,99452 + 0 * 0,10452$
   $\rho = 0$
   Tambahkan nilai *voting* untuk *rho* = 0 dengan *theta* = 6 sebesar 1.

Cara di atas diulang untuk 7 derajat hingga 180 derajat.

Setelah menyelesaikan perhitungan untuk titik koordinat (0,0) maka berikut adalah perubahan yang ditunjukan oleh matriks *accumulator space*:

Tabel 3.2 Nilai *voting* koordinat (0,0) disimpan di matriks *Accumulator Space*

6	0	0	0	0	0	0	0	
5	0	0	0	0	0	0	0	
4	0	0	0	0	0	0	0	
3	0	0	0	0	0	0	0	
2	0	0	0	0	0	0	0	
1	0	0	0	0	0	0	0	
0	1	1	1	1	1	1	1	...
-1	0	0	0	0	0	0	0	
-2	0	0	0	0	0	0	0	
-3	0	0	0	0	0	0	0	
-4	0	0	0	0	0	0	0	
-5	0	0	0	0	0	0	0	
-6	0	0	0	0	0	0	0	
	0	1	2	3	4	5	6	...

Proses di atas diulangi untuk semua piksel (nilai x dan y yang berbeda) dalam citra (hanya piksel yang bernilai 255 saja yang dihitung jaraknya). Jika sudah diterapkan untuk setiap kombinasi x dan y maka keluaran yang didapat adalah:

Tabel 3.3 Matriks *Accumulator Space* dan nilai *voting* yang disimpan

6	0	0	0	0	0	0	0	
5	0	0	0	0	0	0	0	
4	0	0	0	0	0	0	0	
3	0	0	0	0	0	0	0	
2	0	0	0	0	0	0	0	
1	0	0	0	0	0	0	0	
0	5	6	6	9	6	3	5	...
-1	5	5	5	5	5	5	5	
-2	5	4	4	4	4	4	5	
-3	3	4	4	4	4	4	3	
-4	4	3	3	3	3	3	4	
-5	0	0	0	0	0	0	0	
-6	0	0	0	0	0	0	0	
	0	1	2	3	4	5	6	...

Setelah mendapat matriks *accumulator space* selanjutnya adalah mencari nilai *Hough Peaks*. Untuk menentukan nilai *Hough Peaks* diperlukan untuk menentukan nilai *threshold*, *neighbourhood*, dan jumlah *peaks*. Misalkan dalam kasus ini ditentukan nilai *threshold* = 3, neighbourhood = 1, dan *peaks* = 3, maka:

1. Dicari nilai yang memiliki nilai *vote* di atas nilai *threshold*, maka nilai pertama yang diperiksa adalah (diberi warna oranye):

Tabel 3.4 Koordinat (0,0) memiliki nilai di atas *threshold*

6	0	0	0	0	0	0	0	
5	0	0	0	0	0	0	0	
4	0	0	0	0	0	0	0	
3	0	0	0	0	0	0	0	
2	0	0	0	0	0	0	0	
1	0	0	0	0	0	0	0	
0	5	6	6	9	6	3	5	...
-1	5	5	5	5	5	5	5	
-2	5	4	4	4	4	4	5	
-3	3	4	4	4	4	4	3	
-4	4	3	3	3	3	3	4	
-5	0	0	0	0	0	0	0	
-6	0	0	0	0	0	0	0	
	0	1	2	3	4	5	6	...

2. Memeriksa *neighbour* apakah terdapat nilai vote yang memiliki nilai *vote* lebih tinggi, jika ada tetangganya yang memiliki nilai *vote* lebih tinggi maka nilai tersebut ditolak dan sebaliknya jika tidak maka nilai tersebut dipilih sebagai peak:

Tabel 3.5 Memeriksa *neighbour* dari koordinat (0,0)

6	0	0	0	0	0	0	0		0
5	0	0	0	0	0	0	0		0
4	0	0	0	0	0	0	0		2
3	0	0	0	0	0	0	0		2
2	0	0	0	0	0	0	0		4
1	0	0	0	0	0	0	0		5
0	5	6	6	9	6	3	5	...	9
-1	5	5	5	5	5	5	5		0
-2	5	4	4	4	4	4	5		0
-3	3	4	4	4	4	4	3		0
-4	4	3	3	3	3	3	4		0
-5	0	0	0	0	0	0	0		0
-6	0	0	0	0	0	0	0		0
	0	1	2	3	4	5	6	...	

Karena tetangga dari (0,0) terdapat nilai *vote* yang lebih tinggi dari 5 (nilai 9 pada (0,180)) maka nilai *vote* pada (0,0) tidak diterima sebagai *peak*. Cara di atas diulang hingga mendapatkan seluruh *peak* potensial yang ada. Misalkan didapatkan 5 *peak* potensial sebagai berikut:

Tabel 3.6 *Peak* yang belum terurut

Peak	1	2	3	4	5
Jumlah Voting	6	9	5	4	4
Theta	1	3	6	3	10

Setelah mendapatkan sejumlah nilai *peak* potensial kemudian *peak* potensial tersebut diurutkan dari nilai terbesar hingga terkecil (*descending*) berdasarkan nilai *voting*. Maka nilai dari pengurutan *peaks* potensial tersebut adalah:

Tabel 3.7 *Peak* terurut secara *descending* berdasarkan jumlah *voting*

Peak	1	2	3	4	5
Jumlah Voting	9	6	5	4	4
Theta	3	1	6	3	10

3. Karena jumlah *peaks* yang ditentukan adalah 3 maka ambil 3 *peaks* dengan nilai *voting* tiga tertinggi:

Tabel 3.8 Tiga *peaks* yang terpilih

Peak	1	2	3
Jumlah Voting	9	6	5
Theta	3	1	6

4. Setelah mendapatkan sejumlah *peaks* yang diinginkan berikutnya adalah menghitung intensitas kemunculan dari setiap nilai *theta* (rentang niali *theta* yang digunakan adalah 0-180 derajat). Misalkan hasil perhitungan intensitas kemunculan tersebut adalah:

Tabel 3.9 Memeriksa *neighbour* dari koordinat (0,0)

6	0	0	0	0	0	0	0		0
5	0	0	0	0	0	0	0		0
4	0	0	0	0	0	0	0		2
3	0	0	0	0	0	0	0		2
2	0	0	0	0	0	0	0		4
1	0	0	0	0	0	0	0		5
0	5	6	6	9	6	3	5	...	9
-1	5	5	5	5	5	5	5		0
-2	5	4	4	4	4	4	5		0
-3	3	4	4	4	4	4	3		0
-4	4	3	3	3	3	3	4		0
-5	0	0	0	0	0	0	0		0
-6	0	0	0	0	0	0	0		0
	0	1	2	3	4	5	6	...	

3.4.3 *Histogram of Oriented Gradient*

Pada proses *Histogram of Oriented Gradients*, masukan untuk proses ini berupa citra yang berasal dari hasil *preprocessing*. Keluaran dari proses ini adalah matriks fitur vektor dari hasil perhitungan *Histogram of Oriented Gradients*. Berikut merupakan langkah-langkah untuk menghitung matriks fitur vektor. Pada gambar dapat dilihat hasil dari proses *resize* dan *crop* citra *grayscale* berukuran 8 $\times$ 4 piksel.

89	92	88	92
90	88	90	86
91	90	90	94
91	122	91	122
89	90	89	91
90	85	90	86
91	90	92	93
91	122	91	120

Gambar 3.15 Matriks citra hasil *preprocessing*

1. Proses pertama adalah untuk menghitung nilai gradien dari posisi vertikal dan horizontal untuk setiap piksel menggunakan persamaan 2 . 6 dan 2 . 7. Contoh perhitungannya untuk piksel koordinat (2,5) dan hasil dari tahap ini dapat dilihat pada gambar 3.16 dan 3.17 di bawah:

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

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

0	-1	0	0
0	0	-2	0
0	-1	4	0
0	0	0	0
0	0	1	0
0	0	1	0
0	1	3	0
0	0	-2	0

Gambar 3.16 Matriks hasil Perhitungan Gradien sumbu X

0	0	0	0
2	-2	2	2
1	34	1	36
-2	0	-1	-3
-1	-37	-1	-36
2	0	3	2
1	37	1	34
0	0	0	0

Gambar 3.17 Matriks hasil Perhitungan Gradien sumbu Y

2. Untuk setiap piksel, hitung *magnitude* gradien dan arah gradien menggunakan persamaaan 2 . 8 dan 2 . 9. Contoh perhitungannya untuk piksel koordinat (2,5) dan hasil dari tahap ini dapat dilihat pada gambar di bawah:

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

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

0	1	0	0
2	2	2	2
1	34	4	36
2	0	1	3
1	37	1	36
2	0	3	2
1	37	3	34
0	0	2	0

Gambar 3.18 Matriks hasil Perhitungan *Magnitude*

90	0	90	90
90	90	45	90
90	89	14	90
90	90	90	90
90	90	45	90
90	90	71	90
90	88	18	90
90	90	0	90

Gambar 3.19 Matriks hasil Perhitungan Arah

3. Kemudian, tentukan ukuran sel, ukuran blok dan jumlah *oriented histogram bins*. Pada penelitian Dalas dan Triggs untuk mendeteksi pejalan kaki

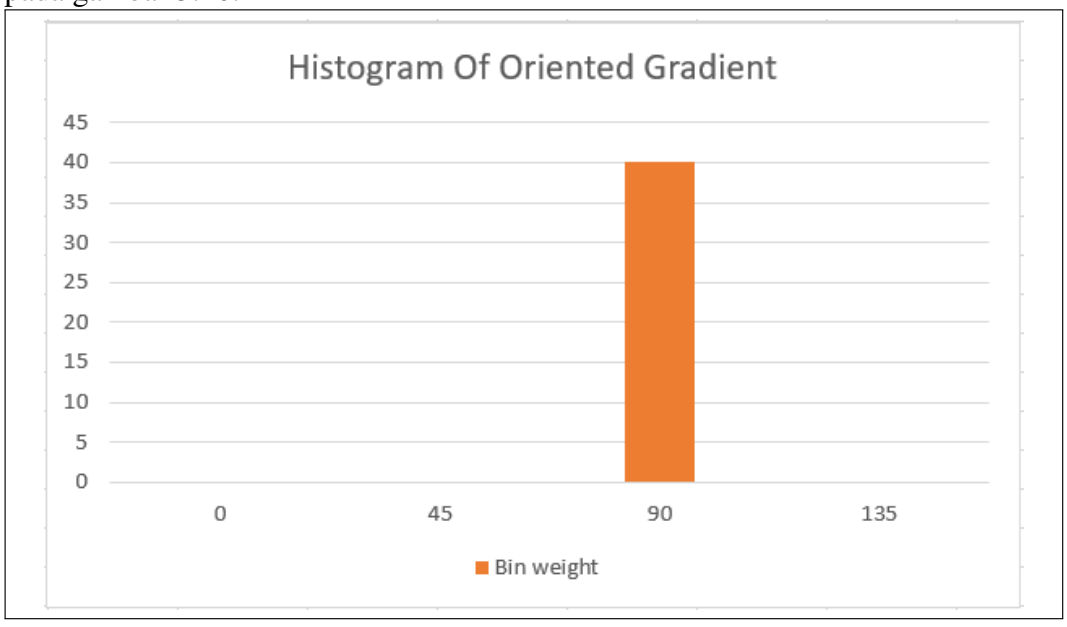
sebelumnya, didapat bahwa ukuran sel sebesar 8 × 8 piksel, ukuran blok sebesar 2 × 2 ukuran sel dan jumlah bin sebanyak 9 sudah dapat menghasilkan akurasi yang dapat mendeteksi pejalan kaki dengan cukup baik dibandingkan dengan ukuran-ukuran lainnya. Untuk contoh perhitungan analisis kali ini jumlah *oriented histogram bins* yang dipakai sebanyak 4 buah, sehingga didapat nilai sudut setiap *histogram bin* yaitu 180 / 4 = 45. Untuk ukuran sel dipilih sebesar 2 × 2 piksel dan ukuran blok sebesar 2 × 2 sel.

4. Kemudian untuk setiap sel, tentukan perhitungan *Histogram of Oriented Gradient* dengan melakukan *voting* dari arah gradien dan *magnitude* gradien, dimana arah gradien akan menjadi sudut *bin*, dan *magnitude* gradien akan menjadi bobot nilai. 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 sehingga masing-masing sel akan mempunyai *Histogram of Oriented Gradient*. Berikut contoh hasil perhitungan metode *Histogram of Oriented Gradient* pada sel yang terdapat koordinat piksel (2,5) dapat dilihat pada gambar 3.20.



Gambar 3.20 Contoh hasil *Histogram of Oriented Gradient* untuk sel yang memiliki piksel dengan koordinat (2,5)

5. Kemudian untuk setiap blok, akan dilakukan normalisasi dengan menggabungkan hasil histogram dari setiap sel dalam bloknya. Adapun proses normalisasi dapat menggunakan 4 algoritma yaitu, *L1-Norm*, *L1-Sqrt*, *L2-Norm*, dan *L2-Hys*. Pada penelitian ini, penulis menggunakan algoritma normalisasi *L2-Norm* karena berdasarkan penelitian sebelumnya, hasil yang didapat lebih baik dari algoritma lainnya. Persamaan algoritma untuk proses normalisasi menggunakan *L2-Norm* didapat dengan menggunakan persamaan 2 . 12. Di bawah adalah contoh perhitungan normalisasi untuk blok pertama:

1	0	4	0	0	2	2	0
0	0.76	36.24	0	2.76	1.24	40	0
0	0	40	0	0	2.27	39.73	0
0	1.65	36.35	0	3.8	1.2	34	0

Gambar 3.21 Matriks hasil Perhitungan Histogram untuk seluruh sel

Berdasarkan matriks pada gambar 3.21. Elemen matriks yang akan kita gunakan dalam perhitungan normalisasi ini adalah seluruh elemen baris pertama dan baris kedua.

$$L2_{Norm} = \sqrt{1^2 + 0^2 + \ldots + 2^2 + 0^2 + 0^2 + 0.76^2 + \ldots + 40^2 + 0^2} = 54.29$$

Kemudian untuk setiap nilai dari histogram dari sel dalam blok tersebut akan dibagi dengan nilai hasil normalisasinya. Di bawah adalah contoh hasil normalisasi histogram dari sel pertama (matriks hasil perhitungan histogram baris pertama kolom 1-4):

$$\begin{bmatrix} 0.018418 & 0 & 0.07367 & 0 \end{bmatrix}$$

Lakukan proses normalisasi untuk setiap blok dengan menggeser secara horizontal sejauh 1 kali ukuran sel dan secara vertikal sejauh 1 kali ukuran sel sampai blok tersebut sudah berada di bawah kanan dari citra. Kemudian hasil dari proses normalisasi akan disusun menjadi matriks besar dengan jumlah kolom sebesar *jumlah bin* $\times$ *lebar blok dalam satuan sel* $\times$ *jumlah pergeseran horizontal* dan jumlah baris sebesar *jumlah pergeseran vertikal* $\times$ *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. Di bawah adalah hasil fitur vektor untuk metode *Histogram of Oriented*

Gradient setelah melewati proses normalisasi.

0.0184	0	0.0736	0	0	0.0368	0.0368	0
0	0.0139	0.6674	0	0.0508	0.0228	0.7367	0
0	0.0097	0.4637	0	0.0353	0.0158	0.5118	0
0	0	0.5118	0	0	0.0290	0.5084	0
0	0	0.5307	0	0	0.0301	0.5271	0
0	0.0218	0.4823	0	0.0504	0.0159	0.4511	0

Gambar 3.22 Matriks hasil Normalisasi

Matriks inilah yang akan dijadikan sebagai masukan bagi metode *Machine Learning* yang akan digunakan dalam penelitian ini.

DAFTAR REFERENSI

[1] Tabrizi, S. S., Cavus, N. (2016). A hybrid KNN-SVM model for Iranian license plate recognition. *Procedia Computer Science, 102*, pp. 588-594.

[2] Gou, C., Wang, K., Yu, Z., Xie, H. (2014, October). License plate recognition using MSER and HOG based on ELM. In *Proceedings of 2014 IEEE International Conference on Service Operations and Logistics, and Informatics* (pp. 217-221). IEEE.

[3] Gou, C., Wang, K., Yao, Y., Li, Z. (2016). Vehicle license plate recognition based on extremal regions and restricted Boltzmann machines. *IEEE Transactions on Intelligent Transportation Systems, 17*(4), 1096-1107.

[4] Rasheed, S., Naeem, A., Ishaq, O. (2012, October). Automated number plate recognition using hough lines and template matching. In *Proceedings of the World Congress on Engineering and Computer Science* (Vol. 1, pp. 24-26).

[5] R. C. Gonzalez and R. E. Woods, *Digital Image Processing*, 2^{nd} ed. Prentice Hall, 1992.

[6] Shih, F.Y. (2010). Image Processing And Pattern Recognition Fundamentals and Techniques, Hoboken:John Wiley & Sons, Inc.

[7] Computer Vision CITS4240: Lab 6, [Online]. Available: http://teaching.csse.uwa.edu.au/units/CITS4240/Labs/Lab6/lab6.html [Accessed: 16-Apr-2019].

[8] Oechsle, Olly (2012). Finding Straight Lines with the Hough Transform, [Online]. Available: http://vase.essex.ac.uk/software/HoughTransform/ [Accessed: 16-Apr-2019].

[9] Ma, Y., & Guo, G. (Eds.). (2014). *Support Vector Machines Applications* (pp. 23-26). New York: Springer.

[10] H. K. Ragb and V. K. Asari, *Multi-feature Fusion and PCA Based Approach for Efficient Human Detection*, Applied Imagery Pattern Recognition Workshop (AIRP) IEEE, Washington, DC, USA, 2016, pp. 1-6.

[11] Ashtari, A. H., Nordin, M. J., Fathy, M. (2014). An Iranian license plate recognition system based on color features. *IEEE transactions on intelligent transportation systems*, 15(4), 1690-1705.

[12] https://homepages.inf.ed.ac.uk/rbf/HIPR2/gryimage.htm