

**PENERAPAN METODE HISTOGRAM OF ORIENTED
GRADIENT UNTUK EKSTRAKSI FITUR PADA SISTEM
PENGENALAN PLAT NOMOR KENDARAAN**

TUGAS AKHIR

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

Disusun Oleh:

Jeffry Saputra

1115040



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

DAFTAR ISI

| | |
|---|------------|
| LEMBAR PENGESAHAN | i |
| LEMBAR PERNYATAAN HASIL KARYA PRIBADI | i |
| ABSTRAK | i |
| ABSTRACT | i |
| PEDOMAN PENGGUNAAN TUGAS AKHIR | i |
| KATA PENGANTAR | i |
| DAFTAR ISI | iii |
| DAFTAR TABEL | iv |
| DAFTAR GAMBAR | vi |
| I PENDAHULUAN | 1-1 |
| 1.1 Latar Belakang | 1-1 |
| 1.2 Rumusan Masalah | 1-2 |
| 1.3 Tujuan Penelitian | 1-2 |
| 1.4 Batasan Masalah | 1-3 |
| 1.5 Kontribusi Penelitian | 1-3 |
| 1.6 Metodologi Penelitian | 1-3 |
| 1.7 Sistematika Pembahasan | 1-4 |
| II LANDASAN TEORI | 2-1 |
| 2.1 Tinjauan Pustaka | 2-1 |
| 2.1.1 Citra Digital | 2-1 |
| 2.1.2 Pengolahan Citra | 2-1 |
| 2.1.3 Pengabuan Citra | 2-2 |
| 2.1.4 Deteksi Tepi | 2-2 |
| 2.1.5 <i>Hough Transform</i> | 2-3 |
| 2.1.6 Segmentasi | 2-5 |
| 2.1.7 Fitur pada Citra | 2-5 |
| 2.1.8 <i>Histogram of Oriented Gradient</i> | 2-6 |

| | | |
|--|---|------------|
| 2.1.9 | <i>Support Vector Machine</i> | 2-8 |
| 2.1.10 | <i>Confusion Matrix</i> | 2-11 |
| 2.1.11 | <i>Character Recognition Rate</i> | 2-13 |
| 2.1.12 | <i>Overall Performance</i> | 2-13 |
| 2.1.13 | Penggunaan <i>Library</i> | 2-13 |
| 2.2 | Tinjauan Studi | 2-16 |
| 2.2.1 | <i>State of the Art</i> | 2-16 |
| 2.2.2 | Pembahasan Penelitian Terkait | 2-18 |
| 2.3 | Tinjauan Objek | 2-19 |
| 2.3.1 | Tanda Nomor Kendaraan Bermotor | 2-20 |
| 2.3.2 | Jenis TNKB | 2-21 |
| 2.3.3 | <i>Dataset Tel-U Vehicle Data-set V1.0</i> | 2-22 |
| III ANALISIS DAN PERANCANGAN SISTEM | | 3-1 |
| 3.1 | Analisis Masalah | 3-1 |
| 3.2 | Kerangka Pemikiran | 3-1 |
| 3.3 | Urutan Proses Global | 3-2 |
| 3.4 | Analisis Manual | 3-5 |
| 3.4.1 | <i>Dataset</i> | 3-6 |
| 3.4.2 | Tahap Pendeteksian Lokasi Plat Nomor | 3-7 |
| 3.4.3 | Tahapan Segmentasi Karakter | 3-12 |
| 3.4.4 | <i>Histogram of Oriented Gradient</i> | 3-13 |
| 3.4.5 | <i>Support Vector Machine</i> | 3-18 |
| IV IMPLEMENTASI DAN PENGUJIAN | | 4-1 |
| 4.1 | Lingkungan Implementasi | 4-1 |
| 4.1.1 | Spesifikasi Perangkat Keras | 4-1 |
| 4.1.2 | Lingkungan Perangkat Lunak | 4-1 |
| 4.2 | Implementasi Perangkat Lunak | 4-1 |
| 4.2.1 | Daftar <i>Class</i> dan <i>Method</i> Gradient | 4-1 |
| 4.2.2 | Daftar <i>Class</i> dan <i>Method</i> GradientCell | 4-3 |
| 4.2.3 | Daftar <i>Class</i> dan <i>Method</i> HOG | 4-3 |
| 4.2.4 | Daftar <i>Class</i> dan <i>Method</i> SVM | 4-4 |
| 4.2.5 | Daftar <i>Class</i> dan <i>Method</i> ConfusionMatrix | 4-5 |
| 4.2.6 | Tampilan Antarmuka Antar Aplikasi | 4-6 |
| 4.2.7 | Implementasi Validasi Plat Kendaraan | 4-9 |
| 4.3 | Pengujian | 4-11 |
| 4.3.1 | Pengujian Kombinasi Parameter | 4-11 |

| | | |
|----------|--|------------|
| 4.3.2 | Pengujian Kombinasi Parameter Campuran | 4-21 |
| 4.4 | Analisis Pengujian | 4-23 |
| 4.5 | Analisis Kesalahan | 4-25 |
| V | PENUTUP | 5-1 |
| 5.1 | Kesimpulan | 5-1 |
| 5.2 | Saran | 5-2 |
| | DAFTAR REFERENSI | vii |

DAFTAR TABEL

| | | |
|------|--|------|
| 2.1 | Tabel fungsi <i>Library</i> OpenCV | 2-13 |
| 2.2 | Tabel fungsi <i>Library</i> Weka | 2-15 |
| 2.3 | Tabel fungsi <i>Library</i> JavaOCR | 2-15 |
| 2.4 | <i>State of the Art</i> | 2-16 |
| 4.1 | Daftar <i>Method Class Gradient</i> | 4-2 |
| 4.2 | Daftar <i>Method Class GradientCell</i> | 4-3 |
| 4.3 | Daftar <i>Method Class HOG</i> | 4-3 |
| 4.3 | Daftar <i>Method Class HOG</i> | 4-4 |
| 4.4 | Daftar <i>Method Class SVM</i> | 4-4 |
| 4.5 | Daftar <i>Method Class ConfusionMatrix</i> | 4-5 |
| 4.5 | Daftar <i>Method Class ConfusionMatrix</i> | 4-6 |
| 4.6 | Klasifikasi karakter dengan parameter CellSize = 2, NumBins = 4, dan Sigma = 0.01 | 4-13 |
| 4.7 | Klasifikasi karakter dengan parameter CellSize = 4, NumBins = 9, dan Sigma = 0.01 | 4-15 |
| 4.7 | Klasifikasi karakter dengan parameter CellSize = 4, NumBins = 9, dan Sigma = 0.01 | 4-16 |
| 4.8 | Klasifikasi karakter dengan parameter CellSize = 8, NumBins = 18, dan Sigma = 0.1 | 4-17 |
| 4.8 | Klasifikasi karakter dengan parameter CellSize = 8, NumBins = 18, dan Sigma = 0.1 | 4-18 |
| 4.9 | Klasifikasi karakter dengan parameter CellSize = 16, NumBins = 9, dan Sigma = 1.0 | 4-19 |
| 4.9 | Klasifikasi karakter dengan parameter CellSize = 16, NumBins = 9, dan Sigma = 1.0 | 4-20 |
| 4.10 | Klasifikasi karakter dengan kombinasi parameter campuran | 4-21 |
| 4.10 | Klasifikasi karakter dengan kombinasi parameter campuran | 4-22 |
| 4.11 | Hasil klasifikasi karakter yang misklasifikasi | 4-26 |

DAFTAR GAMBAR

| | | |
|------|---|------|
| 2.1 | Ilustrasi <i>bin</i> dengan jumlah 4 | 2-7 |
| 2.2 | Contoh <i>Hyperplane</i> pada SVM [12] | 2-9 |
| 2.3 | <i>Confusion Matrix</i> untuk Dua Kelas [13] | 2-12 |
| 2.4 | Contoh dari plat nomor kendaraan Indonesia | 2-20 |
| 2.5 | Ilustrasi ukuran plat nomor untuk kendaraan roda empat Indonesia . | 2-21 |
| 2.6 | Ilustrasi proses pengambilan citra <i>dataset</i> | 2-22 |
| 3.1 | Kerangka Pemikiran | 3-2 |
| 3.2 | <i>Flowchart Global</i> Sistem Pengenalan Plat Nomor Kendaraan . . . | 3-3 |
| 3.3 | Contoh citra plat yang digunakan | 3-6 |
| 3.4 | Contoh citra plat setelah proses <i>preprocessing</i> | 3-6 |
| 3.5 | Citra plat setelah proses segmentasi horizontal | 3-6 |
| 3.6 | Citra plat setelah proses segmentasi vertikal | 3-6 |
| 3.7 | Contoh citra karakter yang digunakan untuk tahap <i>training</i> | 3-7 |
| 3.8 | Skema Alur Pendeteksian Plat | 3-7 |
| 3.9 | Matriks Citra Asal berukuran 5×5 | 3-8 |
| 3.10 | Matriks Citra Hasil <i>Grayscale</i> | 3-9 |
| 3.11 | Contoh Citra hasil deteksi tepi <i>Canny</i> | 3-10 |
| 3.12 | Ilustrasi matriks <i>Accumulator Space</i> | 3-11 |
| 3.13 | Contoh hasil citra plat | 3-12 |
| 3.14 | Contoh hasil keluaran dari tahapan segmentasi | 3-13 |
| 3.15 | Matriks citra hasil <i>preprocessing</i> | 3-14 |
| 3.16 | Matriks hasil Perhitungan Gradien sumbu X | 3-14 |
| 3.17 | Matriks hasil Perhitungan Gradien sumbu Y | 3-14 |
| 3.18 | Matriks hasil Perhitungan <i>Magnitude</i> | 3-15 |
| 3.19 | Matriks hasil Perhitungan Arah | 3-15 |
| 3.20 | Contoh hasil <i>Histogram of Oriented Gradient</i> untuk sel yang memiliki piksel dengan koordinat (2,5) | 3-16 |
| 3.21 | Matriks hasil Perhitungan Histogram untuk seluruh sel | 3-17 |
| 3.22 | Matriks hasil Normalisasi | 3-17 |
| 4.1 | Tampilan antarmuka aplikasi pengenalan plat nomor kendaraan . . | 4-6 |
| 4.2 | Tampilan antarmuka hasil <i>training</i> | 4-7 |
| 4.3 | Keluaran <i>path</i> data untuk <i>testing</i> dan hasil pengenalan plat | 4-8 |
| 4.4 | Keluaran tingkat akurasi pengenalan plat dan <i>Confusion Matrix</i> . . . | 4-8 |
| 4.5 | Keluaran hasil tingkat akurasi pengenalan karakter | 4-9 |

| | | |
|------|--|------|
| 4.6 | Hasil Pengujian Kombinasi Parameter dengan Ukuran Sel 2 | 4-12 |
| 4.7 | Hasil Pengujian Kombinasi Parameter dengan Ukuran Sel 4 | 4-14 |
| 4.8 | Hasil Pengujian Kombinasi Parameter dengan Ukuran Sel 8 | 4-16 |
| 4.9 | Hasil Pengujian Kombinasi Parameter dengan Ukuran Sel 16 | 4-19 |
| 4.10 | Hasil OVR Pengujian Kombinasi Parameter Gabungan | 4-22 |
| 4.11 | Hasil CRR Pengujian Kombinasi Parameter Gabungan | 4-23 |
| 4.12 | Contoh citra (a) Huruf Q (b) Angka 0 | 4-24 |
| 4.13 | Citra Plat asal karakter D yang misklasifikasi | 4-25 |
| 4.14 | Citra hasil segmentasi karakter huruf D | 4-25 |
| 4.15 | Citra Plat asal karakter angka 2 yang misklasifikasi | 4-26 |
| 4.16 | Citra hasil segmentasi karakter misklasifikasi (a) Citra angka 1 dengan satu objek lain (b) Citra angka 2 dengan dua objek lain (c) Citra angka 2 dengan satu objek lain | 4-27 |

BAB I

PENDAHULUAN

1.1 Latar Belakang

Computer Vision 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 (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 pengenalan plat kendaraan dan akurasi pengenalan karakter plat nomor kendaraan. Perhitungan akurasi akan menggunakan *confusion matrix*.

1.2 Rumusan Masalah

Berdasarkan latar belakang di atas rumusan masalah yang didapatkan adalah sebagai berikut:

1. Berapa tingkat akurasi pengenalan plat nomor kendaraan jika menggunakan metode *Histogram of Oriented Gradient* dan *Support Vector Machine* ?
2. Faktor apa saja yang dapat mempengaruhi hasil fitur dari *Histogram of Oriented Gradient* ?

1.3 Tujuan Penelitian

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

1. Menerapkan metode *Histogram of Oriented Gradient* untuk ekstraksi fitur pada

karakter.

2. Menerapkan metode *Support Vector Machine* untuk klasifikasi karakter pada sistem pengenalan plat nomor kendaraan.
3. Menguji *HOG descriptor* dengan beragam ukuran sel dan jumlah *bin*.
4. Menguji akurasi pengenalan karakter pada plat nomor kendaraan dengan metode *Support Vector Machine* dengan beragam nilai sigma.

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. Citra plat kendaraan diambil dalam keadaan lurus dengan kamera. Tidak miring ke kiri dan juga miring ke kanan.
3. Plat kendaraan Indonesia yang akan dideteksi adalah plat nomor kendaraan pribadi (plat hitam dengan tulisan putih) yang berasal dari *dataset Tel-U Vehicle License Plate Data-set V1.0*.

1.5 Kontribusi Penelitian

Kontribusi yang diberikan dari penelitian ini adalah:

1. Membuat penerapan metode *Histogram of Oriented Gradient* untuk proses ekstraksi fitur pada proses pengenalan karakter pada plat nomor kendaraan.
2. Menggabungkan metode *Histogram of Oriented Gradient* dengan *Support Vector Machine* untuk proses klasifikasi karakter.

1.6 Metodologi 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

I. PENDAHULUAN

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 merupakan suatu proses yang berfokus pada dua hal utama, yaitu meningkatkan kualitas informasi pada citra untuk membantu interpretasi manusia dan pemrosesan data citra untuk nantinya disimpan, ditransmisikan, dan digunakan sebagai referensi pengetahuan dari suatu mesin *autonomous* [5].

Terdapat satu paradigma yang mengkategorikan 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 memungkinkan

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 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 [6], dan 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 ini digunakan untuk mendapatkan informasi bentuk dari citra masukan. 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 [5]:

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 suppression* 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. Persamaan 2.2 adalah rumus yang digunakan untuk mencari jarak antara titik *origin* dengan garis yang terbentuk [7]:

$$\rho = x \cos(\theta) + y \sin(\theta) \quad (2.2)$$

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

II. LANDASAN TEORI

adalah antara nilai 0 hingga 180 derajat. Sedangkan rentang nilai ρ (*rho*) yang digunakan dalam akumulator memenuhi aturan pertidaksamaan 2 . 3 [8]:

$$-D \leq \rho \leq D \quad (2 . 3)$$

Dimana:

D = jarak diagonal dari citra

Karena citra yang digunakan berbentuk persegi panjang maka jarak diagonal memenuhi persamaan 2 . 4 [9]:

$$D = \sqrt{N^2 + M^2} \quad (2 . 4)$$

Dimana:

D = jarak diagonal dari citra

N = ukuran *width* dari citra

M = ukuran *height* dari citra

Sehingga rentang nilai *rho* yang digunakan dalam penelitian ini adalah memenuhi aturan seperti ditunjukkan pada pertidaksamaan 2 . 5:

$$-\sqrt{N^2 + M^2} \leq \rho \leq \sqrt{N^2 + M^2} \quad (2 . 5)$$

Dimana:

N = ukuran *width* dari citra

M = ukuran *height* dari citra

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 [10]. Berdasarkan R. Varun, *et al.*, 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.

Algoritme 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. Algoritme pencarian *peak* tersebut dapat menghindari hasil ganda untuk sebuah garis [10].

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 Segmentasi

Segmentasi merupakan proses pembagian daerah dalam suatu citra untuk dipisahkan ke dalam segmen-segmen tertentu. Tujuan utama dari proses segmentasi adalah menyederhanakan dan/atau mengubah representasi citra menjadi sesuatu yang memiliki arti tertentu dan lebih mudah dianalisis [5]. Segmentasi citra secara umum digunakan untuk mengenali objek dan batas-batas (garis) dalam citra.

Secara garis besar, terdapat tiga cara utama untuk melakukan segmentasi, yaitu dengan cara *Thresholding*, segmentasi tepian, dan segmentasi wilayah. Contoh dari metode segmentasi *Thresholding* adalah *Basic Global Thresholding*, *Otsu*, *Multiple Threshold*, dan *Variable Thresholding*. Contoh metode segmentasi tepian adalah deteksi tepi standar (*Sobel*, *Robert*, *Prewitt*, *Canny*), *Marr-Hildreth edge detector*, *Short Response Hilbert Transform*, dan *Watersheds*. Sedangkan untuk metode segmentasi wilayah contohnya adalah *Region Growing*, *Data Clustering*, *Partitional Clustering*, dan *Cheng-Jin Kuo's Method*.

2.1.7 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].

2.1.8 *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 [11]. 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 untuk 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 2 . 6 dan 2 . 7 :

$$G_x(x,y) = I(x+1,y) - I(x-1,y) \quad (2 . 6)$$

$$G_y(x,y) = I(x,y+1) - I(x,y-1) \quad (2 . 7)$$

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 2 . 8 dan 2 . 9:

II. LANDASAN TEORI

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

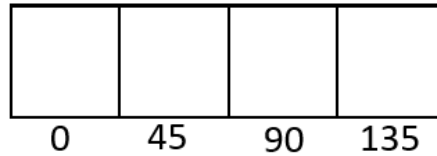
$$\theta(x,y) = \arctan \frac{G_y(x,y)}{G_x(x,y)} \quad (2.9)$$

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. Sebagai contoh, jika jumlah *bin* yang digunakan adalah 4 *bin*. Maka ukuran *range* sudut setiap bin adalah $180/4 = 45$ derajat. Ilustrasinya dapat dilihat pada Gambar 2.1.



Gambar 2.1 Ilustrasi *bin* dengan jumlah 4

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*. Persamaan 2.10 sampai 2.12 berikut merupakan persamaan untuk metode normalisasi *L1-Norm*, *L1-Sqrt*, dan *L2-Norm*:

$$L1\text{-Norm}_i = \frac{V_i}{\sum_{j=1}^N V_j} \quad (2.10)$$

II. LANDASAN TEORI

Dimana:

V = bobot vektor yang merepresentasikan nilai setiap *bin*

j = nilai *counter* dari 1 sampai N

N = jumlah total nilai total *bin* yang digunakan dalam proses normalisasi

$$L1-Sqrt_i = \sqrt{\frac{V_i}{\sum_{j=1}^N V_j}} \quad (2.11)$$

Dimana:

V = bobot vektor yang merepresentasikan nilai setiap *bin*

j = nilai *counter* dari 1 sampai N

N = jumlah total nilai total *bin* yang digunakan dalam proses normalisasi

$$L2-Norm_i = \frac{V_i}{\sqrt{\sum_{j=1}^N V_j^2}} \quad (2.12)$$

Dimana:

V = bobot vektor yang merepresentasikan nilai setiap *bin*

j = nilai *counter* dari 1 sampai N

N = jumlah total nilai total *bin* yang digunakan dalam proses normalisasi

Untuk algoritme rumus normalisasi *L2-Hys* merupakan algoritme 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.

2.1.9 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 masukan ke dalam sebuah ruang beukuran n -dimensional (n adalah jumlah fitur). Konsep dasar dari SVM adalah menemukan sebuah

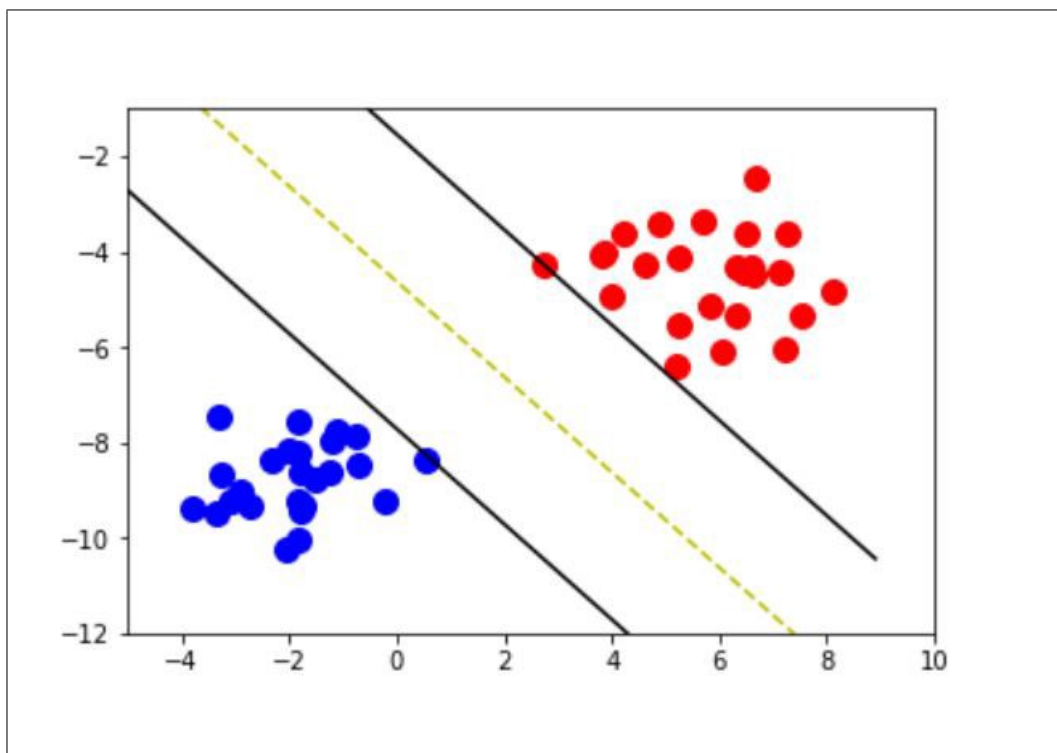
II. LANDASAN TEORI

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 2.2). 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:

$$w^T \cdot x + b = 0 \quad (2 . 13)$$

Dimana w^T adalah vektor berat dan x adalah vektor masukan dan b merupakan nilai bias. Tanda “.” menggambarkan perkalian dot vektor.



Gambar 2.2 Contoh *Hyperplane* pada SVM [12]

SVM pada mulanya digunakan untuk menangani klasifikasi yang terdiri dari 2 kelas saja. Namun seiring 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,

II. LANDASAN TEORI

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 [12].

Untuk proses klasifikasi *non-linear* dapat dicari dengan persamaan 2 . 14 berikut:

$$f(x) = \text{sign}\left(\sum_{i=1}^l \alpha_i y_i K(x, x_i) + b\right) \quad (2 . 14)$$

Dimana:

- $\text{sign}(x)$ = Fungsi *signum*
- 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 2.2. 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:

$$RBF = K(x_i, x_j) = \exp\left(-\frac{\|x_i - x_j\|^2}{2\sigma^2}\right) \quad (2 . 18)$$

Dimana:

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

2.1.10 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 2.3 digunakan pada kasus klasifikasi dua buah kelas sehingga membentuk matriks berukuran 2×2 [13].

II. LANDASAN TEORI

| n=165 | | Predicted: NO | Predicted: YES | |
|----------------|--|------------------|-------------------|-----|
| | | | | |
| Actual: NO | | TN = 50 | FP = 10 | 60 |
| Actual: YES | | FN = 5 | TP = 100 | 105 |
| | | 55 | 110 | |

Gambar 2.3 *Confusion Matrix untuk Dua Kelas* [13]

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 sampel 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.

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

2.1.11 *Character Recognition Rate*

Character Recognition Rate atau CRR adalah alat ukur untuk mengetahui keberhasilan pengenalan karakter dari suatu aplikasi pengenalan karakter [3]. Perhitungan persentase CRR dilakukan dengan cara membagi jumlah karakter yang dapat dikenali terhadap jumlah karakter yang terdeteksi oleh sistem seperti yang dapat dilihat pada persamaan 2 . 22.

$$CRR = \frac{\text{Number of Correctly Recognized Characters}}{\text{Number of All Detected Characters}} \quad (2 . 22)$$

2.1.12 *Overall Performance*

Overall Performance atau OVR adalah alat ukur untuk mengetahui performa keseluruhan dari aplikasi pada penelitian ini [3]. Perhitungan persentase OVR dilakukan dengan cara membagi jumlah plat nomor yang terdeteksi dan dapat dikenali dengan benar terhadap jumlah keseluruhan plat nomor pada *dataset* seperti yang dapat dilihat pada persamaan 2 . 23.

$$OVR = \frac{\text{Number of Correctly Detected and Recognized LPs}}{\text{Number of All Ground Truth LPs}} \quad (2 . 23)$$

2.1.13 *Penggunaan Library*

Bagian ini akan menjelaskan *library* yang digunakan dalam penelitian ini. *Library* yang digunakan diantaranya *OpenCV* untuk melakukan tahapan *preprocessing*, *Weka SVM* untuk melakukan tahapan klasifikasi karakter, *JavaOCR* untuk melakukan segmentasi terhadap citra plat. Untuk penjelasan lebih lanjut dari setiap *library* akan disampaikan pada subbab berikut.

2.1.13.1 *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. |

II. LANDASAN TEORI

Tabel 2.1 Tabel fungsi Library OpenCV (Lanjutan)

| No | Function | Deskripsi |
|----|---|---|
| 2 | Imgproc.cvtColor(Mat src, Mat dst, int code) | Mengkonversi jenis warna citra dengan parameter citra RGB, matriks citra 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> dengan parameter citra <i>grayscale</i> , matriks citra tujuan, 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> dengan parameter citra <i>grayscale</i> , matriks citra tujuan, nilai <i>threshold</i> , nilai maksimum, serta jenis metode <i>thresholding</i> yang digunakan, misalnya metode <i>thresholding</i> Otsu. Keluaran dari <i>function</i> ini adalah citra biner. |
| 6 | Imgproc.findContours(Mat image, List<MatOfPoint> contours, Mat hierarchy, int mode, int method) | Melakukan pencarian kontur terhadap citra biner yang dijadikan masukan. |
| 7 | Imgproc.contourArea(Mat contour) | Melakukan perhitungan luas area untuk setiap kontur dari hasil pencarian kontur menggunakan <i>function</i> <i>Imgproc.findContours()</i> . |
| 8 | Imgproc.imwrite(String filename, Mat img) | Menyimpan citra yang diisikan ke parameter ke <i>path</i> yang dijadikan tujuan penyimpanan. |
| 9 | Imgproc.HoughLines(Mat image, double minTheta, double maxTheta, double rho, Mat peak) | Mengekstraksi segmen garis berdasarkan <i>peaks</i> . <i>Peaks</i> adalah matriks hasil keluaran <i>function</i> <i>houghpeaks</i> yang berisi koordinat baris dan kolom dari <i>hough transform bins</i> yang digunakan untuk melakukan pencarian segmen garis. |

II. LANDASAN TEORI

2.1.13.2 Weka

Weka adalah kumpulan dari algoritme pembelajaran mesin yang digunakan untuk menyelesaikan permasalahan *data mining*. *Library* ini berbasis bahasa pemrograman Java dan dapat berjalan di hampir seluruh *platform*. Dalam penelitian ini, *library* Weka digunakan untuk melakukan proses klasifikasi menggunakan metode *Support Vector Machine*.

Tabel 2.2 Tabel fungsi *Library* Weka

| No | Function | Deskripsi |
|----|--|---|
| 1 | libsvm.setOptions(String[] options) | Mengkonfigurasi dan menginisialisasi parameter dari <i>classifier</i> yang akan digunakan. |
| 2 | libsvm.buildClassifier(Instances insts) | Membangun <i>classifier</i> dari dataset yang diberikan sesuai dengan konfigurasi dan parameter yang digunakan. |
| 3 | libsvm.classifyInstance(Instance instance) | Melakukan prediksi kelas untuk data yang diberikan. |

2.1.13.3 JavaOCR

JavaOCR adalah *library* berbasis Java untuk *Image Processing* dan *Character Recognition*. *Library* ini memiliki *GUI* sendiri dan dapat digunakan untuk pengembangan aplikasi *Character Recognition* pada perangkat Android dikarenakan *library* ini hanya membutuhkan *memory* yang sedikit dan tidak memiliki ketergantungan terhadap *library* lainnya.

Tabel 2.3 Tabel fungsi *Library* JavaOCR

| No | Function | Deskripsi |
|----|--|--|
| 1 | LineExtractor.slice(File inputImage, File outputDir) | Melakukan pencarian batas atas dan batas bawah area karakter dari citra masukan dan melakukan <i>cropping</i> terhadap citra tersebut dan kemudian hasil keluarannya akan berupa citra baru yang merupakan area karakter dari citra masukan, apabila terdapat lebih dari satu kandidat area karakter, maka hasilnya akan menjadi beberapa citra area karakter yang terpisah. |

II. LANDASAN TEORI

Tabel 2.3 Tabel fungsi Library JavaOCR (Lanjutan)

| No | Function | Deskripsi |
|----|--|---|
| 2 | CharacterExtractor.slice(File inputImage, File outputDir, int width, int height) | Melakukan pencarian batas kiri dan batas kanan karakter dari citra masukan. Citra masukan berupa citra hasil keluaran dari <i>function</i> <i>LineExtractor.slice()</i> . Keluaran dari <i>function</i> ini akan berupa citra karakter yang didapatkan dari citra masukan, apabila terdapat lebih dari satu karakter, maka hasilnya akan menjadi beberapa citra karakter yang terpisah. |

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 2.4 *State of the Art* akan menjelaskan perbedaan-perbedaan metode dari jurnal terkait.

Tabel 2.4 State of the Art

| Jurnal | Rumusan Masalah | Metode |
|--|--|---|
| Nugroho, A., Wardhani, K.R.R. (2011). Aplikasi Sistem Pembaca Plat Nomor Mobil Menggunakan Pengolahan Citra dan Metode Learning Vector Quantization. | Apakah dengan menerapkan pengolahan citra untuk deteksi plat kendaraan dan metode <i>Learning Vector Quantization</i> untuk pengenalan karakter dapat menghasilkan akurasi yang baik ? | 1. <i>Pengolahan Citra</i> 2. <i>Learning Vector Machine</i> |

Tabel 2.4 *State of the Art* (Lanjutan)

| 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. <i>IEEE Transactions on Intelligent Transportation Systems</i> , 17(4), 1096-1107. | Apakah dengan menerapkan pendeteksian plat nomor dengan <i>Extremal Region</i> dan pengenalan karakter plat nomor menggunakan <i>Hybrid Discriminative Restricted Boltzmann Machine</i> dapat meningkatkan akurasi pendeteksian dan pengenalan dalam berbagai kondisi cuaca dan <i>background</i> yang kompleks? | <ol style="list-style-type: none"> 1. <i>Extremal Region</i> 2. <i>AdaBoost</i> 3. <i>Histogram of Oriented Gradient</i> 4. <i>Hybrid Discriminative Restricted Boltzmann Machine</i> |
| Gou, C., Wang, K., Yu, Z., Xie, H. (2014, October). License plate recognition using MSER and HOG based on ELM. In <i>Proceedings of 2014 IEEE International Conference on Service Operations and Logistics, and Informatics</i> (pp. 217-221). IEEE. | Apakah dengan menerapkan <i>Maximally Stable Extremal Region</i> untuk pendeteksian plat nomor dan <i>Extreme Learning Machine</i> untuk pengenalan karakter plat nomor dapat meningkatkan performa dan akurasi sistem? | <ol style="list-style-type: none"> 1. <i>Maximally Stable Extremal Region</i> 2. <i>Histogram of Oriented Gradient</i> 3. <i>Extreme Learning Machine</i> |

Tabel 2.4 *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. <i>Procedia Computer Science</i> , 102, pp. 588-594. | Apakah dengan menggabungkan metode klasifikasi <i>K-Nearest Neighbours</i> dan <i>Support Vector Machine</i> akan menghasilkan akurasi yang lebih baik dan mengurangi <i>cost</i> pada proses pengenalan karakter plat nomor kendaraan? | <ol style="list-style-type: none"> 1. <i>Structural Feature</i> 2. <i>Horizontal and Vertical Crossing Count Histogram</i> 3. <i>Zoning Feature Extraction</i> 4. <i>K-Nearest Neighbours</i> 5. <i>Support Vector Machine</i> |
| Rasheed, S., Naeem, A., Ishaq, O. (2012, October). Automated number plate recognition using hough lines and template matching. In <i>Proceedings of the World Congress on Engineering and Computer Science</i> (Vol. 1, pp. 24-26). | Apakah dengan menggunakan metode <i>Hough Transform</i> untuk mendeteksi plat kendaraan dan <i>Template Matching</i> untuk pengenalan karakter dapat menghasilkan akurasi yang baik untuk sistem pengenalan plat nomor kendaraan? | <ol style="list-style-type: none"> 1. <i>Canny Detector</i> 2. <i>Hough Transform</i> 3. <i>Morphological Process</i> 4. <i>Template Matching</i> |

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 [14] menggunakan metode pengolahan citra untuk mencari kandidat-kandidat pita dengan menghitung histogram citra untuk mendapatkan lokasi plat nomor kendaraan, kemudian setiap karakter dari plat nomor yang didapatkan disegmentasi dengan cara menghitung grafik horizontal citra, kemudian untuk metode klasifikasi karakter yang digunakan adalah *Learning Vector Quantization*.

Pada referensi kedua [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 masukan bagi metode klasifikasi *Hybrid Discriminative Restricted Boltzmann Machine*. Penelitian ini menghasilkan akurasi 98,2% untuk pengenalan karakternya.

Pada referensi ketiga [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*. Penelitian ini menghasilkan akurasi 97,90 % untuk pengenalan karakternya.

Pada referensi keempat [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. Penelitian ini menghasilkan akurasi 97,03% untuk pengenalan karakternya.

Pada referensi kelima [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%.

2.3 Tinjauan Objek

Pada bagian ini akan diulas mengenai objek penelitian yaitu plat nomor kendaraan dan hal-hal yang berkaitan seperti spesifikasi plat nomor kendaraan Indonesia, jenis plat nomor kendaraan bermotor di Indonesia, dan *dataset Tel-U Vehicle Data-set V1.0* yang akan digunakan dalam penelitian.

2.3.1 Tanda Nomor Kendaraan Bermotor

Tanda Nomor Kendaraan Bermotor (disingkat TNKB) disebut juga sebagai plat nomor atau nomor polisi adalah plat aluminium tanda kendaraan bermotor di Indonesia yang telah didaftarkan pada Kantor Bersama Samsat.



Gambar 2.4 Contoh dari plat nomor kendaraan Indonesia

Plat nomor kendaraan Indonesia terdiri dari cetakan tulisan dua baris, pada baris pertama terdapat tiga hal yang diinformasikan, yaitu kode wilayah(huruf), nomor polisi(angka), dan kode/seri akhir wilayah(huruf). Sedangkan baris kedua menunjukkan bulan dan tahun masa berlaku, masing-masing dua digit. Contohnya pada Gambar 2.4 angka yang terdapat di baris keduanya adalah 10.21. Hal tersebut menandakan bahwa plat nomor tersebut akan habis masa berlakunya pada bulan Oktober tahun 2021.

Bahan baku dari plat nomor adalah aluminium dengan ketebalan 1mm. Ukuran plat nomor untuk kendaraan bermotor roda dua dan roda tiga adalah 250×105 mm, sedangkan untuk kendaraan bermotor roda empat atau lebih adalah 395×135 mm. Terdapat cetakan garis lurus pembatas selebar 5mm di antara ruang nomor polisi dengan ruang angka masa berlaku (untuk plat nomor lama), sedangkan semenjak tahun 2011 di sekitar plat nomor terdapat garis putih dan tidak ada garis pemisah antara nomor polisi dan masa berlaku.



Gambar 2.5 Ilustrasi ukuran plat nomor untuk kendaraan roda empat Indonesia

Berdasarkan Gambar 2.5. Dengan ukuran panjang sebesar 395mm dan ukuran lebar sebesar 135mm. Maka dapat disimpulkan bahwa rasio untuk ukuran dari plat nomor Indonesia adalah 3:1 (panjang:lebar). Informasi dapat digunakan untuk proses pendeteksian plat kendaraan.

Pada tahun 2014 terjadi perubahan tampilan pada plat nomor untuk kendaraan bermotor roda empat. Plat nomor kini sedikit lebih panjang dari sebelumnya (5 cm lebih panjang) untuk memberi ruang pada kode/seri akhir wilayah yang dulunya berjumlah dua digit menjadi tiga digit.

2.3.2 Jenis TNKB

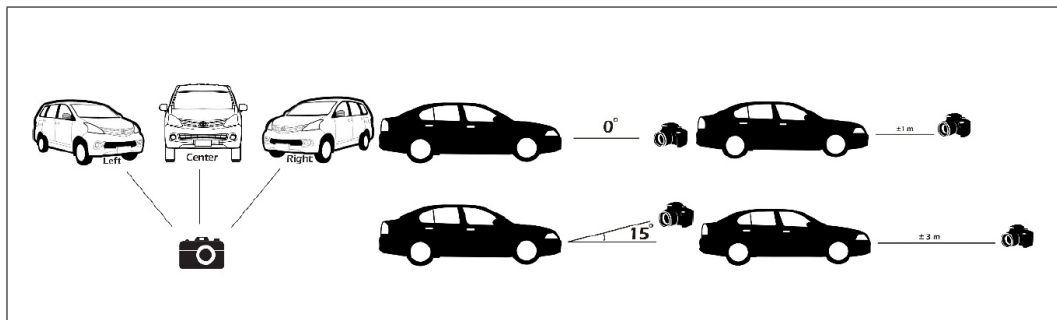
Jenis dari TNKB di Indonesia dibedakan berdasarkan warna dari TNKB tersebut, warna TNKB yang ditetapkan di Indonesia adalah sebagai berikut [15]:

1. Kendaraan pribadi dan sewa: warna dasar hitam dengan tulisan berwarna putih.
2. Kendaraan umum: warna dasar kuning dengan tulisan hitam.
3. Kendaraan milik pemerintah: warna dasar merah dengan tulisan berwarna putih.
4. Kendaraan bermotor sementara: warna dasar putih dengan tulisan berwarna merah.
5. Kendaraan korps diplomatik negara asing: warna dasar putih/merah dengan tulisan berwarna hitam.
6. Kendaraan staf operasional korps diplomatik negara asing: warna dasar hitam dengan tulisan berwarna putih serta terdiri dari lima angka dan kode angka negara yang dicetak lebih kecil dengan format sub-bagian.

2.3.3 *Dataset Tel-U Vehicle Data-set V1.0*

Dataset Tel-U Vehicle Data-set V1.0 merupakan *dataset* untuk penelitian pendeteksian plat nomor kendaraan yang berasal dari Universitas Telkom. Universitas Telkom menyediakan *dataset* ini secara gratis dan dapat digunakan untuk umum.

Dalam *dataset* ini terdapat 228 citra kendaraan dengan plat nomor kendaraan yang diambil dari bagian depan kendaraan. Pengambilan citra dilakukan dari beragam posisi, sudut, jarak, dan pencahayaan.



Gambar 2.6 Ilustrasi proses pengambilan citra *dataset*

Citra dalam *dataset* ini diambil dengan menggunakan kamera DSLR Canon EOS 500 D dan Canon EOS 550 D dengan resolusi 15 dan 18 megapiksel. Citra kemudian diubah resolusinya menjadi 1024×640 piksel. Jenis plat nomor kendaraan yang ada di dalam *dataset* ini adalah plat kendaraan bermotor pribadi (plat hitam dengan karakter putih). Untuk penelitian ini, citra plat kendaraan yang akan digunakan adalah citra plat kendaraan bermotor yang diambil dengan posisi lurus dengan kendaraan dalam jarak kurang lebih satu meter dan kurang lebih 3 meter.

BAB III

ANALISIS DAN PERANCANGAN SISTEM

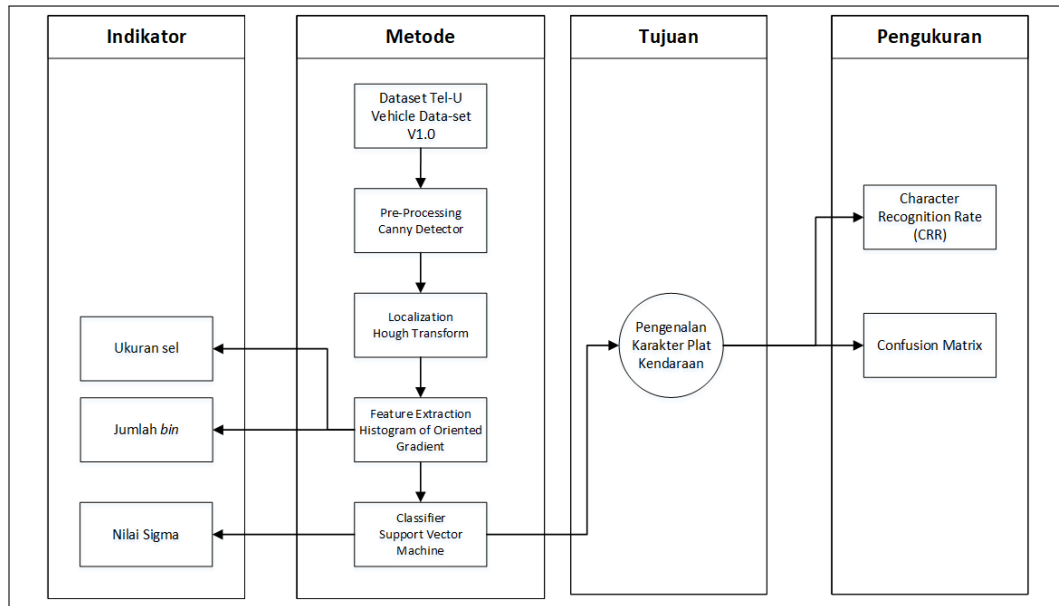
Bab ini memaparkan analisis masalah yang diatasi beserta pendekatan dan alur kerja dari perangkat lunak yang dikembangkan, mengimplementasikan metode yang digunakan dan hasil yang akan ditampilkan.

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, metode yang akan digunakan adalah *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 masukan.

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.

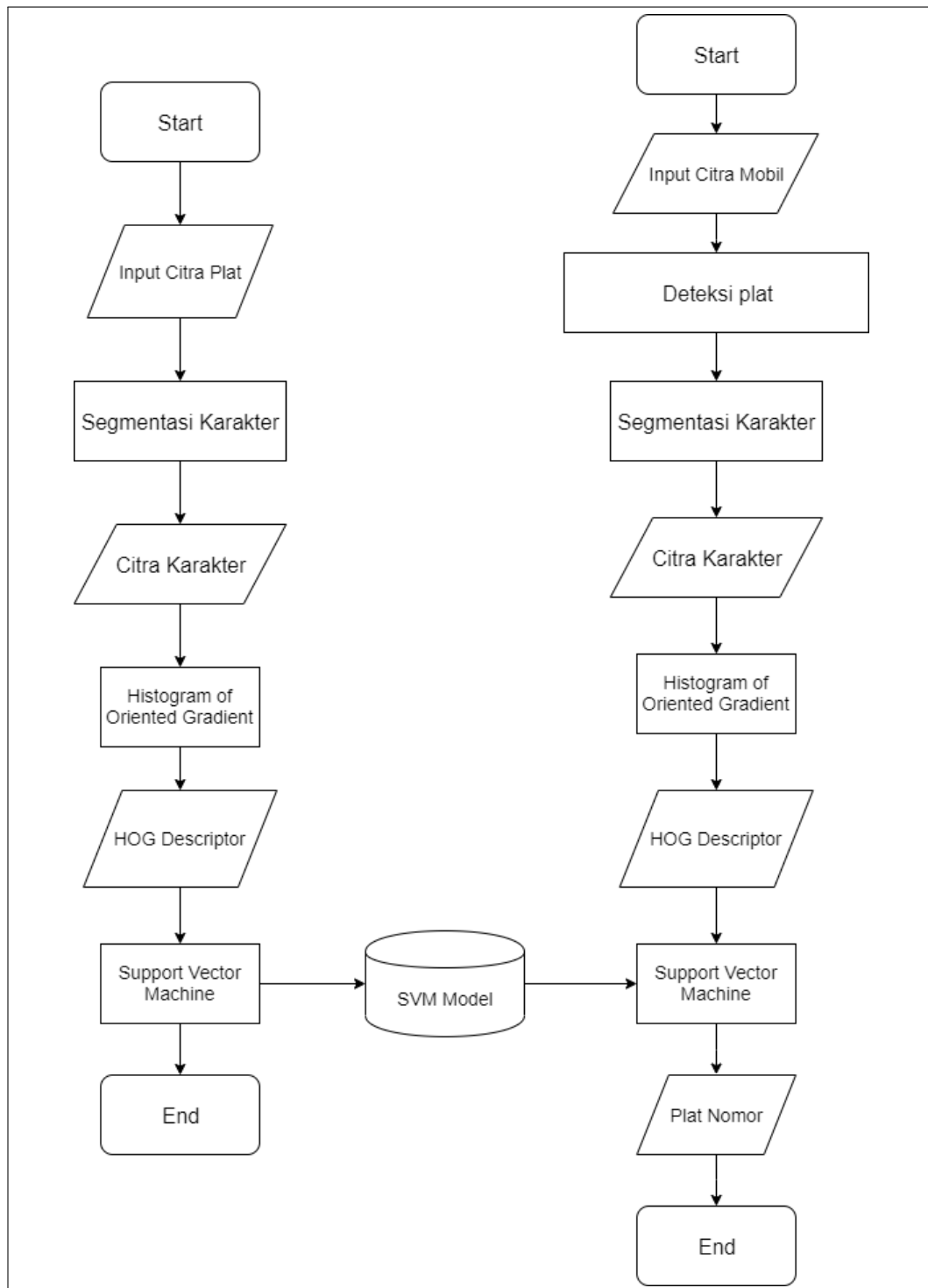


Gambar 3.1 Kerangka Pemikiran

Seperti pada Gambar 3.1, terdapat beberapa variabel indikator yang mempengaruhi hasil dan perlu dilakukan penyesuaian, seperti ukuran sel pada metode *Histogram of Oriented Gradient*, jumlah *bin* yang menentukan batasan sudut yang digunakan, dan nilai sigma untuk *classifier Support Vector Machine*. Penelitian ini memiliki tujuan untuk menerapkan *Histogram of Oriented Gradient* dan *Support Vector Machine* untuk sistem pengenalan karakter pada plat nomor dengan menguji beragam faktor yang diduga akan mempengaruhi hasil akurasi dari penggabungan kedua metode tersebut. Hasil pengenalan karakter akan diukur 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.



Gambar 3.2 Flowchart Global Sistem Pengenalan Plat Nomor Kendaraan

Seperti pada Gambar 3.2. Bagian kiri merupakan *flowchart* dari proses *training* dan bagian kanan merupakan *flowchart* dari proses *testing*. Berikut ini adalah uraian dari proses-proses yang terjadi ketika tahapan *training*:

1. Citra masukan adalah citra plat nomor kendaraan mobil yang di-*crop* secara manual dari citra mobil utuh hal ini untuk memastikan citra yang didapatkan

adalah plat yang benar.

2. Citra plat kemudian akan melalui tahapan *preprocessing* yang meliputi *grayscale* untuk menghilangkan informasi warna yang tidak diperlukan, *Gaussian Smoothing* untuk menghilangkan derau pada citra, *Binarization* untuk mengubah citra menjadi citra biner, menghilangkan objek kecil (untuk menghilangkan objek seperti baut pada plat) dengan cara menghilangkan objek yang luasnya kurang dari *threshold* sebesar 100 piksel, dan terakhir *Inverse Binarization* untuk mendapatkan citra karakter berwarna hitam dengan latar belakang berwarna putih.
3. Citra plat selanjutnya melalui tahapan segmentasi karakter, tahapan segmentasi bertujuan untuk mendapatkan citra-citra karakter dari plat nomor tersebut tahapan segmentasi dilakukan dua tahapan, yaitu segmentasi horizontal dan segmentasi vertikal. Tahapan segmentasi dilakukan dengan menggunakan *library* segmentasi dari *JavaOCR*. Citra hasil segmentasi memiliki ukuran yang beragam, tetapi setiap citra diberi *padding* sebesar 1 piksel dari tepian. Tujuannya agar bentuk objek bisa didapat dengan baik, hal ini akan membuat fitur yang didapat pada tahapan ekstraksi fitur menjadi lebih baik.
4. Citra karakter hasil *segmentasi* akan di-*scaling* menjadi berukuran 32×32 piksel dengan tujuan untuk menyeragamkan ukuran citra karakter. Kumpulan karakter yang digunakan adalah karakter angka dari 0 sampai dengan 9 dan karakter huruf dari A sampai dengan Z, tidak ada karakter huruf kecil dikarenakan plat nomor kendaraan tidak ada yang menggunakan karakter huruf kecil.
5. *Histogram of Oriented Gradient* berfungsi untuk mendapatkan fitur dari citra hasil segmentasi. Hasil dari ekstraksi fitur dengan menggunakan HOG adalah *HOG descriptor*, yang mendeskripsikan distribusi dari gradien berarah pada suatu area citra.
6. Untuk ukuran sel dan jumlah *bin* yang digunakan untuk proses ekstraksi fitur dengan menggunakan *HOG*, dikarenakan kedua parameter tersebut merupakan indikator uji seperti ditunjukkan pada kerangka pemikiran 3.1, maka nilai dari ukuran sel dan jumlah *bin* akan menyesuaikan dengan kondisi yang akan diuji.
7. Setelah tahapan ekstraksi fitur, fitur-fitur akan disimpan dalam berkas CSV

dan proses pelabelan fitur untuk setiap karakter akan dilakukan terhadap berkas CSV tersebut.

8. *Support Vector Machine* (SVM) digunakan untuk mengklasifikasikan fitur-fitur yang sudah didapatkan ke dalam kelas-kelas dari karakter yang akan dikenali. Metode *SVM* yang digunakan pada penelitian ini berasal dari *library WEKA*.

Kemudian untuk proses *testing*, seperti terlihat pada Gambar 3.2, terdapat beberapa proses yang sama seperti pada proses *training*. Berikut ini adalah uraian dari *flowchart* bagian kanan pada Gambar 3.2 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 kendaraan akan melalui tahapan deteksi area plat kendaraan untuk mendapatkan citra plat.
3. Citra plat yang didapatkan kemudian disegmentasi untuk mendapatkan citra karakter.
4. Citra yang akan menjadi masukan dari *HOG* adalah citra hasil dari segmentasi karakter pada citra plat kendaraan hasil deteksi lokasi plat nomor kendaraan.
5. Ukuran dari sel dan blok yang digunakan untuk proses ekstraksi fitur dengan menggunakan *HOG* akan beragam sesuai dengan pengujian yang akan dilakukan.
6. Pada tahap *testing* model SVM yang digunakan berasal dari hasil keluaran model SVM pada tahap *training*.
7. 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 pendeteksian dan pengenalan plat nomor dimulai dari tahapan mempersiapkan *dataset*, mendeteksi lokasi plat nomor, melakukan segmentasi karakter, melakukan ekstraksi fitur dengan *Histogram of Oriented Gradient*, dan melakukan proses latih dengan menggunakan *Support Vector Machine*.

3.4.1 Dataset

Citra plat kendaraan yang digunakan sebagai dataset adalah citra plat yang di-*crop* secara manual seperti terlihat pada Gambar 3.3.



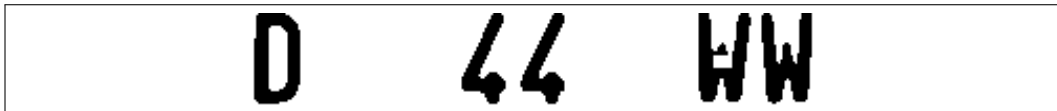
Gambar 3.3 Contoh citra plat yang digunakan

Dari citra plat akan dilakukan tahapan *preprocessing*. Hasil akhir dari tahapan-tahapan tersebut dapat dilihat pada Gambar 3.4.



Gambar 3.4 Contoh citra plat setelah proses *preprocessing*

Dari citra plat hasil *preprocessing*, berikutnya dilakukan proses segmentasi terhadap citra plat. Segmentasi horizontal bertujuan untuk memisahkan area karakter plat nomor dengan karakter tanggal masa berlaku plat nomor. Sedangkan segmentasi vertikal bertujuan untuk mendapatkan karakter karakter. Hasilnya dapat dilihat pada Gambar 3.5 dan Gambar 3.6.



Gambar 3.5 Citra plat setelah proses segmentasi horizontal



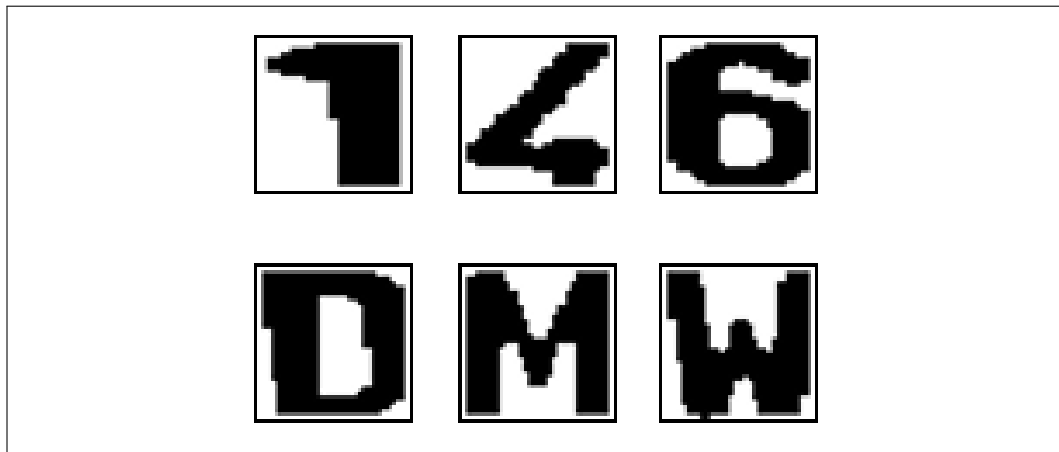
Gambar 3.6 Citra plat setelah proses segmentasi vertikal

Setelah didapatkan karakter-karakter dari plat nomor tersebut, berikutnya dilakukan proses *scaling* dari ukuran seperti pada Gambar 3.6 menjadi ukuran 32×32 piksel. Alasan digunakannya ukuran 32×32 piksel adalah agar ukuran fitur dari *HOG Descriptor* yang dihasilkan tidak terlalu besar. Hasil dari proses *scaling* citra karakter dapat dilihat pada Gambar 3.7. Citra karakter hasil proses *scaling* itulah yang akan digunakan sebagai data latih.

Citra sampling memuat karakter terdiri dari angka 0 sampai dengan 9 dan karakter

III. ANALISIS DAN PERANCANGAN SISTEM

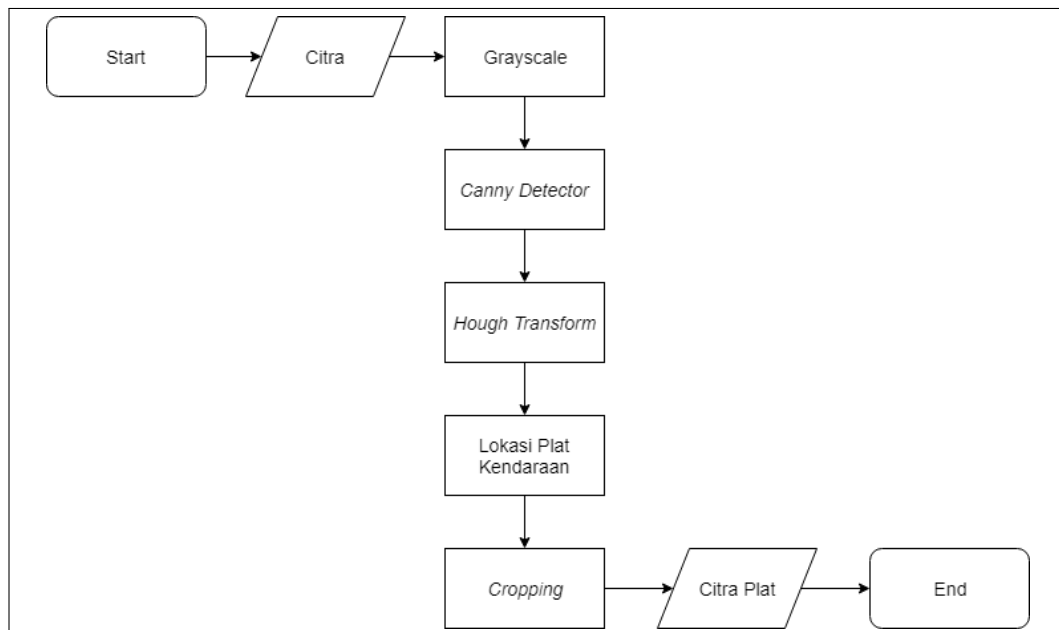
huruf kapital dari A sampai dengan Z. Untuk setiap karakter akan digunakan citra latih sebanyak tiga sampai dengan enam citra, hal ini untuk mengakomodasi beragam kondisi citra karakter sesuai dengan kondisi citra karakter pada plat mobil. Perbedaan dari karakter yang sama bisa disebabkan karena kondisi pencahayaan ketika pengambilan citra, modifikasi pada plat kendaraan misalnya menggunakan *casing* padaudukan plat, dan posisi kamera ketika pengambilan citra. Terlihat contoh citra karakter yang ditunjukkan pada Gambar 3.7 merupakan contoh karakter angka dan huruf kapital yang akan dipakai.



Gambar 3.7 Contoh citra karakter yang digunakan untuk tahap *training*

3.4.2 Tahap Pendeteksian Lokasi Plat Nomor

Skema alur dari tahap pendeteksian lokasi plat nomor adalah:



Gambar 3.8 Skema Alur Pendeteksian Plat

3.4.2.1 Grayscale

Proses pertama adalah mengubah citra masukan dari citra RGB menjadi citra *grayscale*, tujuan dari *grayscale* citra adalah untuk menghilangkan informasi warna dari setiap piksel citra. Untuk menghitung nilai derajat keabuan setiap piksel, diperoleh dengan menggunakan persamaan 2.1. Di bawah merupakan contoh matriks citra asli dengan 3 *channel* warna yaitu *Red*, *Green*, dan *Blue* berukuran 5×5 piksel.

| | | | | |
|-------------------------|-------------------------|----------------------------|----------------------------|----------------------------|
| 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: 26 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 5×5

Dengan menggunakan persamaan 2.1, maka nilai matriks citra *grayscale* pada titik (4,4) akan menjadi sebagai berikut:

$$\begin{aligned}
 \text{Matriks}[4,4] &= (0.299 * 133) + (0.587 * 138) + (0.114 * 144) \\
 &= 137.189 \approx 137
 \end{aligned}$$

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

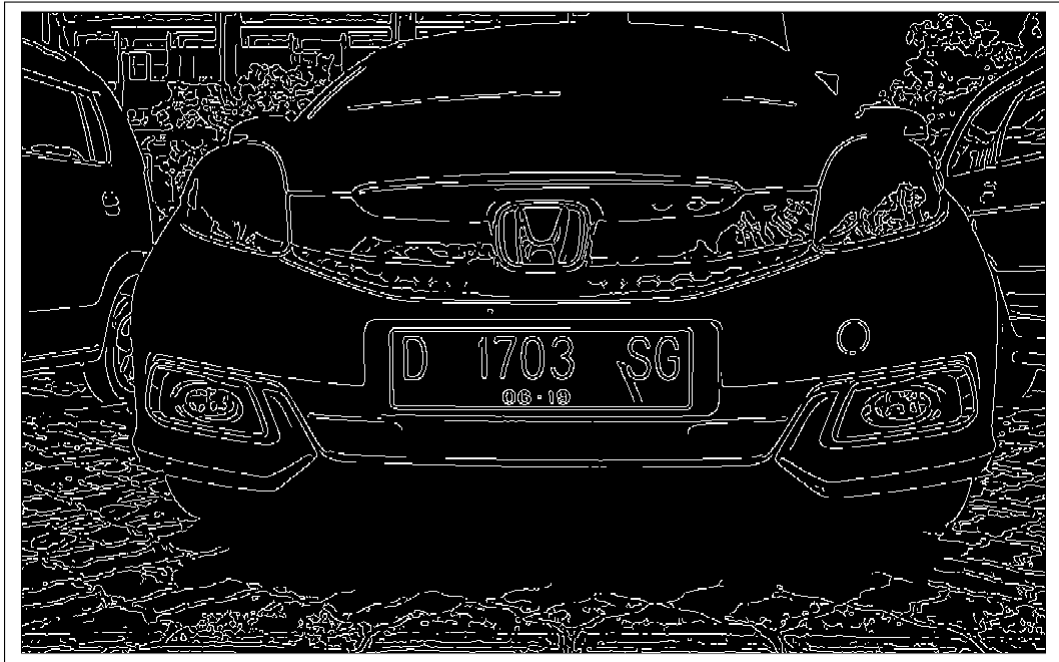
| | | | | |
|----|----|-----|-----|-----|
| 25 | 27 | 33 | 35 | 38 |
| 28 | 32 | 36 | 27 | 27 |
| 29 | 32 | 102 | 130 | 108 |
| 27 | 59 | 107 | 137 | 162 |
| 28 | 42 | 123 | 112 | 151 |

Gambar 3.10 Matriks Citra Hasil *Grayscale*

3.4.2.2 Deteksi Tepi Canny

Proses deteksi tepi dilakukan terhadap citra hasil *grayscale*. Pada penelitian ini, metode *Canny Edge Detection* digunakan untuk mendapatkan tepian. Berikut adalah algoritme dari metode *Canny Edge Detection* untuk mendapatkan tepian.

1. Citra masukan adalah citra dari hasil *grayscale* pada tahapan sebelumnya.
2. Citra masukan diperhalus dengan menggunakan *Gaussian Filter* untuk membuang derau.
3. Lakukan operasi perhitungan gradien menggunakan operator Sobel untuk mendapatkan tepian yang tebal.
4. Untuk menipiskan tepian yang didapat dari operasi sebelumnya maka teknik *Non-Maxima Suppression* dilakukan dengan mencari nilai maksimum pada tepian.
5. Buat 2 nilai *threshold* yaitu *high threshold* dan *low threshold* untuk menentukan piksel mana yang masuk dalam kategori tepian kuat, tepian lemah, dan bukan tepian. Jika nilai dari piksel tersebut di atas *high threshold*, maka piksel tersebut masuk ke dalam kategori tepian kuat, apabila nilai piksel berada di antara batas *high threshold* dan *low threshold*, maka piksel tersebut masuk ke dalam kategori tepian lemah, selebihnya akan masuk ke dalam kategori bukan tepian.
6. Tahapan terakhir adalah *Edge Linking* untuk menghubungkan tepian lemah dengan tepian kuat. Apabila piksel tepian lemah memiliki tetangga piksel (terhubung), maka piksel tersebut akan menjadi tepian.
7. Citra keluaran adalah citra biner yang merupakan hasil pendeteksian tepi.

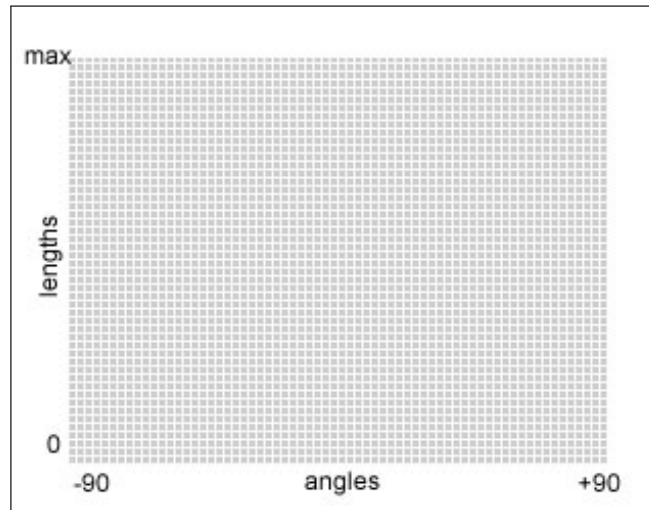


Gambar 3.11 Contoh Citra hasil deteksi tepi *Canny*

3.4.2.3 Hough Transform

Metode *Hough Transform* yang digunakan adalah untuk identifikasi garis lurus. Dalam ekstraksi fitur *Hough Transform* perlu menspesifikasikan *accumulator space* untuk menyimpan nilai *voting*. Untuk tahapan *Hough Transform* ini akan menggunakan *library* dari *OpenCV* yaitu dengan menggunakan *function* *Imgproc.HoughLines()*. *Function* *Imgproc.HoughLines()* ini memiliki parameter masukan berupa citra biner hasil dari metode *Canny Edge Detection*, *range* sudut yang akan digunakan sebagai θ untuk membatasi sudut yang akan dicari, dan nilai ρ yang merupakan panjang garis dalam piksel. Spesifikasi *accumulator space* ditentukan berdasarkan ukuran citra masukan. Berikut adalah algoritme dari metode *Hough Transform* untuk mendapatkan garis lurus:

1. Citra masukan adalah citra biner hasil deteksi tepi pada tahapan sebelumnya.
2. Matriks *Accumulator Space* didefinisikan sebagai *array* 2 dimensi dengan sumbu horizontal menunjukkan nilai sudut (θ) yang digunakan dan sumbu vertikal adalah nilai-nilai dari ρ .



Gambar 3.12 Ilustrasi matriks *Accumulator Space*

3. Untuk setiap nilai piksel dari citra biner hasil deteksi tepian, apabila nilai piksel tersebut adalah 0, piksel tersebut diabaikan. Jika nilai piksel tersebut tidak 0, maka lakukan perhitungan nilai ρ untuk piksel tersebut dengan menggunakan persamaan 2.2 dan lakukan *voting* terhadap setiap nilai θ yang digunakan dengan menambahkan nilai pada matriks akumulator dengan koordinat (θ, ρ) sebesar satu.
4. Hasil dari perhitungan *voting* akan dicari hasil-hasil *voting* tertinggi untuk dijadikan kandidat garis melalui tahapan pencarian *Hough Peaks*. Tahapan ini dilakukan dengan menentukan nilai *threshold*, *neighbourhood*, dan jumlah *peaks* yang akan diambil.
5. Hasil pencarian *Hough Peaks* akan menghasilkan kumpulan nilai ρ dan θ , nilai ini kemudian diubah menjadi koordinat titik.
6. Keluaran dari tahapan ini adalah *array* yang berisi pasangan koordinat titik dari kandidat-kandidat garis yang didapat.

3.4.2.4 Tahap Validasi Plat Kendaraan

Tahap selanjutnya setelah mendapatkan kandidat-kandidat garis adalah menyeleksi area plat nomor. Hal ini dilakukan melalui serangkaian tahapan sebagai berikut:

1. Dari keseluruhan kandidat garis yang didapat, pisahkan kandidat garis vertikal dengan kandidat garis horizontal.
2. Dari kandidat-kandidat garis vertikal akan ada yang menjadi batas kiri dan batas kanan dari area plat kendaraan. Setiap kandidat garis vertikal akan dipasangkan dengan garis vertikal lain dengan cara membandingkan mana

garis yang lebih kanan. Hasilnya akan disimpan dalam *array* yang berisi koordinat x dari masing-masing pasangan garis.

3. Hitung lebar citra yang dibatasi dengan pasangan garis vertikal yang didapatkan, apabila lebar citra sesuai batasan ukuran yang ditentukan, maka pasangan garis tersebut akan menjadi kandidat dari batas kiri dan batas kanan dari plat nomor.
4. Untuk setiap kandidat batas kiri dan batas kanan plat nomor, pasangkan dengan kandidat garis horizontal dan hitung tinggi citra yang dibatasi dengan batas horizontal, apabila tinggi citra sesuai dengan batasan ukuran yang ditentukan, maka pasangan garis tersebut akan menjadi batas atas dan batas bawah dari citra plat kendaraan.
5. Jika masih terdapat lebih dari satu kandidat, maka pilih kandidat dengan rasio panjang : lebar yang paling mendekati rasio plat nomor kendaraan Indonesia, yaitu 1 : 3.
6. Hasil dari tahapan ini adalah 4 titik koordinat yang merupakan koordinat citra plat. Citra plat akan diambil dari citra asal dengan menggunakan titik-titik koordinat tersebut.



Gambar 3.13 Contoh hasil citra plat

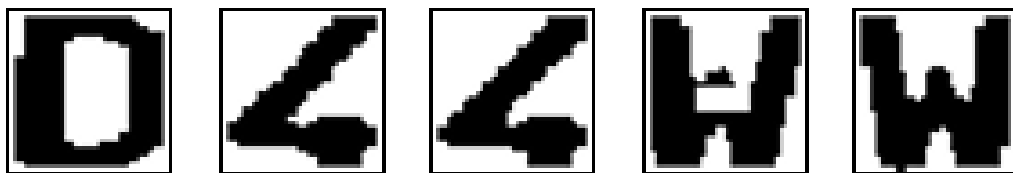
3.4.3 Tahapan Segmentasi Karakter

Setelah mendapatkan kandidat plat, maka berikutnya dilakukan segmentasi karakter untuk mendapatkan citra-citra karakter yang terdapat pada plat nomor kendaraan. Pada tahapan segmentasi karakter, akan dilakukan dua tahapan, yaitu segmentasi vertikal untuk mendapatkan batas atas dan batas bawah daerah karakter, dan segmentasi horizontal untuk mendapatkan batas kiri dan batas kanan untuk setiap karakter. Untuk tahapan segmentasi karakter ini akan menggunakan *method* dari *library JavaOCR* seperti yang disebutkan pada 2.3, yaitu *LineExtractor.slice()* dan *CharacterExtractor.slice()*. *LineExtractor.slice()* digunakan untuk melakukan segmentasi horizontal, parameter dari *method* tersebut adalah berkas citra plat yang sudah melalui tahapan *preprocessing* dan berkas citra yang akan menampung hasil segmentasi horizontal. Sedangkan *CharacterExtractor.slice()* digunakan untuk melakukan segmentasi vertikal dengan

parameter citra hasil segmentasi horizontal, berkas citra yang akan menampung hasil segmentasi vertikal, dan dua parameter berikutnya adalah ukuran lebar dan tinggi citra untuk hasil proses *scaling*. Berikut adalah langkah-langkah dari proses segmentasi karakter:

1. Citra masukan adalah citra plat hasil tahapan deteksi plat kendaraan yang sudah dilakukan *preprocessing* menjadi citra biner.
2. Lakukan segmentasi vertikal untuk mendapatkan batas atas dan batas bawah dari area kandidat karakter.
3. Lakukan segmentasi horizontal untuk mendapatkan batas kiri dan batas kanan dari setiap citra karakter.
4. Setiap citra karakter yang didapatkan akan di-*scaling* menjadi ukuran 32×32 piksel. Hal ini bertujuan untuk menjaga konsistensi ukuran citra karakter yang digunakan untuk proses *training* dan proses *testing*. Hasil dari segmentasi seperti yang ditunjukkan pada Gambar 3.14 ditambahkan satu piksel lebih untuk setiap batas kiri, kanan, atas, dan bawah dari citra, tujuannya adalah agar bentuk karakter yang didapatkan ketika proses ekstraksi fitur dengan menggunakan metode *Histogram of Oriented Gradient* menjadi lebih baik.
5. Keluaran dari tahapan ini adalah citra karakter yang terdapat pada plat nomor.

Pada penelitian ini, tahapan segmentasi karakter akan menggunakan *library* dari Java OCR dan *method* atau *function* yang digunakan dapat dilihat pada tabel 2.3.



Gambar 3.14 Contoh hasil keluaran dari tahapan segmentasi

3.4.4 *Histogram of Oriented Gradient*

Pada proses *Histogram of Oriented Gradients*, masukan untuk proses ini berupa citra yang berasal dari hasil segmentasi. Perhitungan fitur dari metode *Histogram of Oriented Gradient* ini dilakukan per citra karakter hasil segmentasi. 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 3.15 dapat dilihat hasil dari proses *resize* dan

III. ANALISIS DAN PERANCANGAN SISTEM

crop citra *grayscale* berukuran 8×4 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:

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

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

| | | | |
|-----|----|----|-----|
| 92 | -1 | 0 | -88 |
| 88 | 0 | -2 | -90 |
| 90 | -1 | 4 | -90 |
| 122 | 0 | 0 | -91 |
| 90 | 0 | 1 | -89 |
| 85 | 0 | 1 | -90 |
| 90 | 1 | 3 | -92 |
| 122 | 0 | -2 | -91 |

Gambar 3.16 Matriks hasil Perhitungan Gradien sumbu X

| | | | |
|-----|-----|-----|-----|
| 90 | 88 | 90 | 86 |
| 2 | -2 | 2 | 2 |
| 1 | 34 | 1 | 36 |
| -2 | 0 | -1 | -3 |
| -1 | -37 | -1 | -36 |
| 2 | 0 | 3 | 2 |
| 1 | 37 | 1 | 34 |
| -91 | -90 | -92 | -93 |

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

III. ANALISIS DAN PERANCANGAN SISTEM

(2,5) dan hasil dari tahap ini dapat dilihat pada Gambar 3.18:

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

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

| | | | |
|--------|-------|-------|--------|
| 128.70 | 88.01 | 90 | 123.05 |
| 88.03 | 2 | 2.83 | 90.02 |
| 90.01 | 34.02 | 4.12 | 96.93 |
| 122.02 | 0 | 1 | 91.05 |
| 90.01 | 37 | 1.41 | 96.01 |
| 85.02 | 0 | 3.16 | 90.02 |
| 90.01 | 37.01 | 3.16 | 98.08 |
| 152.2 | 90 | 92.02 | 130.12 |

Gambar 3.18 Matriks hasil Perhitungan *Magnitude*

| | | | |
|--------|-------|-------|--------|
| 44.37 | 90.65 | 89.99 | 135.66 |
| 1.30 | 90.03 | 135 | 178.73 |
| 0.64 | 91.69 | 14.04 | 158.19 |
| 179.06 | 0 | 90.06 | 1.89 |
| 179.36 | 90 | 135 | 22.02 |
| 1.35 | 0 | 71.57 | 178.73 |
| 0.64 | 88.45 | 18.44 | 159.72 |
| 143.28 | 90 | 88.76 | 45.62 |

Gambar 3.19 Matriks hasil Perhitungan Arah

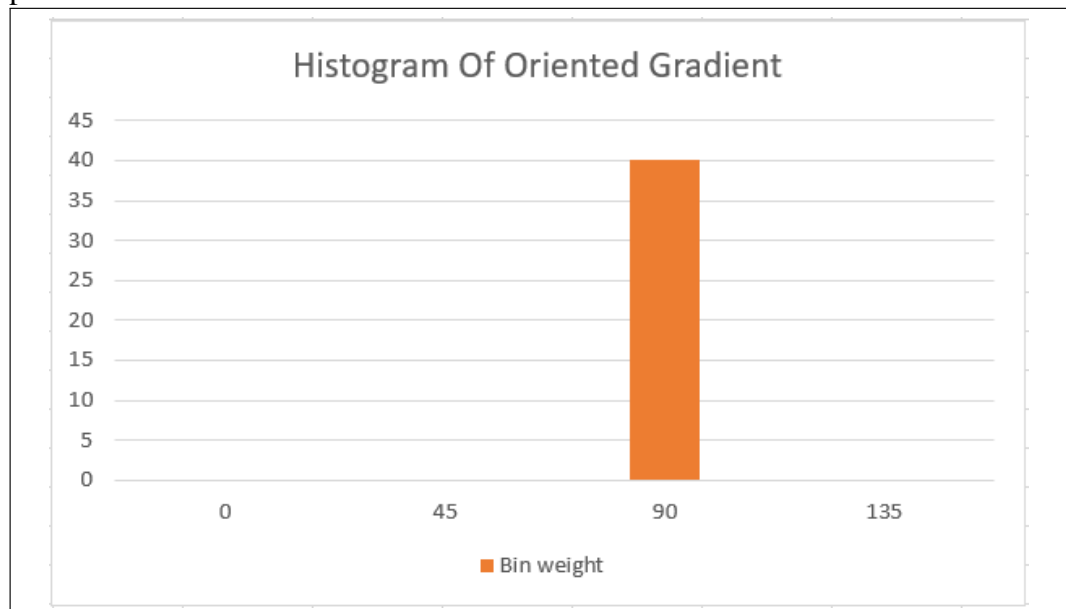
3. Kemudian, tentukan ukuran sel, ukuran blok dan jumlah *oriented histogram bins*. Pada penelitian Dalas dan Triggs untuk mendeteksi pejalan kaki sebelumnya, didapat bahwa ukuran sel sebesar 8×8 piksel, ukuran blok sebesar 2×2 ukuran sel dan jumlah bin sebanyak 9 sudah dapat menghasilkan akurasi yang 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. Untuk setiap blok, terdapat *overlapping* sebesar 50% dari ukuran blok. Dengan demikian akan didapatkan perhitungan sebagai berikut:
 - a. Jumlah sel adalah 8, terdiri dari 4 sel vertikal dan 2 sel horizontal.
 - b. Jumlah blok adalah 3, terdiri dari 3 blok vertikal dan 1 blok horizontal.

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 algoritme yaitu, *L1-Norm*, *L1-Sqrt*, *L2-Norm*, dan *L2-Hys*. Pada penelitian ini, penulis menggunakan algoritme normalisasi *L2-Norm* karena berdasarkan penelitian sebelumnya, hasil yang didapat lebih baik dari algoritme lainnya. Persamaan algoritme untuk proses normalisasi menggunakan *L2-Norm* didapat dengan menggunakan persamaan 2 . 12. Di bawah adalah contoh perhitungan normalisasi untuk

III. ANALISIS DAN PERANCANGAN SISTEM

blok pertama:

| | | | | | | | |
|--------|--------|--------|--------|--------|--------|-------|--------|
| 87.28 | 129.45 | 88.73 | 1.28 | 89.28 | 0 | 89.99 | 126.62 |
| 208.2 | 1.27 | 32.74 | 3.82 | 140.04 | 5.11 | 0.99 | 46.96 |
| 171.21 | 2.55 | 36.99 | 1.28 | 136.49 | 48.28 | 1.87 | 3.96 |
| 116.74 | 2.55 | 125.74 | 124.19 | 55.74 | 132.16 | 91.28 | 44.21 |

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.

$$L2_{Norm} = \sqrt{87.28^2 + 129.45^2 + \dots + 0.99^2 + 46.96^2} = 361.428$$

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):

$$\begin{bmatrix} 0.24148 & 0.35815 & 0.2455 & 0.00353 \end{bmatrix}$$

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 × lebar blok dalam satuan sel × jumlah pergeseran horizontal* dan jumlah baris sebesar *jumlah pergeseran vertikal × tinggi blok dalam satuan sel*, dengan perhitungan tersebut, dalam analisis saat ini didapatkan ukuran matriks sebesar 6×8 . Dalam analisis 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.

| | | | | | | | |
|------|------|------|------|------|------|------|------|
| 0.24 | 0.36 | 0.25 | 0.00 | 0.25 | 0.00 | 0.25 | 0.35 |
| 0.58 | 0.00 | 0.09 | 0.01 | 0.39 | 0.01 | 0.00 | 0.13 |
| 0.61 | 0.00 | 0.10 | 0.01 | 0.41 | 0.01 | 0.00 | 0.14 |
| 0.50 | 0.01 | 0.11 | 0.00 | 0.40 | 0.14 | 0.01 | 0.01 |
| 0.48 | 0.01 | 0.10 | 0.00 | 0.38 | 0.14 | 0.01 | 0.01 |
| 0.33 | 0.01 | 0.35 | 0.35 | 0.16 | 0.37 | 0.26 | 0.12 |

Gambar 3.22 Matriks hasil Normalisasi

Setelah mendapatkan matriks *HOG descriptor* di atas. Langkah berikutnya adalah menjadikan matriks tersebut sebagai vektor. Hal ini dilakukan dengan mengambil setiap baris dari matriks dan memasukkannya ke dalam matriks vektor berukuran $1 \times \text{jumlah fitur}$. Vektor inilah yang akan dijadikan sebagai masukan bagi metode *Machine Learning* yang akan digunakan dalam penelitian ini.

3.4.5 Support Vector Machine

Tahapan terakhir dari sistem deteksi dan pengenalan plat nomor kendaraan adalah klasifikasi karakter. Masukan untuk proses ini berupa fitur *HOG Descriptor* untuk setiap citra karakter plat nomor. Tahapan ini bertujuan untuk mengklasifikasikan fitur-fitur dari *HOG descriptor* yang dihasilkan dari perhitungan metode *Histogram of Oriented Gradient* agar dapat dikenali sebagai karakter. *Support Vector Machine* yang akan digunakan dalam penelitian menggunakan *library* dari Weka SVM. Untuk *function* atau *method* yang digunakan pada *library* tersebut dapat dilihat pada tabel 2.2. *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 karakter yang akan dikenali adalah huruf A sampai dengan Z dan angka dari 0 sampai dengan 9 sehingga akan terdapat 36 kelas untuk proses klasifikasi.

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 dari perangkat keras yang digunakan dalam pembangunan aplikasi adalah sebagai berikut:

1. *Laptop* ASUS A442UQ
2. *Processor* Intel Core i7-7500U CPU @ 2.7GHz
3. *Hard Disk* kapasitas 1TB
4. RAM 16GB

4.1.2 Lingkungan Perangkat Lunak

Spesifikasi dari perangkat lunak yang digunakan dalam pembangunan aplikasi adalah sebagai berikut:

1. Sistem Operasi Windows 10 Home 64 bit.
2. Netbeans IDE 8.2
3. Java Development Kit (JDK) 1.8.0_161
4. *Library* OpenCV 3.4.6

4.2 Implementasi Perangkat Lunak

Pada bab ini akan dijelaskan mengenai implementasi aplikasi untuk pengenalan karakter pada citra plat kendaraan. Di bawah ini merupakan daftar *class* dan *method* beserta penjelasan mengenai cara kerja program.

4.2.1 Daftar Class dan Method Gradient

Berikut adalah tabel berisi *method* pada *class* Gradient. *Class* Gradient digunakan untuk menyimpan nilai *orientation* dan nilai *magnitude* dari suatu piksel citra.

IV. IMPLEMENTASI DAN PENGUJIAN

Tabel 4.1 Daftar *Method Class Gradient*

| No | Nama <i>method</i> | Masukan | Keluaran | Keterangan |
|----|--------------------|---|----------|--|
| 1 | Gradient | double orientation, double magnitude | void | Metode <i>constructor</i> yang digunakan untuk inisialisasi objek dari kelas Gradient dengan nilai <i>orientation</i> dan <i>magnitude</i> yang didapatkan dari perhitungan. |
| 2 | getOrientation() | | double | Metode untuk mengembalikan nilai <i>orientation</i> dari suatu piksel. |
| 3 | getMagnitude() | | double | Metode yang digunakan untuk mengembalikan nilai <i>magnitude</i> dari suatu piksel. |
| 4 | setOrientation() | double orientation | void | Metode untuk mengatur nilai <i>orientation</i> dari objek Gradient berdasarkan nilai <i>orientation</i> yang dijadikan masukkan. |
| 5 | setMagnitude() | double magnitude | void | Metode yang digunakan untuk mengatur nilai <i>magnitude</i> dari objek Gradient berdasarkan nilai <i>magnitude</i> yang dijadikan masukkan. |

4.2.2 Daftar Class dan Method GradientCell

Berikut adalah tabel berisi *method* pada *class* GradientCell. *Class* GradientCell digunakan untuk menyimpan nilai gradien dari setiap sel.

Tabel 4.2 Daftar Method Class GradientCell

| No | Nama <i>method</i> | Masukan | Keluaran | Keterangan |
|----|--------------------|------------|---------------------|--|
| 1 | GradientCell() | int length | void | Metode <i>constructor</i> yang digunakan untuk inisialisasi objek dari kelas GradientCell. |
| 2 | getGradients() | | List <Gradient > | Metode untuk mengembalikan <i>List</i> dari gradien-gradien yang terdapat . |

4.2.3 Daftar Class dan Method HOG

Berikut adalah tabel berisi *method* pada *class* HOG. *Class* HOG digunakan untuk proses ekstraksi fitur dari citra karakter.

Tabel 4.3 Daftar Method Class HOG

| No | Nama <i>method</i> | Masukan | Keluaran | Keterangan |
|----|----------------------|---|----------|---|
| 1 | HOG() | Integer[][] image, int cellHeight, int cellWidth, int blockSize, int numBins | void | Metode <i>constructor</i> yang digunakan untuk inisialisasi objek dari kelas HOG. |
| 2 | extractHOGFeatures() | | double[] | Metode untuk mengekstraksi fitur <i>HOG descriptor</i> dari citra. |

IV. IMPLEMENTASI DAN PENGUJIAN

Tabel 4.3 Daftar *Method Class HOG*

| No | Nama <i>method</i> | Masukan | Keluaran | Keterangan |
|----|-----------------------------|---------|----------|--|
| 3 | calculateGradientAndCells() | | void | Metode yang digunakan untuk menghitung nilai gradien, <i>magnitude</i> , dan orientasi untuk setiap sel. |
| 4 | createHistograms() | | void | Metode untuk membentuk histogram untuk mencatat persebaran arah dari setiap sel. |
| 5 | histogramNormalization() | | void | Melakukan normalisasi <i>L2-Norm</i> untuk setiap elemen pada histogram. |
| 6 | createDescriptor() | | void | Membentuk <i>HOG descriptor</i> dari hasil normalisasi histogram. |

4.2.4 Daftar *Class* dan *Method SVM*

Berikut adalah tabel berisi *method* pada *class SVM*. *Class SVM* digunakan untuk perhitungan klasifikasi.

Tabel 4.4 Daftar *Method Class SVM*

| No | Nama <i>method</i> | Masukan | Keluaran | Keterangan |
|----|----------------------|--|------------|--|
| 1 | calculateRBFKernel() | double[][] data, double sigma, int classSource, int classTarget | double | Menghitung nilai RBF Kernel. |
| 2 | createRBFMatrix() | double[][] data, double[] sigma | double[][] | Membentuk matriks RBF dari data fitur. |

IV. IMPLEMENTASI DAN PENGUJIAN

| | | | | |
|---|------------------------|---|------------|--|
| 3 | createLinearEquation() | double[][] rbfMatrix, double[] classList | double[][] | Membuat persamaan linear dari matriks RBF. |
| 4 | getSolutions() | double[][] linearEquationMatrix, double[] classList | Matrix | Mendapatkan solusi dari persamaan linear yaitu nilai alpha dan bias. |
| 5 | createRBFTestMatrix() | double[][] data, double sigma, double[] classList | double | Membentuk matriks RBF untuk data pengujian. |
| 6 | classify() | double[][] solutions, double[] rbfTest, double[] classList | double | Mendapatkan nilai hasil klasifikasi berdasarkan data uji dan nilai alpha dan bias. |
| 7 | getDataFromText() | String path | double[][] | Membaca matriks fitur dari berkas teks. |

4.2.5 Daftar *Class* dan *Method* ConfusionMatrix

Berikut adalah tabel berisi *method* pada *class* ConfusionMatrix. *Class* ConfusionMatrix digunakan untuk perhitungan akurasi dari hasil klasifikasi karakter yang dilakukan dengan metode SVM. *Confusion Matrix* juga biasanya digunakan sebagai alat ukur untuk menghitung kinerja dari algoritme klasifikasi yang digunakan.

Tabel 4.5 Daftar *Method Class* ConfusionMatrix

| No | Nama <i>method</i> | Masukan | Keluaran | Keterangan |
|----|--------------------|--------------|----------|---|
| 1 | getClassIndex() | String label | int | Metode yang digunakan untuk mengembalikan nilai indeks <i>array</i> dari label yang dimasukkan. |

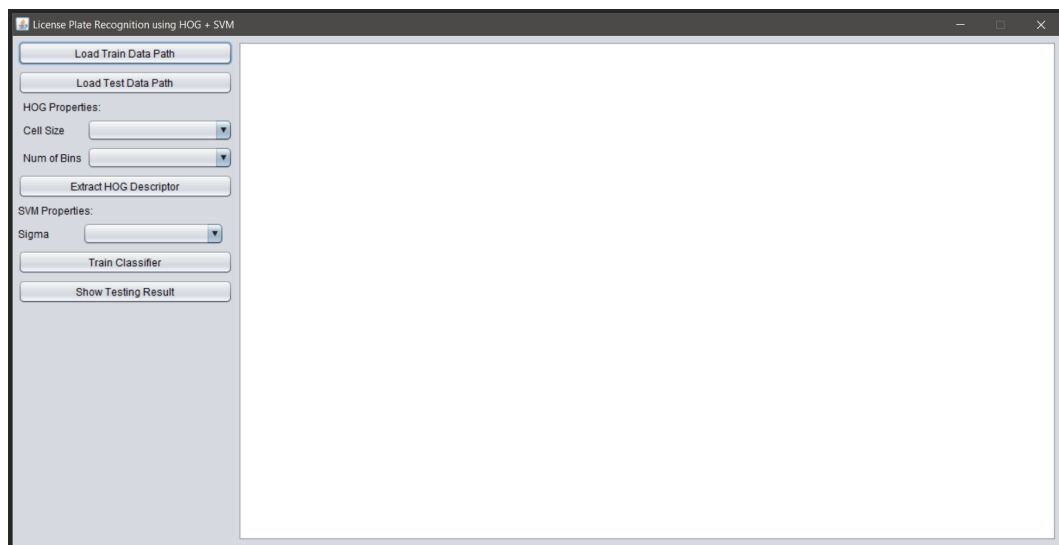
IV. IMPLEMENTASI DAN PENGUJIAN

Tabel 4.5 Daftar *Method Class ConfusionMatrix*

| No | Nama <i>method</i> | Masukan | Keluaran | Keterangan |
|----|-----------------------------------|-----------------------------|----------|---|
| 2 | <code>getConfusionMatrix()</code> | <code>int[][] result</code> | void | Metode untuk melakukan perhitungan <i>Confusion Matrix</i> dan juga menghitung akurasi dari hasil klasifikasi, metode ini juga akan menampilkan hasil <i>Confusion Matrix</i> sebagai keluaran pada antarmuka aplikasi. |

4.2.6 Tampilan Antarmuka Antar Aplikasi

Subbab ini akan menjelaskan tampilan antarmuka dari aplikasi pengenalan plat nomor kendaraan. Tampilan awal dari aplikasi ketika dibuka adalah seperti pada Gambar 4.1.



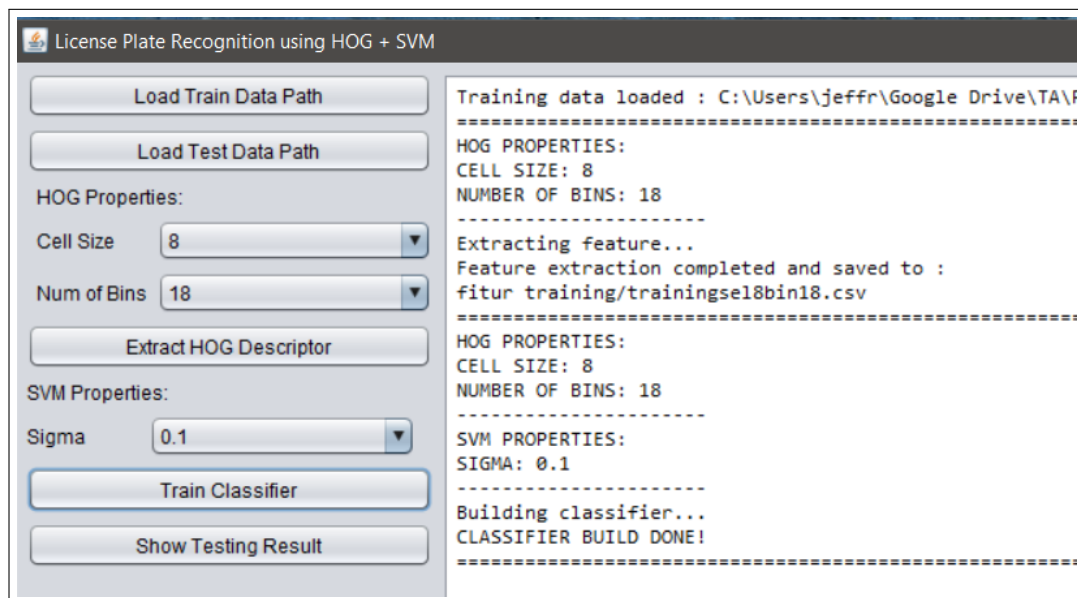
Gambar 4.1 Tampilan antarmuka aplikasi pengenalan plat nomor kendaraan

Pada Gambar 4.1, terdapat beberapa tombol diantaranya adalah tombol *Load Train Data Path*, *Load Test Data Path*, *Extract HOG Descriptor*, *Train Classifier*, dan *Show Testing Result*. Terdapat juga beberapa tombol *dropdown* yang berfungsi untuk memilih nilai dari parameter metode HOG dan SVM. Untuk metode HOG tombol *dropdown* yang tersedia adalah tombol *dropdown* untuk memilih ukuran sel (*Cell Size*) dan jumlah *bins* yang akan digunakan (*Num of Bins*). Sedangkan untuk metode SVM tombol *dropdown* yang tersedia adalah tombol *dropdown*

IV. IMPLEMENTASI DAN PENGUJIAN

untuk memilih nilai sigma yang akan digunakan. Tahapan proses pengenalan plat nomor kendaraan di aplikasi terbagi menjadi dua, yaitu proses *training* dan proses *testing*. Tahapan proses *training* pada aplikasi adalah sebagai berikut:

1. Memasukkan *path* dari kumpulan citra yang akan digunakan sebagai data latih untuk proses pengenalan karakter.
2. Memasukkan ukuran sel dan jumlah *bin* yang digunakan untuk metode HOG dengan memilih menggunakan tombol *dropdown Cell Size* dan tombol *dropdown Num of Bins*.
3. Klik tombol *Extract HOG Descriptor* untuk menjalankan proses ekstraksi fitur.
4. Klik tombol *Train Classifier* untuk menjalankan proses *training*.



Gambar 4.2 Tampilan antarmuka hasil *training*

Pada Gambar 4.2 aplikasi akan menampilkan *path* dari folder citra yang akan digunakan sebagai data latih, ukuran sel dan jumlah *bins* yang digunakan untuk metode HOG, pesan bahwa proses ekstraksi fitur sudah berjalan dan lokasi penyimpanan fitur, dan pesan bahwa proses *training* sudah selesai. Sedangkan untuk tahapan proses *testing* pada aplikasi adalah sebagai berikut:

1. Lakukan proses *training* terlebih dahulu.
2. Setelah proses *training* selesai dilakukan, kemudian masukkan *path* dari folder citra yang akan digunakan sebagai data uji.

IV. IMPLEMENTASI DAN PENGUJIAN

yang digunakan, kode yang digunakan adalah untuk mengkonversi citra RGB yang merupakan citra dengan 3 *channel* warna menjadi citra dengan 1 *channel* warna.

3. Lakukan proses deteksi tepi *Canny* dengan menggunakan *Imgproc.Canny(gray, edge)*. Dengan parameter *gray* adalah citra hasil *grayscale* dan parameter *edge* adalah penampung citra hasil deteksi tepi *Canny*.
4. Lakukan proses *Hough Transform* sebanyak dua kali, yang pertama untuk mencari kandidat garis vertikal dengan *range* θ -90 sampai dengan -85 derajat dan yang kedua adalah untuk mencari garis horizontal dengan *range* θ -10 sampai dengan 10 derajat. Masing-masing dari kandidat garis vertikal dan horizontal disimpan dalam variabel dengan jenis *ArrayList<Line>*.
5. Untuk setiap elemen dalam *ArrayList* yang berisi kandidat garis vertikal. Lakukan perbandingan antar elemen dengan membandingkan garis mana yang lebih kiri dan mana yang lebih kanan, kemudian dihitung lebar dari area yang dibatasi oleh kedua garis tersebut, apabila ukurannya berada dalam kisaran 255 - 390 piksel. Maka koordinat x dari titik awal kedua garis tersebut akan disimpan ke dalam matriks 2 dimensi yang berfungsi untuk menampung batas kiri, kanan, atas, dan bawah dari kandidat area plat.
6. Untuk setiap elemen dalam *ArrayList* yang berisi kandidat garis horizontal. Lakukan perbandingan antar elemen dengan membandingkan garis mana yang lebih atas dan mana yang lebih bawah, kemudian pasangkan dengan garis batas kiri dan kanan pada *array* penampung kandidat batas kiri dan kanan plat nomor, kemudian hitung tinggi dari area yang dibatasi kedua garis horizontal, apabila berada di kisaran 85 - 130 piksel, maka koordinat y dari titik awal kedua garis tersebut akan ditambahkan pada elemen matriks yang berisi kandidat batas kiri dan kanan tadi, sehingga matriks 2 dimensi akan berisi koordinat titik yang merupakan area plat.
7. Jika kandidat masih lebih dari satu kandidat, maka koordinat yang dipakai adalah koordinat yang rasio tinggi terhadap lebarnya paling mendekati 0.33.
8. Kemudian lakukan pemotongan citra RGB dengan menggunakan koordinat yang didapat, dan hasil dari proses ini adalah citra plat.

4.3 Pengujian

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 karakter pada sistem pengenalan karakter, oleh karena itu perlu diketahui berapa ukuran sel, ukuran blok, dan jumlah *bin* yang akan menghasilkan fitur yang paling baik untuk akurasi pengenalan karakter. Pengujian ini akan dilakukan dengan data latih sebanyak 117 citra karakter hasil segmentasi dari plat nomor untuk 36 kelas karakter yang terdiri dari 10 kelas angka dan 26 kelas huruf, dimana setiap kelas karakter memiliki jumlah data latih sebanyak 3-5 citra.

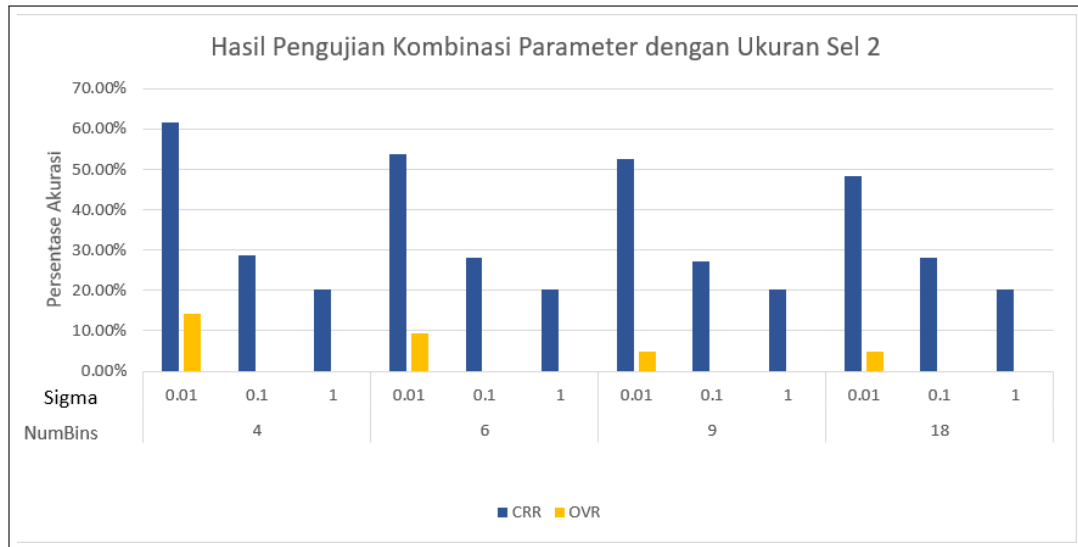
Fitur dari metode *HOG* akan digunakan pada proses klasifikasi karakter menggunakan *library* SVM dari Weka dan akan diukur akurasi menggunakan *Confusion Matrix*. Adapun nilai dari setiap parameter yang akan digunakan untuk kombinasi, yaitu:

1. Ukuran sel yang akan digunakan (dalam satuan piksel) : 2, 4, 8, 16
2. Jumlah *bin* yang akan digunakan : 4, 6, 9, 18
3. Nilai *sigma* pada metode *SVM* yang akan digunakan: 0.01, 0.1, 1
4. Campuran : Kombinasi parameter yang menghasilkan akurasi paling optimal untuk setiap karakter.

4.3.1 Pengujian Kombinasi Parameter

Pada skenario pengujian ini, pengujian akan dilakukan dengan menggunakan ukuran sel berukuran 2×2 piksel, 4×4 piksel, 8×8 piksel, dan 16×16 piksel. Setiap ukuran sel akan menggunakan ukuran blok 2×2 sel. 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:

IV. IMPLEMENTASI DAN PENGUJIAN



Gambar 4.6 Hasil Pengujian Kombinasi Parameter dengan Ukuran Sel 2

Berdasarkan Gambar 4.6, dapat disimpulkan bahwa tingkat akurasi pengenalan karakter (CRR) maksimal yang didapatkan apabila menggunakan ukuran sel 2×2 piksel adalah 61.53%. Kombinasi parameter yang digunakan untuk mencapai hasil tersebut adalah ukuran sel 2×2 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 masukan berukuran 32×32 piksel. Maka panjang vektor fitur dari *HOG descriptor* yang dihasilkan adalah 3600 fitur.

Grafik batang berwarna biru menunjukkan tingkat akurasi pengenalan karakter (CRR). CRR sendiri merupakan akronim dari *Character Recognition Rate* yang memiliki rumus jumlah karakter yang dikenali dibagi dengan keseluruhan karakter yang terdeteksi. Berdasarkan hasil CRR tertinggi pada Gambar 4.6, dari 143 karakter yang terdeteksi, sebanyak 88 di antaranya dapat diklasifikasikan dengan baik. Grafik batang berwarna kuning menunjukkan performa keseluruhan aplikasi (OVR). OVR merupakan akronim dari *Overall Performance* yang memiliki rumus jumlah plat nomor yang terdeteksi dan dikenali dengan benar dibagi dengan keseluruhan jumlah plat nomor yang ada. Berdasarkan hasil OVR tertinggi dari Gambar 4.6, dari 21 plat nomor yang terdeteksi, hanya 3 plat nomor yang dapat dikenali dengan baik.

Tabel 4.6 merupakan tabel yang menunjukkan hasil klasifikasi karakter dengan parameter HOG (*CellSize* dan *NumBins*) masing-masing 2 dan 4, dan nilai *sigma* untuk metode SVM 0.01.

IV. IMPLEMENTASI DAN PENGUJIAN

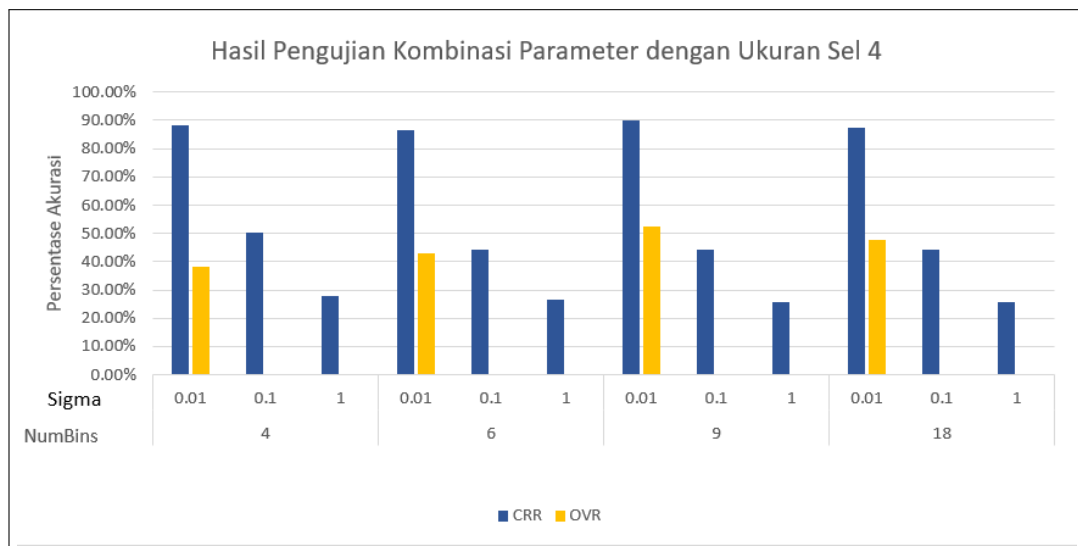
Tabel 4.6 Klasifikasi karakter dengan parameter CellSize = 2, NumBins = 4, dan Sigma = 0.01

| No | Karakter | Prediksi Benar | Prediksi Salah | Akurasi |
|----|----------|----------------|----------------|---------|
| 1 | 0 | 2 | 0 | 100.00% |
| 2 | 1 | 15 | 7 | 68.18% |
| 3 | 2 | 6 | 3 | 66.67% |
| 4 | 3 | 2 | 6 | 25.00% |
| 5 | 4 | 3 | 5 | 37.50% |
| 6 | 5 | 2 | 2 | 50.00% |
| 7 | 6 | 1 | 3 | 25.00% |
| 8 | 7 | 6 | 7 | 46.15% |
| 9 | 8 | 1 | 2 | 33.33% |
| 10 | 9 | 2 | 4 | 33.33% |
| 11 | A | 0 | 0 | - |
| 12 | B | 1 | 3 | 25.00% |
| 13 | C | 1 | 0 | 100.00% |
| 14 | D | 20 | 0 | 100.00% |
| 15 | E | 1 | 0 | 100.00% |
| 16 | F | 0 | 0 | - |
| 17 | G | 1 | 0 | 100.00% |
| 18 | H | 1 | 0 | 100.00% |
| 19 | I | 1 | 2 | 33.33% |
| 20 | J | 2 | 1 | 66.67% |
| 21 | K | 2 | 0 | 100.00% |
| 22 | L | 4 | 0 | 100.00% |
| 23 | M | 1 | 0 | 100.00% |
| 24 | N | 0 | 0 | - |
| 25 | O | 1 | 0 | 100.00% |
| 26 | P | 1 | 3 | 25.00% |
| 27 | Q | 2 | 0 | 100.00% |
| 28 | R | 1 | 1 | 50.00% |
| 29 | S | 1 | 1 | 50.00% |
| 30 | T | 1 | 0 | 100.00% |
| 31 | U | 1 | 0 | 100.00% |
| 32 | V | 1 | 0 | 100.00% |
| 33 | W | 2 | 5 | 28.57% |
| 34 | X | 0 | 0 | - |

IV. IMPLEMENTASI DAN PENGUJIAN

| No | Karakter | Prediksi Benar | Prediksi Salah | Akurasi |
|----|----------|----------------|----------------|---------|
| 35 | Y | 1 | 0 | 100.00% |
| 36 | Z | 1 | 0 | 100.00% |

Dengan tingkat akurasi pengenalan karakter (*Character Recognition Rate*) sebesar 61.53%, dapat dilihat pada tabel 4.6 bahwa dari 36 karakter yang ada, yang dapat diprediksi dengan benar 100% adalah sebanyak 16 karakter, terdapat juga beberapa karakter yang memiliki akurasi pengenalan yang rendah, diantaranya adalah angka 3, angka 4, angka 6, angka 7, angka 8, angka 9, huruf B, huruf I, huruf P, dan huruf W. Dari hasil akurasi klasifikasi di atas, dapat disimpulkan bahwa penggunaan ukuran sel 2×2 piksel kurang tepat untuk digunakan dalam proses ekstraksi fitur HOG dalam sistem pengenalan karakter ini.



Gambar 4.7 Hasil Pengujian Kombinasi Parameter dengan Ukuran Sel 4

Berdasarkan Gambar 4.7, dapat disimpulkan bahwa tingkat akurasi pengenalan karakter (CRR) maksimal yang didapatkan apabila menggunakan ukuran sel 4×4 piksel adalah 90.20%. Kombinasi parameter yang digunakan untuk mencapai hasil tersebut adalah ukuran sel 4×4 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 masukan berukuran 32×32 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.

Berdasarkan hasil CRR tertinggi pada Gambar 4.7, dari 143 karakter yang terdeteksi, sebanyak 129 di antaranya dapat diklasifikasikan dengan baik.

IV. IMPLEMENTASI DAN PENGUJIAN

Sedangkan dari hasil OVR tertinggi pada Gambar 4.7, dari 21 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.

Tabel 4.7 merupakan tabel yang menunjukkan hasil klasifikasi karakter dengan parameter HOG (*CellSize* dan *NumBins*) masing-masing 4 dan 9, dan nilai *sigma* untuk metode SVM 0.01.

Tabel 4.7 Klasifikasi karakter dengan parameter *CellSize* = 4, *NumBins* = 9, dan *Sigma* = 0.01

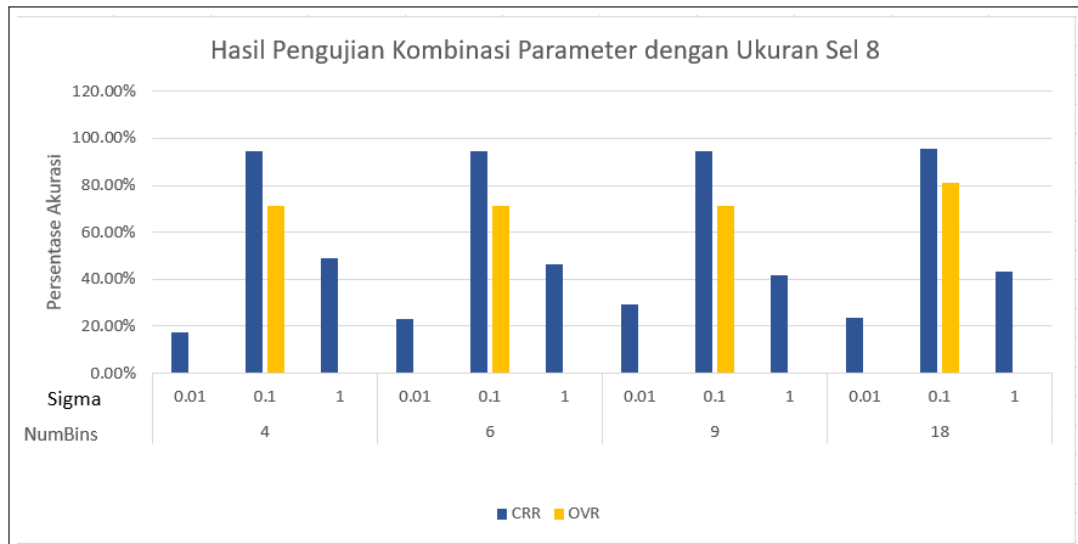
| No | Karakter | Prediksi Benar | Prediksi Salah | Akurasi |
|----|----------|----------------|----------------|---------|
| 1 | 0 | 0 | 2 | 0.00% |
| 2 | 1 | 18 | 4 | 81.82% |
| 3 | 2 | 7 | 2 | 77.78% |
| 4 | 3 | 8 | 0 | 100.00% |
| 5 | 4 | 8 | 0 | 100.00% |
| 6 | 5 | 4 | 0 | 100.00% |
| 7 | 6 | 4 | 0 | 100.00% |
| 8 | 7 | 13 | 0 | 100.00% |
| 9 | 8 | 3 | 0 | 100.00% |
| 10 | 9 | 6 | 0 | 100.00% |
| 11 | A | 0 | 0 | - |
| 12 | B | 4 | 0 | 100.00% |
| 13 | C | 1 | 0 | 100.00% |
| 14 | D | 19 | 1 | 95.00% |
| 15 | E | 1 | 0 | 100.00% |
| 16 | F | 0 | 0 | - |
| 17 | G | 1 | 0 | 100.00% |
| 18 | H | 1 | 0 | 100.00% |
| 19 | I | 3 | 0 | 100.00% |
| 20 | J | 2 | 1 | 66.67% |
| 21 | K | 2 | 0 | 100.00% |
| 22 | L | 4 | 0 | 100.00% |
| 23 | M | 1 | 0 | 100.00% |
| 24 | N | 0 | 0 | - |
| 25 | O | 1 | 0 | 100.00% |
| 26 | P | 4 | 0 | 100.00% |

IV. IMPLEMENTASI DAN PENGUJIAN

Tabel 4.7 Klasifikasi karakter dengan parameter CellSize = 4, NumBins = 9, dan Sigma = 0.01

| No | Karakter | Prediksi Benar | Prediksi Salah | Akurasi |
|----|----------|----------------|----------------|---------|
| 27 | Q | 0 | 2 | 0.00% |
| 28 | R | 2 | 0 | 100.00% |
| 29 | S | 2 | 0 | 100.00% |
| 30 | T | 1 | 0 | 100.00% |
| 31 | U | 0 | 1 | 0.00% |
| 32 | V | 1 | 0 | 100.00% |
| 33 | W | 6 | 1 | 85.71% |
| 34 | X | 0 | 0 | - |
| 35 | Y | 1 | 0 | 100.00% |
| 36 | Z | 1 | 0 | 100.00% |

Dengan tingkat akurasi pengenalan karakter (*Character Recognition Rate*) sebesar 90.20%, dapat dilihat pada tabel 4.7 bahwa dari 36 karakter yang ada, yang dapat diprediksi dengan benar 100% adalah sebanyak 24 karakter, terdapat juga beberapa karakter yang memiliki akurasi pengenalan yang rendah, diantaranya adalah angka 0, huruf Q, dan huruf U. Dari hasil akurasi klasifikasi di atas, dapat disimpulkan bahwa penggunaan ukuran sel 4×4 piksel sudah dapat meningkatkan akurasi pengenalan karakter namun masih belum optimal.



Gambar 4.8 Hasil Pengujian Kombinasi Parameter dengan Ukuran Sel 8

Berdasarkan Gambar 4.8, dapat disimpulkan bahwa tingkat akurasi pengenalan karakter maksimal yang didapatkan apabila menggunakan ukuran sel 8×8 piksel adalah 95.80%. Kombinasi parameter yang digunakan untuk mencapai hasil

IV. IMPLEMENTASI DAN PENGUJIAN

tersebut adalah ukuran sel 8×8 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 masukan berukuran 32×32 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.

Dari hasil CRR tertinggi pada Gambar 4.8, dari 143 karakter yang terdeteksi, sebanyak 137 di antaranya dapat diklasifikasikan dengan baik. Sedangkan dari hasil OVR tertinggi pada Gambar 4.8 menunjukkan, dari 21 plat nomor yang terdeteksi, 17 plat nomor dapat dikenali dengan baik, hal ini merupakan peningkatan apabila dibandingkan dengan hasil pengujian sebelumnya.

Tabel 4.8 merupakan tabel yang menunjukkan hasil klasifikasi karakter dengan parameter HOG (*CellSize* dan *NumBins*) masing-masing 8 dan 18, dan nilai *sigma* untuk metode *SVM* 0.1.

Tabel 4.8 Klasifikasi karakter dengan parameter *CellSize* = 8, *NumBins* = 18, dan *Sigma* = 0.1

| No | Karakter | Prediksi Benar | Prediksi Salah | Akurasi |
|----|----------|----------------|----------------|---------|
| 1 | 0 | 2 | 0 | 100.00% |
| 2 | 1 | 21 | 1 | 95.45% |
| 3 | 2 | 7 | 2 | 77.78% |
| 4 | 3 | 8 | 0 | 100.00% |
| 5 | 4 | 8 | 0 | 100.00% |
| 6 | 5 | 4 | 0 | 100.00% |
| 7 | 6 | 4 | 0 | 100.00% |
| 8 | 7 | 13 | 0 | 100.00% |
| 9 | 8 | 3 | 0 | 100.00% |
| 10 | 9 | 6 | 0 | 100.00% |
| 11 | A | 0 | 0 | - |
| 12 | B | 4 | 0 | 100.00% |
| 13 | C | 1 | 0 | 100.00% |
| 14 | D | 19 | 1 | 95.00% |
| 15 | E | 1 | 0 | 100.00% |
| 16 | F | 0 | 0 | - |
| 17 | G | 1 | 0 | 100.00% |
| 18 | H | 1 | 0 | 100.00% |

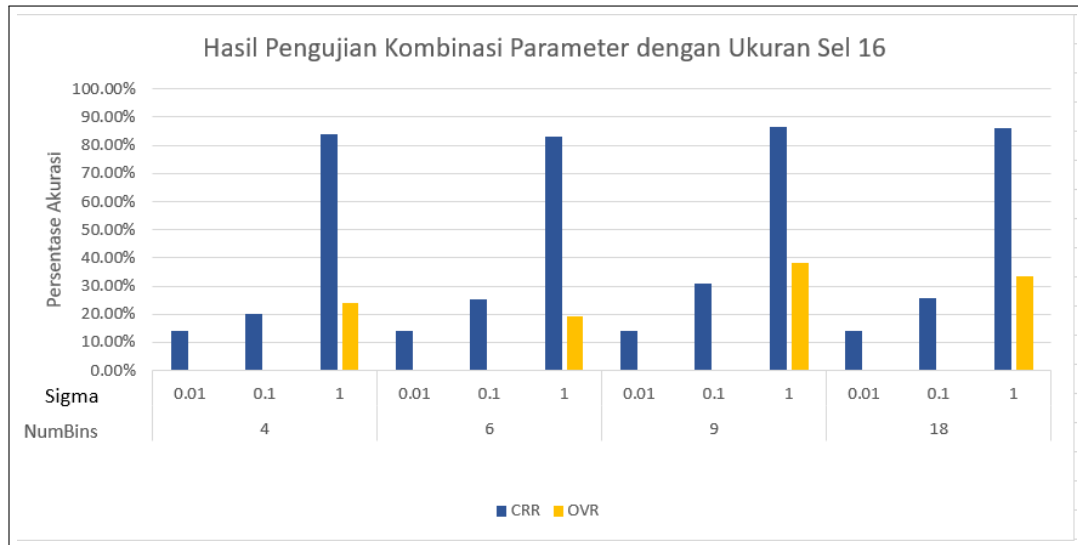
IV. IMPLEMENTASI DAN PENGUJIAN

Tabel 4.8 Klasifikasi karakter dengan parameter CellSize = 8, NumBins = 18, dan Sigma = 0.1

| No | Karakter | Prediksi Benar | Prediksi Salah | Akurasi |
|----|----------|----------------|----------------|---------|
| 19 | I | 3 | 0 | 100.00% |
| 20 | J | 3 | 0 | 100.00% |
| 21 | K | 2 | 0 | 100.00% |
| 22 | L | 4 | 0 | 100.00% |
| 23 | M | 1 | 0 | 100.00% |
| 24 | N | 0 | 0 | - |
| 25 | O | 1 | 0 | 100.00% |
| 26 | P | 4 | 0 | 100.00% |
| 27 | Q | 0 | 2 | 0.00% |
| 28 | R | 2 | 0 | 100.00% |
| 29 | S | 2 | 0 | 100.00% |
| 30 | T | 1 | 0 | 100.00% |
| 31 | U | 1 | 0 | 100.00% |
| 32 | V | 1 | 0 | 100.00% |
| 33 | W | 7 | 0 | 100.00% |
| 34 | X | 0 | 0 | - |
| 35 | Y | 1 | 0 | 100.00% |
| 36 | Z | 1 | 0 | 100.00% |

Dengan tingkat akurasi pengenalan karakter (*Character Recognition Rate*) sebesar 95.80%, dapat dilihat pada tabel 4.8 bahwa dari 36 karakter yang ada, yang dapat diprediksi dengan benar 100% adalah sebanyak 29 karakter, dari karakter yang tersisa, hanya satu karakter yang memiliki akurasi rendah, yaitu huruf Q yang mana dari 2 karakter yang terdapat di data *testing*, tidak ada yang dapat diklasifikasikan dengan benar. Dari hasil pengujian ini dapat disimpulkan bahwa penggunaan ukuran sel 8×8 piksel sudah dapat menghasilkan akurasi pengenalan karakter yang baik.

IV. IMPLEMENTASI DAN PENGUJIAN



Gambar 4.9 Hasil Pengujian Kombinasi Parameter dengan Ukuran Sel 16

Berdasarkan Gambar 4.9, dapat disimpulkan bahwa akurasi pengenalan karakter maksimal yang didapatkan apabila menggunakan ukuran sel 16×16 piksel adalah 86.71%. Kombinasi parameter yang digunakan untuk mencapai hasil tersebut adalah ukuran sel 16×16 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 masukan berukuran 32×32 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 95.80%.

Berdasarkan hasil CRR tertinggi pada Gambar 4.9, dari 143 karakter yang terdeteksi, sebanyak 123 di antaranya dapat diklasifikasikan dengan baik. Sedangkan berdasarkan hasil OVR tertinggi pada Gambar 4.9 dari 21 plat nomor yang terdeteksi, 7 plat nomor dapat dikenali dengan baik, hal ini merupakan dampak dari penurunan akurasi pengenalan karakter.

Tabel 4.9 merupakan tabel yang menunjukkan hasil klasifikasi karakter dengan parameter HOG (*CellSize* dan *NumBins*) masing-masing 16 dan 9, dan nilai *sigma* untuk metode SVM 1.

Tabel 4.9 Klasifikasi karakter dengan parameter *CellSize* = 16, *NumBins* = 9, dan *Sigma* = 1.0

| No | Karakter | Prediksi Benar | Prediksi Salah | Akurasi |
|----|----------|----------------|----------------|---------|
| 1 | 0 | 2 | 0 | 100.00% |
| 2 | 1 | 20 | 2 | 90.91% |
| 3 | 2 | 6 | 3 | 66.67% |

IV. IMPLEMENTASI DAN PENGUJIAN

Tabel 4.9 Klasifikasi karakter dengan parameter CellSize = 16, NumBins = 9, dan Sigma = 1.0

| No | Karakter | Prediksi Benar | Prediksi Salah | Akurasi |
|----|----------|----------------|----------------|---------|
| 4 | 3 | 8 | 0 | 100.00% |
| 5 | 4 | 8 | 0 | 100.00% |
| 6 | 5 | 4 | 0 | 100.00% |
| 7 | 6 | 4 | 0 | 100.00% |
| 8 | 7 | 13 | 0 | 100.00% |
| 9 | 8 | 3 | 0 | 100.00% |
| 10 | 9 | 6 | 0 | 100.00% |
| 11 | A | 0 | 0 | - |
| 12 | B | 3 | 1 | 75.00% |
| 13 | C | 1 | 0 | 100.00% |
| 14 | D | 19 | 1 | 95.00% |
| 15 | E | 1 | 0 | 100.00% |
| 16 | F | 0 | 0 | - |
| 17 | G | 1 | 0 | 100.00% |
| 18 | H | 1 | 0 | 100.00% |
| 19 | I | 0 | 3 | 0.00% |
| 20 | J | 3 | 0 | 100.00% |
| 21 | K | 2 | 0 | 100.00% |
| 22 | L | 4 | 0 | 100.00% |
| 23 | M | 1 | 0 | 100.00% |
| 24 | N | 0 | 0 | - |
| 25 | O | 1 | 0 | 100.00% |
| 26 | P | 4 | 0 | 100.00% |
| 27 | Q | 0 | 2 | 0.00% |
| 28 | R | 2 | 0 | 100.00% |
| 29 | S | 2 | 0 | 100.00% |
| 30 | T | 1 | 0 | 100.00% |
| 31 | U | 1 | 0 | 100.00% |
| 32 | V | 1 | 0 | 100.00% |
| 33 | W | 0 | 7 | 0.00% |
| 34 | X | 0 | 0 | - |
| 35 | Y | 1 | 0 | 100.00% |
| 36 | Z | 1 | 0 | 100.00% |

IV. IMPLEMENTASI DAN PENGUJIAN

Dengan tingkat akurasi pengenalan karakter (*Character Recognition Rate*) sebesar 86.71%, dapat dilihat pada tabel 4.9 bahwa dari 36 karakter yang ada, yang dapat diprediksi dengan benar 100% adalah sebanyak 25 karakter, terdapat juga karakter yang memiliki akurasi rendah, yaitu huruf I, huruf Q, dan huruf W. Apabila dibandingkan dengan hasil pengujian yang sebelumnya, dengan menggunakan ukuran sel 16 piksel justru malah mengurangi akurasi dari pengenalan karakter dan yang tadinya karakter tersebut sudah dapat dikenali dengan baik (huruf I dan huruf W) malah menjadi tidak bisa diklasifikasikan dengan benar sama sekali (akurasi 0% untuk kedua karakter tersebut). Dari pengujian ini dapat disimpulkan bahwa penggunaan ukuran sel 16×16 piksel dapat menghasilkan akurasi pengenalan karakter yang baik namun belum optimal.

4.3.2 Pengujian Kombinasi Parameter Campuran

Dari hasil pengujian kombinasi parameter pada subbab sebelumnya, akan didapatkan kombinasi parameter yang paling optimal untuk setiap karakter. Kombinasi parameter inilah yang akan digunakan dalam skenario pengujian, tujuan dari skenario ini adalah untuk mengetahui apakah penggabungan kombinasi parameter dapat meningkatkan tingkat akurasi pengenalan karakter. Kombinasi parameter yang paling optimal untuk karakter-karakter selain huruf D dan huruf Q adalah ukuran sel 8×8 piksel, jumlah *bin* sebanyak 18, dan nilai sigma sebesar 0.1. Khusus untuk huruf D dan huruf Q, kombinasi parameter yang digunakan adalah ukuran sel 2×2 piksel, jumlah *bin* sebanyak 4, dan nilai sigma sebesar 0.01. Untuk ukuran blok yang digunakan adalah sebesar 2×2 sel. Berikut adalah hasil klasifikasi karakter untuk penggunaan kombinasi parameter campuran:

Tabel 4.10 Klasifikasi karakter dengan kombinasi parameter campuran

| No | Karakter | Prediksi Benar | Prediksi Salah | Akurasi |
|----|----------|----------------|----------------|---------|
| 1 | 0 | 2 | 0 | 100.00% |
| 2 | 1 | 21 | 1 | 95.45% |
| 3 | 2 | 7 | 2 | 77.78% |
| 4 | 3 | 8 | 0 | 100.00% |
| 5 | 4 | 8 | 0 | 100.00% |
| 6 | 5 | 4 | 0 | 100.00% |
| 7 | 6 | 4 | 0 | 100.00% |
| 8 | 7 | 13 | 0 | 100.00% |
| 9 | 8 | 3 | 0 | 100.00% |
| 10 | 9 | 6 | 0 | 100.00% |

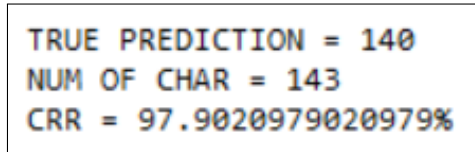
IV. IMPLEMENTASI DAN PENGUJIAN

Tabel 4.10 Klasifikasi karakter dengan kombinasi parameter campuran

| No | Karakter | Prediksi Benar | Prediksi Salah | Akurasi |
|----|----------|----------------|----------------|---------|
| 11 | A | 0 | 0 | - |
| 12 | B | 4 | 0 | 100.00% |
| 13 | C | 1 | 0 | 100.00% |
| 14 | D | 20 | 0 | 100.00% |
| 15 | E | 1 | 0 | 100.00% |
| 16 | F | 0 | 0 | - |
| 17 | G | 1 | 0 | 100.00% |
| 18 | H | 1 | 0 | 100.00% |
| 19 | I | 3 | 0 | 100.00% |
| 20 | J | 3 | 0 | 100.00% |
| 21 | K | 2 | 0 | 100.00% |
| 22 | L | 4 | 0 | 100.00% |
| 23 | M | 1 | 0 | 100.00% |
| 24 | N | 0 | 0 | - |
| 25 | O | 1 | 0 | 100.00% |
| 26 | P | 4 | 0 | 100.00% |
| 27 | Q | 2 | 0 | 100.00% |
| 28 | R | 2 | 0 | 100.00% |
| 29 | S | 2 | 0 | 100.00% |
| 30 | T | 1 | 0 | 100.00% |
| 31 | U | 1 | 0 | 100.00% |
| 32 | V | 1 | 0 | 100.00% |
| 33 | W | 7 | 0 | 100.00% |
| 34 | X | 0 | 0 | - |
| 35 | Y | 1 | 0 | 100.00% |
| 36 | Z | 1 | 0 | 100.00% |

PLATE IDENTIFIED = 19
PLATE PROCESSED = 21
OVR = 90.47619047619048%

Gambar 4.10 Hasil OVR Pengujian Kombinasi Parameter Gabungan



```
TRUE PREDICTION = 140
NUM OF CHAR = 143
CRR = 97.9020979020979%
```

Gambar 4.11 Hasil CRR Pengujian Kombinasi Parameter Gabungan

Dari Gambar 4.11 *Character Recognition Rate* yang didapatkan adalah sebesar 97.90%. Tingkat akurasi *Character Recognition Rate* (CRR) meningkat 2% terhadap hasil CRR kombinasi parameter dengan ukuran sel 8×8 piksel, jumlah *bin* 18, dan nilai sigma 0.1. Dari 143 karakter yang terdeteksi, sebanyak 140 di antaranya dapat diklasifikasikan dengan baik, hal ini juga meningkatkan jumlah plat nomor yang berhasil dikenali. Sedangkan berdasarkan hasil OVR pada Gambar 4.10 dari 21 plat nomor yang terdeteksi 19 plat nomor dapat dikenali dengan baik.

4.4 Analisis Pengujian

Setiap skenario pengujian yang dilakukan pada subbab 4.3.1 dan 4.3.2 diukur dengan melihat tingkat akurasi ada *Character Recognition Rate* dan *Overall Performance* yang mengukur performa keseluruhan aplikasi. Berdasarkan hasil pengujian pada Gambar 4.6 sampai Gambar 4.9, tingginya tingkat akurasi OVR sangat bergantung kepada tingginya tingkat akurasi CRR, semakin tinggi tingkat akurasi dari CRR maka tingkat akurasi dari OVR juga akan semakin meningkat. Pada beberapa hasil pengujian terlihat juga OVR menunjukkan angka 0%, hal ini disertai dengan rendahnya tingkat akurasi dari CRR pada hasil pengujian tersebut (biasanya terjadi ketika angka CRR berada di bawah 51%). Hal ini disebabkan karena OVR menghitung jumlah plat yang dikenali dengan benar terhadap jumlah keseluruhan plat yang terdeteksi oleh sistem. Kondisi plat disebut dikenali dengan benar adalah ketika keseluruhan hasil prediksi karakter untuk plat tersebut sama persis dengan karakter asli pada citra plat, satu saja hasil prediksi karakter yang tidak sama akan membuat plat tersebut tidak dapat dimasukkan ke dalam kategori plat yang dapat dikenali dengan benar. Oleh karena itu, jika aplikasi tidak dapat mengenali setiap karakter dengan baik (angka akurasi CRR rendah) maka nilai OVR yang didapat pun akan rendah dan bahkan 0.

Tingkat akurasi CRR yang didapatkan bergantung terhadap komposisi parameter yang digunakan. Berdasarkan hasil CRR hasil pengujian pada Gambar 4.6 sampai Gambar 4.9 tingkat akurasi CRR optimal yang didapatkan cenderung meningkat ketika menggunakan kombinasi parameter dengan ukuran sel 2×2 piksel sampai dengan 8×8 piksel. Namun ketika menggunakan kombinasi parameter dengan

IV. IMPLEMENTASI DAN PENGUJIAN

ukuran sel 16×16 piksel, tingkat akurasi CRR optimal yang didapatkan menurun. Dari pola tersebut dapat disimpulkan untuk menghasilkan tingkat CRR yang optimal diperlukan komposisi parameter yang tepat.

Untuk nilai sigma pada metode *SVM* yang digunakan agar tingkat akurasi CRR yang dihasilkan optimal, kecenderungan yang didapatkan dari hasil pengujian pada Gambar 4.6 sampai Gambar 4.9 adalah cenderung mengikuti besarnya ukuran sel, semakin ukuran selnya besar, maka semakin tinggi nilai sigma yang dibutuhkan.

Kombinasi parameter terbaik untuk setiap karakter dapat ditentukan dengan melihat hasil klasifikasi dari setiap hasil pengujian kombinasi parameter, yaitu pada tabel 4.6 sampai dengan tabel 4.9. Dari tabel-tabel tersebut didapati bahwa kombinasi parameter dengan ukuran sel 8×8 piksel dapat mengklasifikasikan mayoritas karakter dengan baik terkecuali huruf Q dan huruf D. Kedua huruf tersebut memiliki akurasi klasifikasi yang paling baik ketika diuji menggunakan kombinasi parameter dengan ukuran 2×2 piksel, jumlah *bin* 4, dan nilai sigma sebesar 0.01. Untuk huruf Q, ketika ia diuji menggunakan kombinasi parameter selain dengan ukuran sel 2×2 piksel, maka huruf tersebut akan selalu tertukar dengan angka 0.



(a)

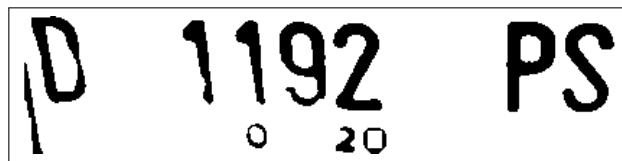


(b)

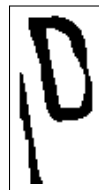
Gambar 4.12 Contoh citra (a) Huruf Q (b) Angka 0

Dari Gambar 4.12 dapat terlihat bahwa karakteristik objek yang dimiliki huruf Q dengan angka 0 memang mirip sehingga ketika dilakukan ekstraksi fitur terhadap kedua karakter tersebut tidak menutup kemungkinan fitur yang dihasilkan mirip. Oleh karena itu penggunaan ukuran sel 2×2 piksel memang tepat jika digunakan untuk mengklasifikasikan huruf Q dikarenakan ukuran sel 2×2 piksel dapat menangkap detail dari objek huruf Q dengan lebih baik dibandingkan ukuran sel lainnya.

Untuk huruf D, dari keseluruhan citra huruf D yang terdeteksi sistem, terdapat 1 citra yang awalnya dapat diklasifikasi dengan benar ketika menggunakan ukuran sel $2 \times$ piksel, namun menjadi misklasifikasi ketika menggunakan ukuran sel lain. Setelah dilakukan analisis lebih lanjut, satu citra yang misklasifikasi tersebut ternyata memiliki masalah pada hasil segmentasi karakternya. Hal itu juga disebabkan karena citra karakter berasal dari citra plat yang bermasalah (citra plat miring dan tidak tersegmentasi horizontal dengan baik). Hal ini tentunya akan berdampak sangat besar terhadap hasil segmentasi vertikalnya. Citra plat hasil *preprocessing* dan segmentasi dan citra karakter D yang misklasifikasi dapat dilihat pada Gambar 4.13 dan Gambar 4.14.



Gambar 4.13 Citra Plat asal karakter D yang misklasifikasi



Gambar 4.14 Citra hasil segmentasi karakter huruf D

Untuk meningkatkan akurasi pengenalan huruf Q dan huruf D, digunakanlah dua kombinasi parameter seperti yang sudah disinggung pada subbab 4.3.2. Dan dari hasil pengujian yang didapatkan terbukti dapat meningkatkan akurasi pengenalan terhadap huruf Q dan huruf D. Dari hasil pengujian tersebut dapat disimpulkan bahwa kombinasi parameter yang berbeda dapat diterapkan pada sistem untuk mencapai hasil yang lebih optimal.

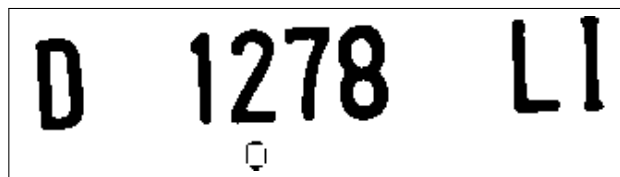
4.5 Analisis Kesalahan

Berdasarkan keseluruhan hasil skenario pengujian, didapatkan bahwa akurasi pengenalan karakter paling tinggi dihasilkan dari penggabungan kombinasi parameter yang paling optimal untuk setiap karakter. Namun berdasarkan tabel 4.10, terdapat tiga citra karakter yang mengalami misklasifikasi. Karakter-karakter tersebut adalah angka 1 sebanyak satu citra dan angka 2 sebanyak dua citra. Hasil misklasifikasi karakter-karakter tersebut dapat dilihat pada tabel 4.11

Tabel 4.11 Hasil klasifikasi karakter yang misklasifikasi

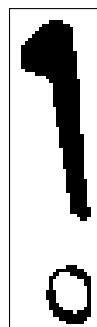
| No | Karakter | Hasil Klasifikasi |
|----|----------|-------------------|
| 1 | 1 | D |
| 2 | 2 | D |

Seperti yang ditunjukkan pada tabel 4.11, keseluruhan angka 1 dan 2 yang misklasifikasi dianggap sebagai karakter huruf D. Kombinasi parameter yang digunakan untuk mengklasifikasikan karakter angka 1 dan angka 2 adalah kombinasi parameter yang paling optimal untuk kedua karakter tersebut berdasarkan hasil pengujian pada subbab 4.3.1, yaitu ukuran sel 8×8 piksel, jumlah *bin* 18, dan nilai sigma 0.1. Setelah ditelusuri, ternyata karakter angka 1 dan satu dari dua angka 2 yang misklasifikasi berasal dari citra plat yang sama, yaitu citra plat pada Gambar 4.13, sedangkan citra angka 2 yang satunya berasal dari citra plat seperti yang ditunjukkan pada Gambar 4.15.

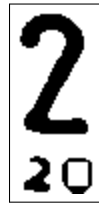


Gambar 4.15 Citra Plat asal karakter angka 2 yang misklasifikasi

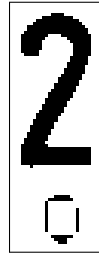
Pada gambar 4.13 dan Gambar 4.15, dapat dilihat bahwa citra plat yang didapatkan tidak lurus dan tidak tersegmentasi horizontal dengan baik. Hal ini akan mempengaruhi hasil segmentasi vertikal ketika mencari area karakter dari plat tersebut. Hasil segmentasi untuk setiap karakter dapat dilihat pada Gambar 4.16.



(a)



(b)



(c)

Gambar 4.16 Citra hasil segmentasi karakter misklasifikasi (a) Citra angka 1 dengan satu objek lain (b) Citra angka 2 dengan dua objek lain (c) Citra angka 2 dengan satu objek lain

Dari citra hasil segmentasi pada Gambar 4.16 dapat dilihat bahwa terdapat objek lain selain objek utama pada hasil segmentasi vertikal ketiga citra karakter angka 1 dan citra karakter angka 2 yang misklasifikasi. Hal ini akan berdampak pada hasil fitur yang akan dihasilkan dari metode *HOG* nantinya. Oleh karena itu, bisa disimpulkan kondisi citra plat yang kurang baik yang menyebabkan kedua karakter ini tidak dapat diidentifikasi.

BAB V

PENUTUP

Bab ini berisi kesimpulan yang dilandasi oleh penelitian dan pengujian yang telah dilakukan, serta dilengkapi dengan saran yang dapat untuk perkembangan ke depan.

5.1 Kesimpulan

Pengujian dari penerapan metode *Histogram of Oriented Gradient* dan *Support Vector Machine* menghasilkan beragam hasil. Hasil pengujian-pengujian tersebut menghasilkan kesimpulan sebagai berikut:

1. Hasil akurasi terbaik pengenalan plat nomor kendaraan dengan menggunakan metode *Histogram of Oriented Gradient* dan *Support Vector Machine* adalah 97.90% yang dicapai ketika menggunakan ukuran sel 8×8 piksel yang digabungkan dengan ukuran sel 2×2 piksel untuk karakter-karakter yang kurang dapat dikenali dengan baik ketika menggunakan ukuran sel 8×8 piksel, ukuran blok 2×2 sel (16×16 piksel untuk ukuran sel 8×8 piksel dan 4×4 piksel untuk ukuran sel 2×2 piksel), jumlah *bin* sebanyak 18 untuk ukuran sel 8×8 piksel dan 4 untuk ukuran sel 2×2 piksel, dan nilai sigma untuk metode *Support Vector Machine* sebesar 0.1 untuk ukuran sel 8×8 piksel dan 0.01 untuk ukuran sel 2×2 piksel.
2. Karakteristik objek berpengaruh terhadap ukuran sel HOG yang digunakan, untuk karakter yang memiliki karakteristik yang umum tidak akan mengalami masalah ketika menggunakan ukuran sel yang besar, sedangkan untuk karakter yang memiliki karakteristik khusus memerlukan ukuran sel yang lebih kecil. Hal ini dapat terlihat dari akurasi karakter huruf C, E, G, H, K, L, M, O, T, V, Y, dan huruf Z yang memiliki rata-rata akurasi mencapai 100% untuk setiap ukuran sel. Sedangkan huruf Q merupakan huruf khusus yang baru dapat dikenali dengan baik ketika menggunakan ukuran sel 2×2 piksel, selain ukuran sel tersebut, huruf Q akan selalu tertukar dengan karakter huruf O.
3. Nilai sigma pada metode *Support Vector Machine* yang dapat menghasilkan akurasi yang optimal bergantung terhadap ukuran sel yang digunakan.

Kecenderungannya adalah semakin besar ukuran sel yang digunakan, maka nilai sigma yang diperlukan semakin besar dan sebaliknya.

5.2 Saran

Saran untuk pengembangan sistem pengenalan plat nomor kendaraan adalah:

1. Ekstraksi fitur dapat dicoba menggunakan metode-metode seperti *zoning*, metode yang berbasis *moments* seperti *Geometric Moments*, *Zernike Moment*, dan *Orthogonal Fourier-Mellin Moments*, atau metode seperti *Discrete Wavelet Transform* dan *Restricted Boltzman Machine* yang dapat menangani permasalahan translasi dan rotasi terhadap karakter.
2. Untuk pengembangan lebih lanjut, pengenalan karakter dapat dicoba dengan menggunakan metode *Deep Learning* seperti misalnya metode *Convolutional Neural Network* yang dapat melakukan ekstraksi fitur dan klasifikasi dalam satu metode.

DAFTAR REFERENSI

- [1] S. S. Tabrizi and N. Cavus, "A Hybrid KNN-SVM Model for Iranian License Plate Recognition," *Procedia Computer Science*, vol. 102, pp. 588–594, 2016.
- [2] C. Gou, K. Wang, Z. Yu, and H. Xie, "License plate recognition using MSER and HOG based on ELM," *Proceedings of 2014 IEEE International Conference on Service Operations and Logistics, and Informatics*, 2014.
- [3] C. Gou, K. Wang, Y. Yao, and Z. Li, "Vehicle License Plate Recognition Based on Extremal Regions and Restricted Boltzmann Machines," *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, no. 4, pp. 1096–1107, 2016.
- [4] S. Rasheed, A. Naeem, and O. Ishaq, "Automated Number Plate Recognition Using Hough Lines and Template Matching," *Proceedings of the World Congress on Engineering and Computer Science*, vol. 1, pp. 24–26, 2012.
- [5] R. C. Gonzalez and R. E. Woods, *Digital Image Processing*, 4th ed. Pearson, 2017.
- [6] H. K. Ragb and V. K. Asari, "Multi-feature fusion and PCA based approach for efficient human detection," *2016 IEEE Applied Imagery Pattern Recognition Workshop (AIPR)*, 2016.
- [7] F. Y. Shih, *Image Processing and Pattern Recognition: Fundamentals and Techniques*, 1st ed. Wiley-IEEE Press, 2010.
- [8] "Computer Vision CITS4240: Lab 6", [Online]. Available: <http://teaching.csse.uwa.edu.au/units/CITS4240/Labs/Lab6/lab6.html> [Accessed: 16-Apr-2019].
- [9] "Diagonals of a Rectangle", [Online]. Available: <https://www.mathopenref.com/rectanglediagonals.html> [Accessed: 13-May-2019].
- [10] O. Oechsle, "Finding Straight Lines with the Hough Transform", 2012. [Online]. Available: <http://vase.essex.ac.uk/software/HoughTransform/> [Accessed: 16-Apr-2019].

- [11] N. Dalal and B. Triggs, "Histograms of Oriented Gradients for Human Detection," *Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, Jul. 2005.
- [12] Y. Ma and G. Guo, *Support Vector Machines Applications*, 2014th ed. New York: Springer, 2014.
- [13] K. Markham, "Simple Guide to Confusion Matrix Terminology", 2014. [Online]. Available: <https://www.dataschool.io/simple-guide-to-confusion-matrix-terminology/> [Accessed: 22-Apr-2019].
- [14] A. Nugroho and K.R.R. Wardhani, "Aplikasi Sistem Pembaca Plat Nomor Mobil Menggunakan Pengolahan Citra dan Metode Learning Vector Quantization," *Institut Teknologi Harapan Bangsa (ITHB)*, 2011.
- [15] "Peraturan Kapolri Nomor 5 tahun 2012 tentang Registrasi dan Identifikasi Kendaraan Bermotor", [Online]. Available: <http://kepri.polri.go.id/pid/wp-content/uploads/2019/01/PERATURAN-KAPOLRI-NOMOR-5-TAHUN-2012-TENTANG-REGISTRASI-DAN-IDENTIFIKASI-KENDARAAN-BERMOTOR.pdf> [Accessed: 22-Apr-2019]
- [16] A. H. Ashtari, M. J. Nordin, and M. Fathy, "An Iranian License Plate Recognition System Based on Color Features," *IEEE Transactions on Intelligent Transportation Systems*, vol. 15, no. 4, pp. 1690–1705, 2014.
- [17] R. Fisher, et all, "Grayscale Images", 2003. [Online]. Available: <https://homepages.inf.ed.ac.uk/rbf/HIPR2/gryimage.htm> [Accessed: 26-Jun-2019]