KLASIFIKASI GOLONGAN KENDARAAN BERDASARKAN FITUR HISTOGRAM OF ORIENTED GRADIENTS (HOG) MENGGUNAKAN METODE K-NEAREST NEIGHBORS (K-NN) BERBASIS RASPBERRY PI 3

SKRIPSI

Untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Teknik

Disusun oleh:
Lilo Nofrizal Akbar
NIM: 145150300111090

PROGRAM STUDI TEKNIK KOMPUTER
JURUSAN TEKNIK INFORMATIKA
FAKULTAS ILMU KOMPUTER
UNIVERSITAS BRAWIJAYA
MALANG
2019

PENGESAHAN

KLASIFIKASI GOLONGAN KENDARAAN BERDASARKAN FITUR *HISTOGRAM OF ORIENTED GRADIENTS* (HOG) MENGGUNAKAN METODE K-*NEAREST NEIGHBORS* (K-NN) BERBASIS RASPBERRY PI 3

SKRIPSI

Untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Teknik

Disusun Oleh :
Lilo Nofrizal Akbar
NIM: 145150300111090

Skripsi ini telah diuji dan dinyatakan lulus pada
31 Juli 2019
Telah diperiksa dan disetujui oleh:

Dosen Pembimbing I

Dr. Eng. Fitri Utaminingrum, S.T., M.T.
NIP. 19820710 200812 2 001

Mengetahui
Ketua Jurusan Teknik Informatika

Tri Astoto Kurniawan, S.T., M.T., Ph.D
NIP. 19710518 200312 1 001

PERNYATAAN ORISINALITAS

Penulis menyatakan dengan sebenar-benarnya bahwa sepanjang pengetahuan saya, di dalam naskah skripsi ini tidak terdapat karya ilmiah yang pernah diajukan oleh orang lain untuk memperoleh gelar akademik di suatu perguruan tinggi, dan tidak terdapat karya atau pendapat yang pernah ditulis atau diterbitkan oleh orang lain, kecuali yang secara tertulis disitasi dalam naskah ini dan disebutkan dalam daftar pustaka.

Apabila ternyata di dalam naskah skripsi ini dapat dibuktikan terdapat unsur-unsur plagiasi, saya bersedia skripsi ini digugurkan dan gelar akademik yang telah saya peroleh (sarjana) dibatalkan, serta diproses sesuai dengan peraturan perundang-undangan yang berlaku (UU No. 20 Tahun 2003, Pasal 25 ayat 2 dan Pasal 70).

Malang, 11 Juli 2019

Lilo Nofrizal Akbar

NIM: 145150300111090

KATA PENGANTAR

Segala puji hanya milik Allah SWT. Shalawat dan salam selalu tercurahkan kepada Rasulullah SAW. Berkat limpahan dan rahmat-Nya penyusun mampu menyelesaikan skripsi yang berjudul "Klasifikasi Golongan Kendaraan Berdasarkan Fitur Histogram of Oriented Gradients (HOG) Menggunakan Metode k-Nearest Neighbors (K-NN) Berbasis Raspberry Pi 3" guna memperoleh gelar Sarjana Teknik.

Dalam penyusunan dan penulisan skripsi ini tidak lepas dari bantuan-bantuan yang diberikan oleh berbagai pihak, maka penulis mengucapkan banyak terima kasih kepada :

1. Bapak Wayan Firdaus Mahmudy, S.Si., M.T., Ph.D. selaku Dekan Fakultas Ilmu Komputer Universitas Brawijaya Malang.
2. Bapak Tri Astoto Kurniawan, S.T., M.T., Ph.D. selaku Ketua Jurusan Teknik Informatika Universitas Brawijaya Malang.
3. Bapak Dahnial Syauqy, S.T., M.T., M.Sc. selaku Ketua Program Studi Teknik Komputer Universitas Brawijaya Malang.
4. Ibu Dr. Eng. Fitri Utaminingrum, S.T., M.T. selaku dosen pembimbing I yang telah memberikan banyak pengarahan, bimbingan dan semangat sehingga dapat menyelesaikan skripsi ini dengan baik.
5. Bapak dan Ibuk, Pakde dan Bude, yang telah berjasa hingga penulis bisa menyelesaikan perkuliahan.
6. Aga, Wahyu, Jup, Fau, Giri, Intan, Silvia, Putri, dan Tiya yang menemani masa-masa akhir perkuliahan, serta selalu membantu dan memberikan semangat selama proses pengerjaan skripsi ini.
7. Seluruh teman-teman Teknik Komputer 2014 yang pernah menemani selama waktu perkuliahan.
8. Seluruh pihak yang tidak dapat disebutkan satu persatu yang telah berperan dalam penyelesaian skripsi ini.

Penulis menyadari dalam penyusunan skripsi ini terdapat banyak kekurangan, sehingga saran dan kritik yang membangun sangat dibutuhkan untuk ke depannya agar lebih baik lagi. Semoga penelitian ini dapat memberikan manfaat bagi semua pihak yang membacanya.

Malang, 11 Juli 2019

Lilo Nofrizal Akbar

nofrizal.lilo@gmail.com

ABSTRAK

Lilo Nofrizal Akbar, Klasifikasi Golongan Kendaraan Berdasarkan Fitur Histogram of Oriented Gradients (HOG) Menggunakan Metode K-Nearest Neighbors (K-NN) Berbasis Raspberry Pi 3

Pembimbing: Dr. Eng. Fitri Utaminingrum, S.T., M.T.

Antrean pada jalan tol masih menjadi permasalahan di Indonesia yang seharusnya jalan tol menjadi jalan bebas hambatan, hal ini dapat terjadi karena beberapa faktor, salah satunya tidak adanya sistem otomatisasi untuk melakukan klasifikasi terhadap kendaraan yang melewati jalan tol, terlebih untuk kendaraan besar seperti truk yang dibedakan menjadi beberapa golongan menyebabkan petugas pintu tol harus secara manual membedakan golongan kendaraan, ini tentunya semakin menambah potensi kemacetan. Salah satu upaya untuk mengatasi permasalahan tersebut dalam penelitian ini dilakukan perancangan sistem untuk mengklasifikasi jenis golongan kendaraan menggunakan *image processing* berdasarkan fitur tampilan lokal kendaraan dari samping. Metode dalam ekstraksi fitur kendaraan menggunakan *Histogram of Oriented Gradients* (HOG), dan untuk klasifikasi golongan kendaraan menggunakan *k-Nearest Neighbors* (k-NN). Sistem dalam menangkap video kendaraan yang lewat menggunakan *webcam*, untuk kemudian citra diproses pada Raspberry Pi 3 untuk mendapatkan fitur HOG, yang kemudian nilai fitur HOG dilakukan klasifikasi k-NN dengan menghitung jarak antara data uji dengan data latih menggunakan perhitungan *Euclidean Distance* untuk mendapatkan hasil klasifikasi golongan kendaraan. Pengujian sistem dilakukan dengan 5 data uji tiap golongan kendaraan, terdapat sebanyak 5 golongan kendaraan. Didapatkan hasil akurasi pada golongan 1 sebesar 80%, golongan 2 sebesar 80%, golongan 3 sebesar 60%, golongan 4 sebesar 60%, golongan 5 sebesar 60%.

Kata kunci: klasifikasi kendaraan, jalan tol, *histogram of oriented gradients, k-nearest neighbors, background substraction,* raspberry pi

ABSTRACT

Lilo Nofrizal Akbar, Klasifikasi Golongan Kendaraan Berdasarkan Fitur Histogram of Oriented Gradients (HOG) Menggunakan Metode K-Nearest Neighbors (K-NN) Berbasis Raspberry Pi 3

Supervisor: Dr. Eng. Fitri Utaminingrum, S.T., M.T.

Queue on toll roads is still a problem in Indonesia which toll roads should be a freeway, this can occur due to several factors, one of which is the absence of an automation system for classify incoming vehicles, especially for large vehicles such as trucks that are divided into several class cause toll roads officers must have to manually differentiate them, this certainly adds to the potential for congestion. One effort to overcome these problems in this study is to design a system to classify the types of vehicles using image processing based on the local features of the vehicle from the side. The method for extracting vehicle features using Histogram of Oriented Gradients (HOG), and for classifying vehicle class using k-Nearest Neighbors (k-NN). The System for capturing video vehicles using a webcam, then the image is processed on the Raspberry Pi 3 to get the HOG feature, which then the HOG feature is performed by k-NN classification by calculating the distance between the test data and training data using the Euclidean Distance calculation to get the classification results vehicle class. System testing is carried out with 5 test data for each class of vehicles, there are 5 classes of vehicles. Accuracy obtained on vehicle class 1 by 80%, vehicle class 2 by 80%, vehicle class 3 by 60%, vehicle class 4 by 60%, vehicle class 5 by 60%.

***Keywords**: vehicle classification, toll roads, histogram of oriented gradients, k-nearest neighbors, background substraction,* raspberry pi.

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

DAFTAR LAMPIRAN

BAB 1 PENDAHULUAN

Bab ini membahas latar belakang, rumusan masalah, tujuan, manfaat, batasan masalah, dan sistematika pembahasan dari penelitian ini.

1.1 Latar Belakang

Jalan Tol di negara Indonesia digunakan sebagai sarana jalan bebas hambatan yang dikhususkan untuk kendaraan roda empat atau lebih, seperti mobil, truk, dan bis. Untuk menggunakan jalan tol ini pengendara diharuskan membayar sejumlah uang sesuai golongan kendaraan yang mereka gunakan, pembayaran dilakukan pada gerbang tol/pintu tol/gardu tol. Sistem e-Toll yang ada sekarang masih belum terdapat fasilitas dalam mengklasifikasikan jenis kendaraan berdasarkan golongannya, sehingga untuk kendaraan besar khususnya truk yang dibedakan menjadi 5 golongan harus dibedakan secara manual golongannya oleh petugas Jasa Marga, ini tentunya tidak praktis karena kendaraan harus berhenti lebih lama yang menyebabkan antrean panjang ketika volume kendaraan meningkat drastis seperti pada saat liburan, atau mudik lebaran. Bahkan tragedi Brexit beberapa tahun yang lalu, yang diiringi dengan adanya kematian dari pengguna jalan yang disebabkan antrean yang sangat banyak saat libur lebaran. Belasan korban meninggal karena kecelakaan hingga kelelahan akibat macet yang tak kunjung terurai (Kumparan, 2016).

Computer vision merupakan salah satu bidang teknologi komputer yang memiliki banyak sekali manfaat dalam penerapan di kehidupan sehari-hari. Teknologi pada *computer vision* memungkinkan kita dapat memproses suatu gambar dengan mengekstrak informasi dari gambar tersebut untuk keperluan deteksi objek, pengenalan objek, dan pengklasifikasian objek pada suatu gambar. Tujuannya adalah untuk dapat meniru penglihatan manusia dalam mendeteksi, mengenali, dan menganalisis suatu objek. *Computer vision* dimanfaatkan dalam berbagai sektor bidang seperti, kesehatan, industri, militer, riset, dll.

Secara luas *computer vision* berkaitan dengan bidang lain dan dapat dikombinasikan dengan artificial intelligence (kecerdasan buatan), robotika, otomasi industri, *neurobiology*, *intelligent transportation systems*, dll. Pada bidang industri contohnya digunakan untuk proses otomasi *quality control* dimana *computer vision* bertugas untuk memerikasa apakah terdapat cacat produk pada saat produksi. Pada bidang otomotif *computer vision* juga digunakan, contohnya pada kendaraan *autonomous* (kendaraan yang dapat berjalan sendiri), fitur keselamatan pengereman otomatis saat mendeteksi objek didepan kendaraan untuk menghindari tabrakan, dan lain-lain. Pada bidang robotika *computer vision* juga digunakan contohnya pada *mobile robot* seperti *drone* dan robot penjelajah untuk mengenali lingkungan sekitar, mendeteksi objek, dan lain-lain. Sedangkan pada bidang *Intelligent transportation systems*, dapat digunakan untuk perhitungan jumlah kendaraan, pengenalan rambu lalu lintas, dan lain sebagainya.

Dari permasalahan yang sudah dipaparkan tadi akan dibuat alat untuk mendeteksi kendaraan yang lewat pada gerbang tol untuk dilakukan klasifikasi sesuai golongan kendaraan tersebut secara otomatis dengan memanfaatkan Kamera sebagai media yang menangkap gambar. Pendeteksian kendaraan dilakukan dengan cara mengambil gambar kendaraan pada daerah pintu gerbang Tol, kemudian gambar tersebut diolah menggunakan metode *Histogram of Oriented Gradient* (HOG) untuk mendapatkan fitur yang sesuai dengan golongan kendaraannya.

Histogram of Oriented Gradient (HOG) adalah *feature descriptor* yang digunakan pada bidang teknologi computer vision dan image processing untuk mengenali tampilan lokal dari objek. Teknik pada HOG melakukan perhitungan orientasi gradien pada bagian gambar yang akan dicari fitur HOG-nya. Digunakan metode HOG karena memiliki keuntungan fleksibilitas jika dibanding metode lain, karena pada metode ini mengacu pada *feature based* yang melihat tampilan lokal serta melihat objek yang terdapat pada gambar yang berdasar pada intensitas dan distribusi gradien (Kachouane, et al., 2012).

Setelah mendapat nilai fitur HOG dari gambar, selanjutnya dilakukan proses untuk mengklasifikasi objek yang terdapat pada gambar menggunakan algoritma k-Nearest Neighbors (K-NN) untuk menentukan golongan kendaraan. Digunakan metode k-NN karena merupakan teknik yang sederhana, efisien, dan efektif digunakan untuk mengenali pola, maupun pengolahan objek yang lainnya, dan mampu untuk melakukan *training* data dengan skala besar (Bathia, 2010). K-NN termasuk pada kelompok algoritma pembelajaran *instance-based learning* yang membandingkan data baru terhadap data latih yang disimpan pada memori. K-NN dalam melakukan klasifikasi terhadap data baru dengan cara membandingkan data baru tersebut dengan data latih yang sudah diberi label, kemudian dicari sebanyak *k* data yang memiliki jarak terdekat, selanjutnya kelas yang paling dominan pada *k data* digunakan untuk prediksi kelas data uji. Contoh kasusnya, misal dibutuhkan solusi untuk menangani suatu penyakit pada pasien baru, dengan melihat solusi yang diberikan kepada pasien lama. Untuk menemukan solusi pada pasien baru maka dicari kemiripan kasus dengan pasien lama, solusi dari kasus pasien lama yang memiliki kemiripan dengan kasus pasien baru itu yang nantinya digunakan untuk memberi solusi tindakan pada pasien dengan kasus baru (Leidiyana, 2013).

Dengan adanya sistem ini nantinya dapat dikembangkan dan diterapkan untuk menunjang sistem *Multi Lane Free Flow* (MLFF) dimana pengendara tidak perlu berhenti saat melakukan pembayaran jalan tol akan tetapi pembayaran langsung ditagihkan melalui data pemilik kendaraan berdasarkan plat nomernya, sehingga hal ini akan mengurangi tingkat antrian yang panjang pada gerbang Tol dan efisiensi pada penggunaan sumber daya manusia (Petugas Jasa Marga) yang bertugas menjaga gerbang tersebut.

1.2 Rumusan Masalah

Rumusan masalah yang diangkat dalam penelitian ini sebagai berikut:

1. Bagaimana mengklasifikasikan kendaraan pada jalan Tol sesuai dengan golongannya dari video tangkapan kamera?
2. Berapa nilai *k* tetangga tedekat yang paling optimal untuk melakukan klasifikasi?
3. Berapa tingkat akurasi masing-masing klasifikasi golongan kendaraan?

1.3 Tujuan

Tujuan dilakukannya penelitian ini adalah seperti yang disebutkan sebagai berikut:

1. Membuat sistem yang dapat mengklasifikasi kendaraan dari masukan berupa video tangkapan kamera secara *real time*.
2. Membuat sistem yang dapat mengklasifikasikan kendaraan sesuai dengan golongan kendaraan pada jalan Tol.

1.4 Manfaat

Manfaat yang dapat diperoleh dari penelitian ini diantaranya disebutkan sebagai berikut:

1. Dapat digunakan untuk mengklasifikasikan golongan kendaraan yang lewat di jalan Tol secara otomatis.
2. Dapat diterapkan untuk sistem pembayaran jalan Tol otomatis, sehingga mempermudah pengendara dalam melakukan pembayaran jalan Tol.
3. Jika dikembangkan lebih lanjut dapat diterapkan pada jalan Tol MLFF, sehingga menghindari antrean yang panjang pada jalan Tol yang disebabkan oleh pengendara berhenti di gerbang Tol saat melakukan pembayaran secara manual.

1.5 Batasan Masalah

Pada penelitian ini terdapat beberapa batasan permasalahan yang ditentukan agar penelitian tidak terlalu meluas, diantaranya sebagai berikut:

1. Kendaraan yang akan diklasifikasi diambil video dari samping kendaraan mencakup seluruh badan kendaraan.
2. Video kendaraan yang akan diklasifikasi diambil pada siang hari.
3. Video kendaraan yang akan diklasifikasi diambil dari jarak 15 meter.
4. Kendaraan yang diklasifikasi hanya kendaraan yang diperbolehkan masuk Tol.
5. Pelatihan k-NN menggunakan 40 data latih pada tiap golongan kendaraan.
6. Pengujian menggunakan 5 data uji pada tiap golongan kendaraan.
7. Dalam menghitung jarak terdekat antara data uji dan data latih digunakan persamaan Euclidean distance.

1.6 Sistematika Pembahasan

Penjelasan singkat dari pembahasan tiap bab pada penelitian ini dijelaskan pada sistematika pembahasan berikut:

BAB 1 Pendahuluan

Bab ini menjelaskan latar belakang penelitian, rumusan masalah yang diangkat, tujuan penelitian, manfaat penelitian, batasan masalah dalam penelitian, dan sistematika pembahasan dalam penelitian.

BAB 2 Landasan Kepustakaan

Bab ini menjelaskan penelitian-penelitian sebelumnya yang dijadikan rujukan dalam menyelesaikan penelitian ini, serta teori-teori yang dipakai dalam penelitian.

BAB 3 Metodologi

Bab ini menjelaskan urutan langkah kerja yang dilakukan dalam penelitian ini, seperti studi literatur, analisis kebutuhan, perancangan sistem dan implementasi sistem, pengujian dan evaluasi, serta kesimpulan dari penelitian yang dilakukan.

BAB 4 Rekayasa Kebutuhan

Bab ini menjelaskan kebutuhan apa saja yang diperlukan oleh sistem, meliputi kebutuhan perangkat keras yang digunakan,kebutuhan perangkat lunak yang digunakan, kebutuhan fungsional yang harus dipenuhi oleh sistem, kebutuhan non fungsional sistem, serta penjelasan gambaran umum dari sistem.

BAB 5 Perancangan dan Implementasi

Bab ini menjelaskan tentang perancangan dan implementasi sistem nantinya. Perancangan akan menjelaskan alur program dan metode yang digunakan. Kemudian program dan metode yang sudah dirancang diimplementasikan pada tahap implementasi.

BAB 6 Pengujian

Bab ini menjelaskan tujuan dan prosedur dalam melakukan pengujian dari sistem yang sudah dirancang dan diimplementasi, serta ditunjukkan hasil dan analisis dari pengujian yang sudah dilakukan.

BAB 7 Penutup

Bab ini menjelaskan kesimpulan dari analisis terhadap penelitian yang sudah dilakukan, serta pemberian saran dari penulis untuk meningkatkan kinerja sistem yang untuk pengembangan sistem selanjutnya.

BAB 2 LANDASAN KEPUSTAKAAN

Bab ini menjelaskan penelitian-penelitian sebelumnya yang dijadikan rujukan dalam menyelesaikan penelitian ini, serta teori-teori yang dipakai dalam penelitian.

2.1 Isi Landasan Kepustakaan

Metode HOG memiliki keuntungan fleksibilitas jika dibanding metode lain, karena pada metode ini mengacu pada *feature based* yang melihat tampilan lokal serta melihat objek yang terdapat pada gambar yang berdasar pada intensitas dan distribusi gradien. Kecepatan sistem dalam melakukan pendeteksian dipengaruhi beberapa faktor yaitu kualitas gambar yang diambil, intensitas pencahayaan pada lingkungan sekitar objek, dan penulisan kode program yang dibuat (Febriliyanti, Faradila, Taqwa, 2017).

Background substraction sangat penting pada banyak pengaplikasian *computer vision*. Kita menggunakannya untuk menghitung kendaraan yang melewati pintu tol, menghitung orang yang masuk dan keluar toko, mendeteksi pergerakan objek, dan lain-lain (Rosebrock, 2015).

Ada beberapa metode yang digunakan dalam mendeteksi tepi citra, diantaranya Canny, Robert, dan Prewitt. Dari penelitian yang sudah dilakukan sebelumnya menunjukkan metode sobel memiliki hasil paling bagus dalam melakukan deteksi tepi citra dengan pengurangan *noise* yang signifikan (Adistya, Muslim, 2016).

Terdapat beberapa metode yang digunakan untuk menghitung jarak kedekatan antara data uji dengan data latih, metode tersebut antara lain euclidean distance, dan manhattan distance, namun penggunaannya yang paling sering yaitu euclidean distance (Bramer,2007).

Metode pengklasifikasian data menggunakan algoritma k-*nearest neighbors* termasuk salah satu metode yang konsisten dalam melakukan klasifikasi. Proses klasifikasi dilakukan dengan menghitung kedekatan nilai tiap fitur yang dimiliki antara data uji dengan data pelatihan (Kusrini dan Lutfhi, 2009). Telah banyak penelitian tentang algoritma k-*nearest neighbors* yang dilakukan sebelumnya dan didapatkan kesimpulan, k-*nearest neighbors* merupakan teknik yang sederhana, efisien, dan efektif digunakan untuk mengenali pola, mengkategori teks, maupun pengolahan objek yang lainnya, dan mampu untuk melakukan *training* data dengan skala besar (Bathia, 2010).

2.2 Dasar Teori

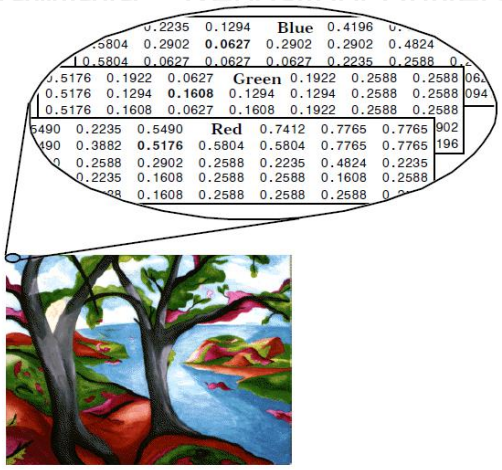
2.2.1 Citra gambar *digital*

Citra gambar digital merupakan gambar statis (foto) maupun gambar bergerak (video) yang *pixel (picture element)* atau elemen terkecil pada gambar direpresentasikan dengan nilai angka yang dapat diolah oleh komputer, *pixel*

merupakan perpotongan antara *m* kolom dan *n* baris pada matriks dua dimensi (*x, y*) (Kusumanto, Tompunu, 2011). Pada penelitian ini jenis citra yang digunakan yaitu citra berwarna atau RGB *(Red, Green, Blue), grayscale,* dan *binary*.

a. Citra gambar warna atau RGB (*Red, Green, Blue*)

Pada citra gambar berwarna atau disebut RGB tiap *pixel*-nya terdiri atas tiga *channel* warna yaitu merah, hijau, dan biru, yang tiap warnanya memiliki nilai antara 0 – 255. Pada Gambar 2.1 ditunjukkan citra gambar berwarna.

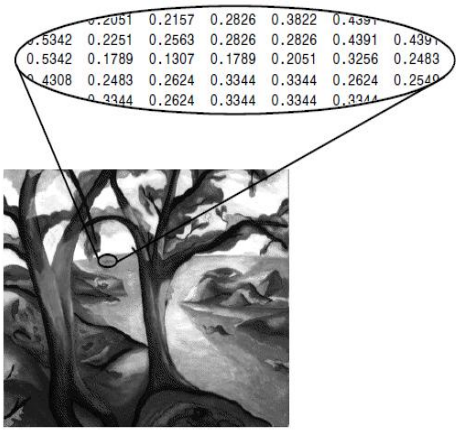


Gambar 2.1 Citra gambar berwarna atau RGB

Sumber: (Robotix, 2018)

b. Citra gambar *black and white (grayscale)*.

Pada citra gambar *black and white* atau disebut *grayscale* tiap *pixel*-nya hanya terdiri dari satu *channel* warna yaitu abu-abu yang memiliki nilai antara 0 – 255. Pada Gambar 2.2 ditunjukkan citra gambar *grayscale*.

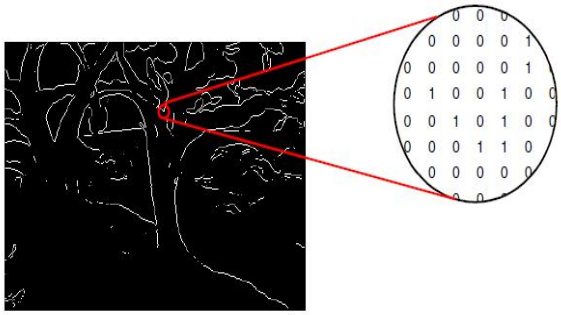


Gambar 2.2 Citra gambar *grayscale*

Sumber: (Robotix, 2018)

c. Citra gambar *binary*

Pada citra gambar *binary* atau disebut biner tiap *pixel*-nya hanya memiliki 2 nilai warna yaitu 0 (hitam), dan 1 (putih). Pada Gambar 2.3 ditunjukkan citra gambar biner.

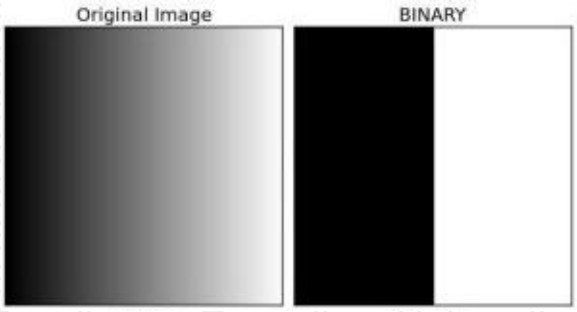


Gambar 2.3 Citra gambar biner

Sumber: (Robotix, 2018)

2.2.2 *Treshold*

Treshold yang dimaksud dalam penelitian ini yaitu, nilai batas yang ditentukan untuk melakukan konversi citra dari *grayscale* menjadi *binary*. Jika nilai dari suatu pixel lebih kecil dari nilai *treshold*, *pixel* tersebut akan diberi nilai 0, sebaliknya jika nilai *pixel* sama atau diatas nilai *treshold*, *pixel* akan diberi nilai maksimal. Hasil proses *treshold* ditunjukkan seperti pada Gambar 2.4.

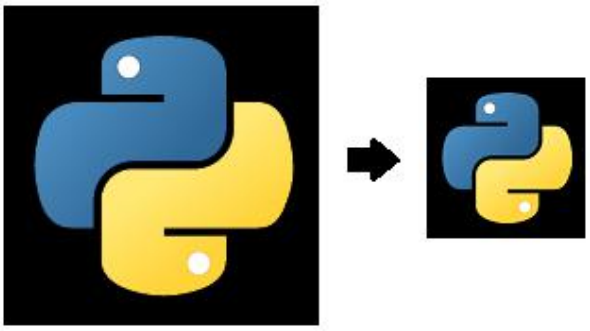


Gambar 2.4 Citra hasil treshold

Sumber: (OpenCV, 2019)

2.2.3 *Resize*

Resize merupakan proses mengubah dimensi gambar baik lebar, tinggi, ataupun keduanya, juga aspek rasio dari gambar asli dapat dipertahankan pada gambar hasil *resize*. Hasil proses *resize* ditunjukkan seperti pada Gambar 2.5.

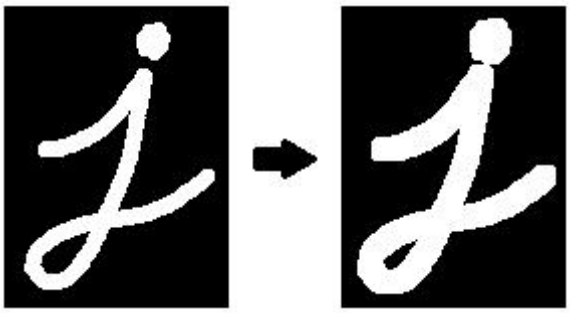


Gambar 2.5 *Resizing* gambar

Sumber: (Tutorialkart, 2019)

2.2.4 Dilasi

Dilasi merupakan teknik untuk memperbesar objek pada gambar sesuai dengan bentuk objek aslinya. Operasi ini melibatkan proses konvolusi dari suatu gambar dengan suatu kernel, yang dapat memiliki berbagai bentuk atau ukuran, tapi biasanya berbentuk persegi dan lingkaran. Kernel tersebut memiliki titik pusat, biasanya berada di pusat kernel. Proses berjalan dengan melakukan pemindaian kernel terhadap gambar, kemudian menghitung nilai *pixel* maksimal yang bertindihan (*overlapped*) dengan kernel dan mengganti *pixel* gambar yang berada pada titik pusat dengan nilai maksimal tersebut. Hasil proses dilasi ditunjukkan seperti pada Gambar 2.6.



Gambar 2.6 *Resizing* gambar

Sumber: (OpenCV, 2019)

2.2.5 Konversi warna

Konversi warna merupakan proses mengubah suatu satuan warna ke satuan warna lain. Dalam penelitian dilakukan konversi warna RGB ke *grayscale*, dan RGB ke *binary*.

a. RGB ke *grayscale*

Konversi warna dari RGB ke *grayscale* dilakukan dengan mengubah dari sebelumnya 3 *channel* warna *red, green, blue,* menjadi 1 *channel* warna abu-abu. Pada Persamaan 2.2 memperlihatkan persamaan untuk mengubah satuan warna dari RGB ke *grayscale* (Opencv, 2019).

$$RGB\ to\ Grayscale = ((0.299 * R) + (0.587 * G) + (0.114 * B)) \quad (2.2)$$

Keterangan Persamaan 2.2:

$R = nilai\ intensitas\ warna\ merah.$

$G = nilai\ intensitas\ warna\ hijau.$

$B = nilai\ intensitas\ warna\ biru.$

2.2.6 Kontur

Kontur yaitu sekumpulan titik yang dihubungkan oleh garis sepanjang batas luar yang memiliki intensitas warna yang sama. Kontur merupakan alat yang berguna untuk analisis bentuk, deteksi dan pengenalan objek (Opencv, 2019).

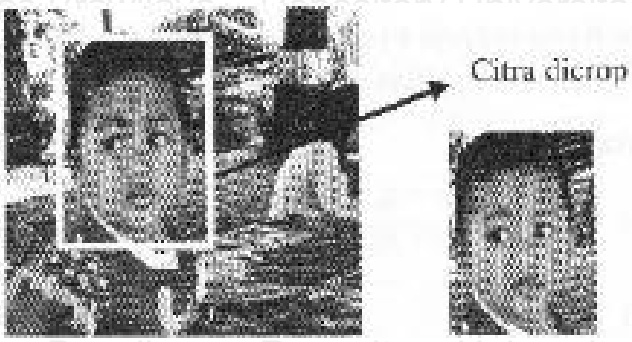
2.2.7 Smoothing

Smoothing atau juga disebut *blurring* merupakan operasi yang sering dipakai dalam *image processing.* Dalam penelitian ini smoothing digunakan untuk tujuan mengurangi *noise* pada gambar dengan menggunakan *Gaussian filter,* yaitu dengan melakukan konvolusi pada tiap titik pada masukan *array* (*pixel* masukan) dengan *Gaussian kernel* dan kemudian dijumlahkan semua untuk menghasilkan keluaran *array* (*pixel* keluaran) (Opencv, 2019). Hasil proses *blurring* ditunjukkan seperti pada Gambar 2.7.

Gambar 2.7 Hasil proses *blurring*

2.2.8 *Cropping*

Cropping atau proses memotong pada gambar bertujuan untuk membuang bagian gambar yang tidak diinginkan, dan menyisahkan bagian yang diinginkan. Pada Gambar 2.8 ditunjukkan contoh gambar yang dilakukan proses *cropping.*



Gambar 2.8 *Cropping* citra

Sumber : (Budisanjaya & Kumara, 2013)

2.2.9 *Bounding*

Bounding merupakan fitur pada OpenCV yang digunakan untuk memberi *border* dapat berupa persegi atau lingkaran pada *region of interest* (ROI) yang kita inginkan. Pada penelitian ini digunakan fungsi *straight bounding rectangle*, dimana akan dilakukan *bounding* terhadap objek tanpa memperhatikan rotasi dari objek tersebut (Opencv, 2019). Hasil proses *bounding* ditunjukkan seperti pada Gambar 2.9.

Gambar 2.9 *Hasil proses bounding*

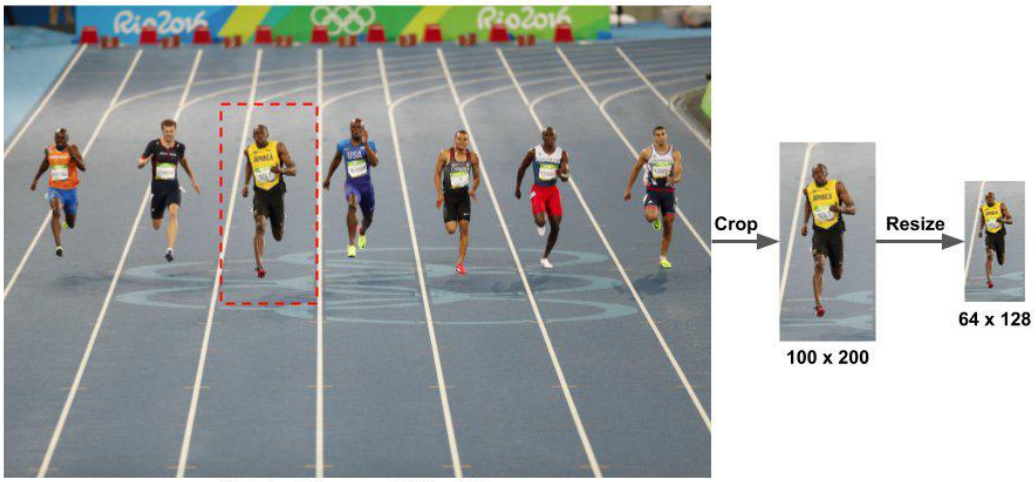
2.2.10 *Background Substraction*

Background substraction merupakan teknik *preprocessing* yang penting dalam kebanyakan aplikasi berbasis *vision.* Teknik ini digunakan untuk mendapatkan objek *foreground* yang terdapat pada *background*, dengan cara mengurangkan gambar yang terdapat objek dengan gambar *background* yang tidak terdapat objek, maka akan didapatkan gambar hanya objek *foreground* tanpa terdapat *background* (Opencv, 2019).

2.2.11 *Histogram of Oriented Gradients*

Histogram of Oriented Gradients (HOG) adalah *feature descriptor* yang digunakan pada bidang teknologi *computer vision* dan *image processing* untuk keperluan deteksi objek. *Histogram of Oriented Gradient* (HOG) adalah *feature descriptor* yang digunakan pada bidang teknologi *computer vision* dan *image processing* untuk mengenali tampilan lokal dari objek. Teknik pada HOG melakukan perhitungan orientasi gradien pada bagian gambar yang akan dicari fitur HOG-nya. Metode HOG memiliki keuntungan fleksibilitas jika dibanding metode lain, karena pada metode ini mengacu pada *feature based* yang melihat tampilan lokal serta melihat objek yang terdapat pada gambar yang berdasar pada intensitas dan distribusi gradien (Kachouan, et al., 2012).

Langkah dalam mencari nilai fitur HOG diawali dengan menentukan objek yang akan diekstraksi nilai fiturnya dengan memotong dari gambar asli dengan rasio ukuran 1:2 biasanya berukuran 64x128 *pixel* seperti pada Gambar2.10.



Gambar 2.10 Pemotongan objek pada gambar untuk menentukan *Region of Interest* (ROI)

Sumber: (Mallick, 2018)

Kemudian dihitung nilai *gradient horizontal* dan *vertical*-nya dengan cara melakukan *filter* pada gambar menggunakan *kernel*, *kernel filter* biasanya menggunakan operator Sobel, seperti ditunjukkan Persamaan 2.3.

$$G_x = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix} \quad G_y = \begin{bmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} \tag{2.3}$$

Keterangan Persamaan 2.3:

G_x = matriks operator Sobel sisi $horizontal$.

G_y = matriks operator Sobel sisi $vertical$.

Kemudian setelah mendapatkan nilai *gradient horizontal* dan *vertical*, dicari nilai *gradient magnitude* menggunakan Persamaan 2.4 dimana G_x = matriks operator Sobel sisi *horizontal*, dan G_y = matriks operator Sobel sisi *vertical*. Kemudian nilai *gradient direction* dicari menggunakan Persamaan 2.5 dimana G_x = matriks operator Sobel sisi *horizontal*, dan G_y= matriks operator Sobel sisi *vertical*.

$$g = \sqrt{g_x^2 + g_y^2} \tag{2.4}$$

Keterangan Persamaan 2.4:

$g = gradient\ magnitude$.

g_x = matriks operator Sobel sisi $horizontal$.

g_y = matriks operator Sobel sisi $vertical$.

$$\theta = \arctan\frac{g_y}{g_x} \quad (2.5)$$

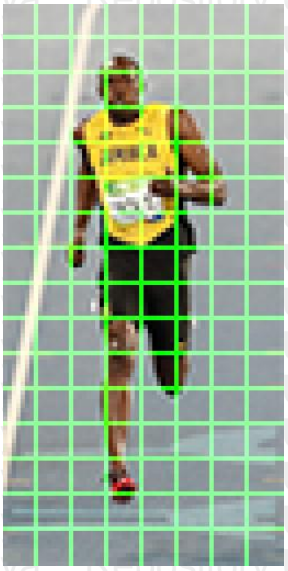
Keterangan Persamaan 2.5:

$\theta = gradient\ orientation.$

$g_x =$ matriks operator Sobel sisi $horizontal.$

$g_y =$ matriks operator Sobel sisi $vertical.$

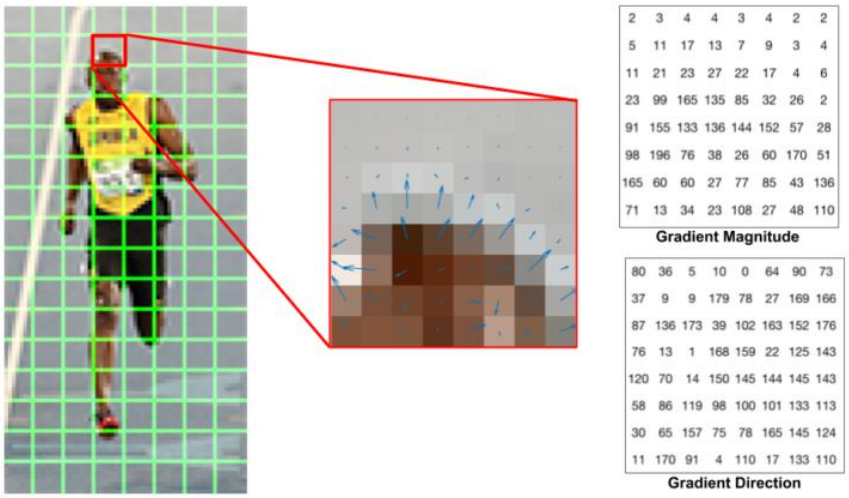
Proses selanjutnya mencari *histogram* gradien dari tiap sel berukuran 8x8 *pixel*, pada tahap ini gambar dibagi menjadi banyak sel berukuran 8x8 *pixel* seperti pada Gambar 2.11.



Gambar 2.11 Sel berukuran 8x8 *pixel* pada gambar

Sumber: (Mallick, 2018)

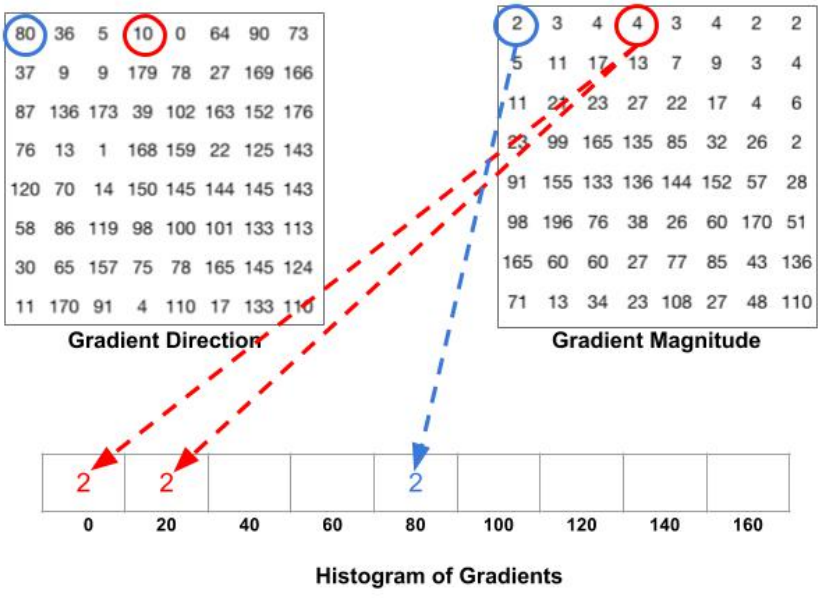
Tiap sel memiliki 8x8x3 = 192 nilai piksel, gradien pada tiap sel memiliki dua nilai yaitu *direction*, dan *magnitude* seperti pada Gambar 2.12.



Gambar 2.12 Nilai *gradient direction* dan *gradient magnitude* pada tiap sel

Sumber: (Mallick, 2018)

Histogram biasanya berupa *bin* yang berjumlah 9 buah bernilai 0°-160°, tiap *bin* selisih 20°. Nilai *direction* pada tiap piksel menentukan masuk kedalam *bin* mana, dan nilai *magnitude* menjadi bobot pada *bin*, contohnya *pixel* yang memiliki nilai *gradient magnitude* 2 dan *gradient direction* 80° akan masuk kedalam *bin* 80° ditunjukkan dengan panah biru, sedangkan *pixel* yang memiliki nilai *gradient magnitude* 4 dan *gradient direction* 10° akan masuk kedalam *bin* 0° dan 20° ditunjukkan dengan panah merah seperti pada Gambar 2.13.

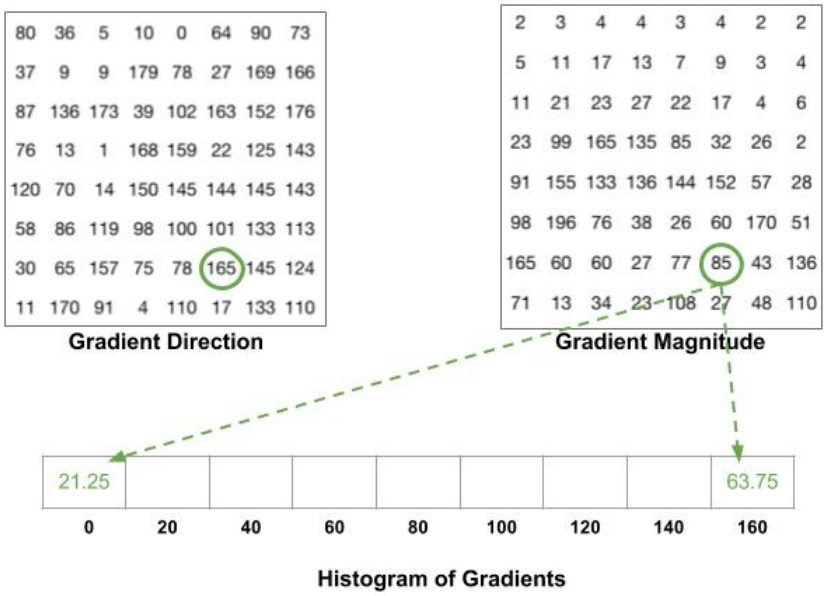


Gambar 2.13 *Bin Histogram of Gradients*

Sumber: (Mallick, 2018)

Perlu diperhatikan dalam memasukkan nilai pada *bin*, jika nilai *direction* lebih dari 160° (160°-180°). Sebagai contoh *pixel* dengan nilai *gradient direction* 165°

akan berkontribusi kedalam *bin* 0° dan *bin* 160° seperti pada Gambar 2.14, maka digunakan Persamaan 2.6.



Gambar 2.14 Nilai *bin Histogram of Gradients* lebih dari 160°

Sumber: (Mallick, 2018)

$$\text{Vote} = \frac{\Delta R_i}{H} \times G_i \text{ dan } \frac{1-\Delta R_i}{H} \times G_i \quad (2.6)$$

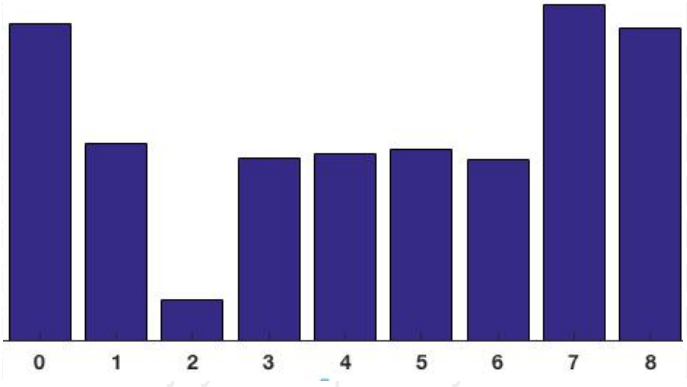
Keterangan Persamaan 2.6:

$H =$ Rentang nilai sudut.

$G_i =$ Nilai $Gradient$.

$\Delta R_i =$ Batas atas signed $gradient$ – Nilai orientasi $gradient$.

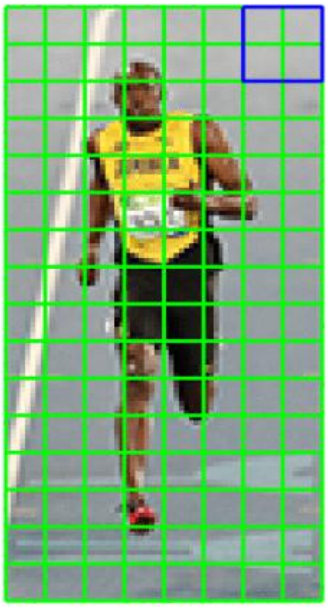
Gabungan dari tiap *pixel* pada sel 8x8 akan ditambahkan untuk membuat *histogram* 9 *bin*. *Histogram* pada tiap sel akan terlihat seperti pada Gambar 2.15.



Gambar 2.15 Histogram pada tiap sel

Sumber: (Mallick, 2018)

Setelah dilakukan perhitungan *histogram* pada semua sel, selanjutnya akan digabungkan tiap sel untuk membentuk sebuah blok 16x16 *pixel* seperti pada Gambar 2.16 yang terdiri dari 4 *histogram*.



Gambar 2.16 Blok 16x16 yang terdiri dari 4 sel

Sumber: (Mallick, 2018)

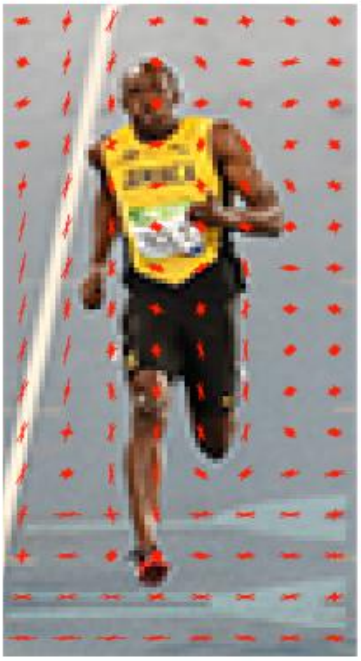
Setelah itu blok tersebut dilakukan normalisasi L2 dengan Persamaan 2.7. Sehingga didapatkan 36 x 1 elemen vektor tiap *window*, dan proses tersebut berulang sampai semua sel ternormalisasi.

$$\|x\|_2 := \sqrt{x^2{}_1 + \ldots + x^2{}_n} \quad (2.7)$$

Keterangan Persamaan 2.7:

x = fitur *histogram* vektor

Untuk mendapatkan vektor fitur akhir keseluruhan gambar, vektor 36 x 1 digabunggkan untuk membentuk vektor besar. Pada gambar keseluruhan kita memiliki 7 *horizontal* dan 15 *vertical* posisi *window* sehingga menghasilkan 1 x 15 = 105 posisi. Tiap 16 x 16 blok direpresentasikan dengan 36 x 1 vektor fitur. Jadi ketika digabungkan semua kita memiliki 36 x 105 = 3780 vektor dimensi. Hasil visualisasi akhir dari fitur HOG terlihat seperti pada Gambar 2.17.

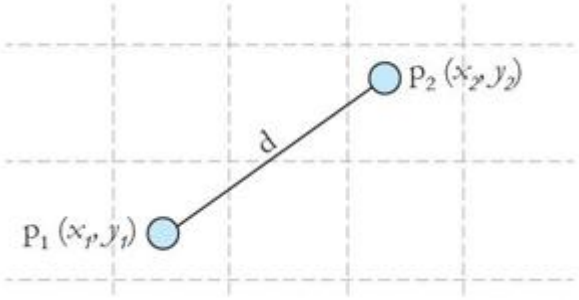


Gambar 2.17 Visualisasi fitur *histogram of oriented gradients*

Sumber: (Mallick, 2018)

2.2.12 K-*Nearest Neighbors*

K-*Nearest Neighbors* (k-NN) termasuk pada kelompok algoritma pembelajaran *instance-based learning* yang membandingkan data baru terhadap data latih yang disimpan pada memori. K-NN melakukan klasifikasi dengan memproyeksikan data baru terhadap sekumpulan data pembelajaran pada ruang berdimensi banyak. Ruang ini dibagi menjadi beberapa dimensi yang merepresentasikan fitur dari data pembelajaran. Setiap data pembelajaran direpresentasikan dengan titik data pada ruang berdimensi banyak. Selanjutnya klasifikasi data baru dilakukan dengan menghitung jarak terdekat antara titik data baru dengan data pembelajaran, misal nilai k = 3 maka sebanyak 3 titik data terdekat dengan data baru dilakukan *votting*, kelas dengan jumlah *votting* terbanyak/yang paling dominan dapat merepresentasikan kelas dari data baru (Advernesia, 2019). Teknik dalam menentukan titik data tetangga terdekat secara umum dilakukan dengan menggunakan formula jarak *Euclidean* seperti pada Persamaan 2.8. Pada Gambar 2.18 menunjukkan contoh jarak *Euclidean* pada dua titik koordinat.



Gambar 2.18 Jarak *Euclidean*

Sumber: (ArcGIS, 2019)

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} \quad (2.8)$$

Keterangan Persamaan 2.8:

$d =$ jarak $Euclidean$.
$x =$ nilai sumbu x.
$y =$ nilai sumbu y.

2.2.13 Raspberry Pi 3

Raspberry Pi 3 dapat dilihat pada Gambar 2.19 merupakan komputer mini ukuran saku yang memiliki *port input* dan *output* seperti pada *micro controller*. Raspberry Pi juga dibekali *port* HDMI untuk koneksi *display* berupa televisi atau *monitor*, dan terdapat *port* USB untuk koneksi *keyboard*, *mouse*, dan lain-lain. Raspberry Pi dapat berjalan di berbagai sistem operasi seperti contohnya Raspbian.

Gambar 2.19 Raspberry Pi 3 Model B

Sumber: (Raspberry Pi Foundation, 2019)

2.2.14 Kamera Logitech C270

Kamera Logitech C270 dapat dilihat pada Gambar 2.20 merupakan kamera berjenis *webcam* yang dapat digunakan untuk mengambil foto dengan resolusi 5

mega pixels, dan membuat video dengan resolusi VGA. Logitech C270 dapat beroperasi di berbagai macam sistem operasi seperti Windows, Linux, dan Mac OS.

Gambar 2.20 Logitech C270

Sumber: (Logitech, 2019)

2.2.15 LCD Raspberry Pi 3.5 Inch

Layar LCD (*Liquid Crystal Display*) dapat dilihat pada Gambar 2.21 digunakan untuk menampilkan tulisan atau gambar yang terbentuk dari susunan *pixel* di dalamnya. Kelebihan LCD yaitu hemat daya dan memiliki ukuran dimensi yang relatif kecil.

Gambar 2.21 LCD Raspberry Pi 3.5 Inch

Sumber: (Waveshare, 2019)

BAB 3 METODOLOGI

Bab ini menjelaskan secara sistematik metode-metode yang digunakan untuk menyelesaikan masalah dalam penelitian ini, seperti tipe penelitian, strategi dan rancangan penelitian yang meliputi metode secara umum, objek penelitian, lokasi penelitian, metode analisis kebutuhan, metode perancangan sistem, metode implementasi sistem, metode pengumpulan data, dan metode analisis data,.

3.1 Tipe Penelitian

Tipe penelitian yang digunakan disini yaitu implementatif berupa perancangan sistem yang digunakan untuk mengklasifikasi kendaraan yang melewati jalan Tol sesuai golongan kendaraan yang berlaku pada jalan Tol.

3.2 Strategi dan Perancangan Sistem

3.2.1. Metode Secara Umum

Metode yang digunakan dalam penelitian ini yaitu kualitatif, dimana penelitian dilaksanakan secara sistematis. Dalam penyampaian dan analisa hasil penelitian disampaikan dalam bentuk angka pada tabel maupun grafik.

3.2.2. Objek Penelitian

Dalam penelitian ini yang digunakan sebagai objek penelitian yaitu kendaraan yang melewati jalan Tol, dimana kendaraan tersebut akan dilakukan klasifikasi golongan sesuai golongan kendaraan yang berlaku pada jalan Tol

3.2.3. Lokasi Penelitian

Penelitian ini dilakukan pada jalan Tol Pandaan – Malang.

3.2.4. Metode Analisis Kebutuhan

Dalam penelitian ini kebutuhan yang diperlukan oleh sistem meliputi, kebutuhan perangkat keras yang digunakan, kebutuhan perangkat lunak yang digunakan, kebutuhan fungsional, dan kebutuhan non fungsional. Pada kebutuhan perangkat keras akan disebutkan dan dijelaskan apa saja perangkat keras yang digunakan dalam pembuatan sistem. Begitupun pada kebutuhan perangkat lunak akan dijelaskan apa saja perangkat lunak yang digunakan dalam pembuatan sistem pada penelitian ini. Pada kebutuhan fungsional menjelaskan kebutuhan yang harus dicapai oleh sistem yang sudah dirancang untuk dapat mencapai hasil yang diinginkan. Sedangkan pada kebutuhan non fungsional menjelaskan batasan sistem pada pengimplementasiannya.

3.2.5. Metode Perancangan Sistem

Perancangan sistem nantinya berdasarkan analisis kebutuhan yang sudah dijelaskan sebelumnya. Di dalamnya terbagi menjadi dua *sub* bab yaitu,

perancangan perangkat keras, dan perancangan perangkat lunak. Serta akan dijelaskan pula alur dari sistem secara keseluruhan.

3.2.6. Metode Implementasi Sistem

Implementasi sistem nantinya sesuai dengan rancangan sistem yang sudah dibuat sebelumnya. Implementasi sistem dibagi menjadi dua *sub* bab, yaitu implementasi perangkat keras, dan implementasi perangkat lunak. Pada implementasi perangkat keras dijelaskan bagaimana rangkaian perangkat keras diimplementasikan. Sedangkan pada implementasi perangkat lunak dijelaskan alur jalannya kode program beserta penjelasannya.

3.2.7. Metode Pengumpulan Data

Data yang digunakan dalam penelitian ini terdiri dari data latih dan data uji. Untuk data latih berupa foto tampak samping kendaraan golongan 1 sampai golongan 5 yang didapatkan dari internet dan pengambilan foto langsung oleh peneliti sebanyak total 200 data yang tiap golongan kendaraan terdiri dari 40 data. Sedangkan untuk data uji berupa video tampak samping kendaraan golongan 1 sampai golongan 5 yang melewati jalan Tol, data uji diambil dengan menempatkan kamera pada sisi jalan Tol untuk kemudian mengambil video kendaraan yang lewat, data uji diambil sebanyak total 25 data dengan masing-masing golongan sebanyak 5 data.

3.2.8. Metode Analisis Data

Metode yang digunakan dalam menganalisa data diawali dengan melakukan pengujian terhadap sistem yang sudah diimplementasi berdasarkan parameter apa saja yang ingin dicapai. Adapun parameter yang ingin dicapai pada penelitian ini yaitu:

1. Pengujian untuk mencari nilai parameter *k* terbaik dalam melakukan klasifikasi kendaraan.
2. Pengujian pengolahan data citra dari tangkapan kamera apakah sudah dapat melakukan klasifikasi terhadap kendaraan.
3. Pengujian tingkat akurasi sistem dalam melakukan klasifikasi kendaraan dari video tangkapan kamera.

Setelah dilakukan pengujian sistem dengan mengikuti prosedur yang sudah ditentukan, kemudian hasil dari pengujian dilakukan analisa untuk menarik kesimpulan dan pemberian saran dari penelitian yang sudah dilakukan untuk pengembangan sistem selanjutnya.

BAB 4 REKAYASA KEBUTUHAN

Rekayasa kebutuhan menjelaskan tentang kebutuhan apa saja yang diperlukan oleh sistem, meliputi kebutuhan perangkat keras, kebutuhan perangkat lunak, kebutuhan fungsional sistem, dan kebutuhan non fungsional sistem. Pada bab ini juga menjelaskan gambaran umum dari sistem.

4.1 Gambaran Umum Sistem

Sistem klasifikasi golongan kendaraan berdasarkan fitur *Histogram of Oriented Gradients* menggunakan metode k-*Nearest Neighbors* merupakan sistem yang digunakan untuk mengklasifikasi jenis kendaraan sesuai golongan pada jalan tol. Sistem mengklasifikasi jenis kendaraan dengan melihat bentuk objek melalui proses ekstraksi nilai fitur berupa arah distribusi intensitas *gradient* dari foto tampak samping kendaraan. Setelah nilai fitur dari objek didapatkan kemudian dilakukan proses klasifikasi dengan menggunakan metode k-*Nearest Neighbors* untuk menentukan kelas dari data baru yang diujikan dengan menghitung jarak terdekat antara titik data baru dengan titik data latih kemudian dicari sebanyak nilai *k* tetangga terdekat untuk dilakukan *votting*, kelas yang mendapat *votting* terbanyak merepresentasikan kelas dari data baru yang diujikan.

4.2 Analisis Kebutuhan Sistem

Analisis kebutuhan sistem bertujuan untuk mengetahui kebutuhan apa saja yang diperlukan dalam sistem ini. Adapun kebutuhan dalam sistem ini dibagi menjadi tiga, yaitu kebutuhan perangkat keras, kebutuhan perangkat lunak,dan kebutuhan fungsional.

4.2.1 Kebutuhan Perangkat Keras

Untuk pengimplementasian sistem ini dibutuhkan perangkat keras berupa:

1. Raspberry Pi 3
   Digunakan untuk memproses masukan dari kamera berupa video yang kemudian menghasilkan klasifikasi golongan dari kendaraan. Hasil klasifikasi dari sistem akan ditampilkan melalui *monitor* yang terhubung pada *port* HDMI.
2. Kamera Logitech C270
   Digunakan untuk mengambil citra berupa video yang nantinya digunakan sebagai masukan untuk diproses pada Raspberry Pi 3.

3. LCD Raspberry Pi 3,5 *Inch*
   Digunakan untuk menampilkan hasil keluaran dari sistem berupa tangkapan video dari kamera dan juga untuk menampilkan hasil klasifikasi dari golongan kendaraan.

4.2.2 Kebutuhan Perangkat Lunak

Dalam perancangan dari sistem ini terdapat beberapa *library* yang diperlukan dalam pembuatan perangkat lunak ini. Diantaranya sebagai berikut:

1. Raspbian Stretch
   Sebagai sistem operasi dari *board* Raspberry Pi 3.
2. Python 3
   Dalam penulisan kode program dari sistem ini menggunakan bahasa pemrograman python 3.
3. *Library* OpenCV
   Digunakan *library* OpenCV untuk menangani pemrosesan gambar.
4. *Library* Scikit-Image
   Digunakan *library* Scikit-Image untuk menangani pemrosesan gambar.
5. *Library* Scikit-Learn
   Digunakan *library* Scikit-Learn untuk menangani klasifikasi data.
6. *Library* Imutils
   Digunakan *library* Imutils untuk menangani pemrosesan gambar dan manajemen direktori file.
7. *Library* Argparse
   Digunakan *library* Argparse untuk membuat perintah *switch* saat menjalankan program pada *command line*.

4.2.3 Kebutuhan Fungsional

Sistem ini harus memenuhi kebutuhan fungsional yang harus dapat diselesaikan. Kebutuhan fungsional yang dimaksud sebagai berikut:

1. Sistem mampu mengambil citra berupa video dari tangkapan kamera secara *real time* yang kemudian digunakan sebagai masukan yang akan diproses oleh Raspberry Pi 3 untuk mendapatkan nilai fitur HOG kendaraan yang masuk dalam *frame* video, kemudian sistem melakukan klasifikasi untuk menentukan golongan dari kendaraan yang lewat, dan selanjutnya hasil dari klasifikasi

ditampillkan pada layar LCD beserta tangkapan video dari kamera secara *real time*.

4.2.4 Kebutuhan Non Fungsional

Sistem agar dapat berjalan sesuai rancangan yang sudah dibuat, terdapat batasan pada implementasinya. Diantaranya sebagai berikut:

1. Kamera yang digunakan untuk mengambil citra berupa video harus dalam posisi diam dan tidak berubah posisi.
2. Pengambilan citra berupa video dari kendaraan yang akan diklasifikasi harus dilakukan dari samping sejajar dengan kendaraan mencakup seluruh badan kendaraan dan dengan pencahayaan yang cukup.

BAB 5 PERANCANGAN DAN IMPLEMENTASI

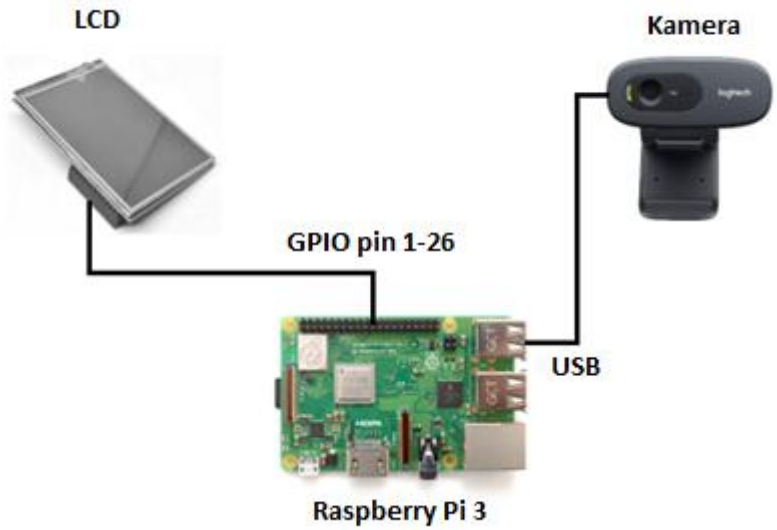
Bab ini menjelaskan tentang perancangan dan implementasi sistem. Perancangan akan menjelaskan alur program dan metode yang digunakan. Kemudian program dan metode yang sudah dirancang diimplementasikan pada tahap implementasi.

5.1 Perancangan Sistem

Perancangan sistem mejelaskan rancangan dari sistem yang akan dibuat. Perancangan sistem akan dijelaskan menjadi sub bab yaitu, perancangan perangkat keras, dan perancangan perangkat lunak.

5.1.1 Perancangan Perangkat Keras

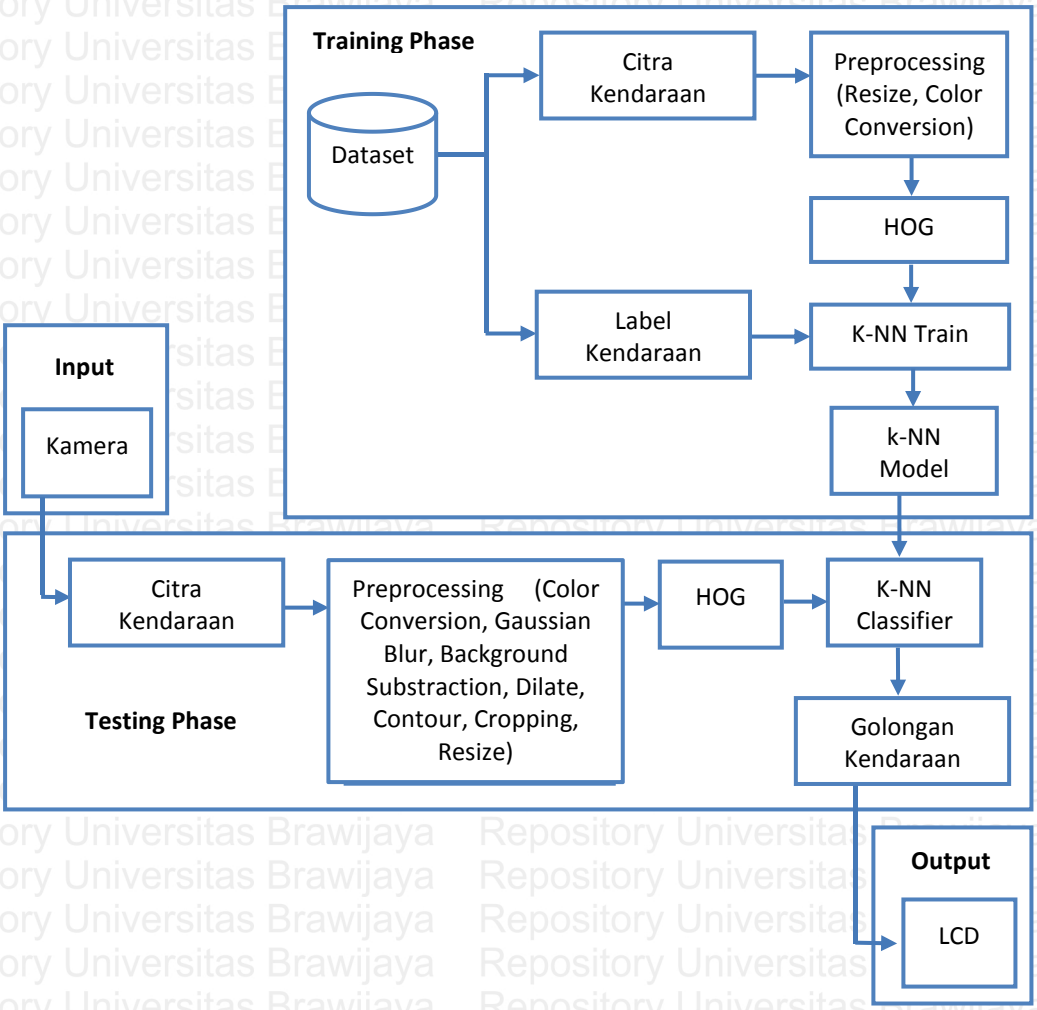
Sistem ini menggunakan tiga perangkat keras berupa kamera Logitech C207 yang digunakan sebagai penangkap citra berupa video, Raspberry Pi 3 digunakan sebagai pemroses masukan berupa citra video yang kemudian dilakukan proses klasifikasi untuk menentukan golongan dari kendaraan, dan layar LCD sebagai penampil keluaran sistem berupa video tangkapan kamera secara *real time* beserta hasil klasifikasi dari golongan kendaraan. Perancangan perangkat keras dapat dilihat pada Gambar 5.1.



Gambar 5.1 Perancangan rangkaian perangkat keras

5.1.2 Perancangan Perangkat Lunak

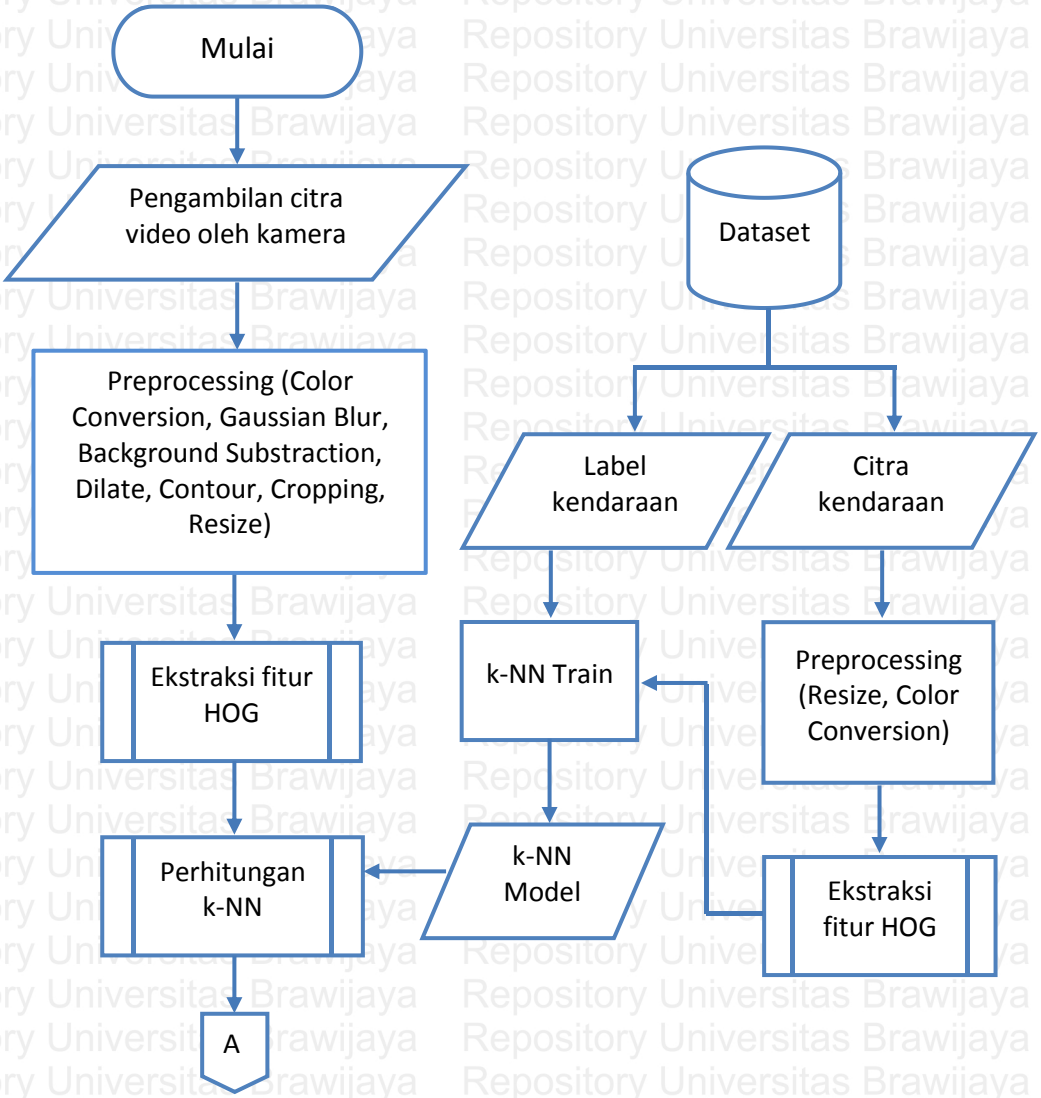
Dalam proses pengolahan citra video untuk mendapatkan hasil klasifikasi golongan kendaraan dilakukan dua tahapan proses utama, yaitu proses *training phase* dan *testing phase* di dalam program perangkat lunak. Struktur perangkat lunak dapat dilihat seperti blok diagram pada Gambar 5.2.

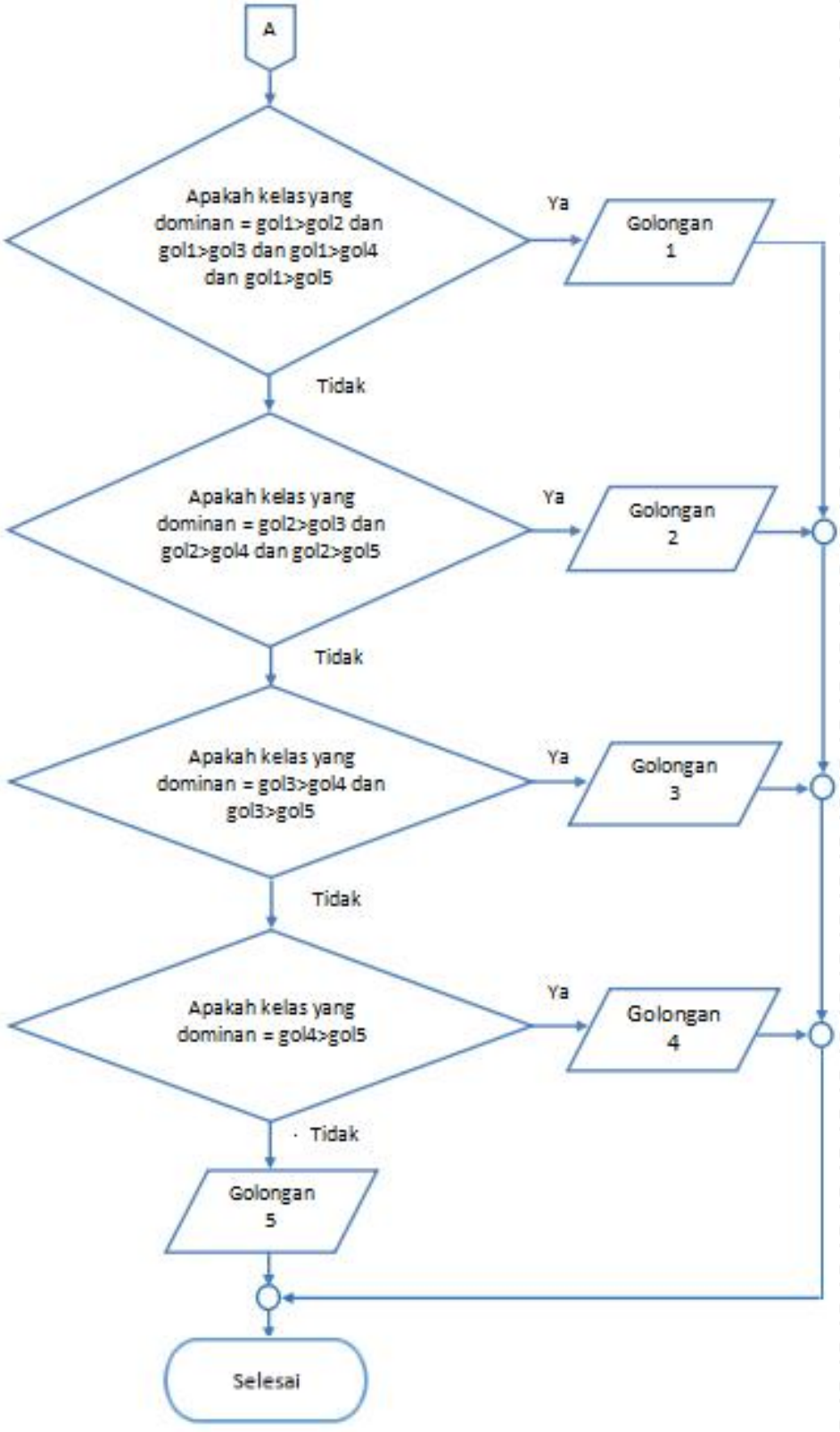


Gambar 5.2 Blok diagram perancangan perangkat lunak

Proses diawali pada *training phase*, dimana citra gambar kendaraan diambil dari *dataset* untuk dilakukan tahap *preprocessing* yaitu *resize* ukuran gambar menjadi 200 x 100 *pixel*, konversi warna dari RGB menjadi *grayscale*, setelah itu dilakukan ekstraksi fitur HOG, selanjutnya fitur HOG dilakukan proses *traning* k-NN beserta pemberian label kendaraan yang diambil dari *dataset*, keluaran dari proses *training* k-NN berupa data model dari *dataset*. Setelah itu masuk pada *testing phase* dimana citra video kendaraan diambil oleh kamera, kemudian dilakukan *preprocessing* berupa *color conversion* dari RGB menjadi *grayscale*, gaussian *blur* untuk mengurangi *noise*, *background substraction* untuk mencari perbedaan *frame background* dengan *frame foreground* yang digunakan untuk mencari objek kendaraan, dilasi untuk membuat *pixel* objek yang terdeteksi agar terlihat menyatu, *contour* untuk mengambil kontur yang terdeteksi, *cropping* untuk mengambil hanya objek kendaraan pada *frame*, *resize* untuk mengubah ukuran gambar hasil *cropping* menjadi 200 x 100 *pixel* agar seragam saat dilakukan pengolahan HOG, setelah itu nilai fitur HOG dilakukan perhitungan k-NN terhadap model k-NN untuk mencari data *k* tetangga terdekat antara data uji dengan data

latih, jika kelas yang dominan muncul pada *k* tetangga terdekat merupakan kendaraan golongan 1, maka sistem mengklasifikasi data uji sebagai kendaraan golongan 1, namun jika kelas yang dominan muncul pada *k* tetangga terdekat merupakan kendaraan golongan 2, maka sistem mengklasifikasi data uji sebagai kendaraan golongan 2, begitu seterusnya sampai pada golongan 5. Urutan pemrosesan pada perangkat lunak dapat dilihat seperti pada Gambar 5.3.





Gambar 5.3 Diagram alir perangkat lunak

a. Proses Deteksi Kendaraan Pada Video

Contoh proses deteksi kendaraan pada video:

Sebelum dilakukan proses deteksi kendaraan pada video, dilakukan inisialisasi *background* (*frame* video yang tidak terdapat kendaraan) yang digunakan untuk referensi saat melakukan *background substraction*. Sebelum inisialisasi *background*, *frame* yang akan digunakan sebagai *background* dilakukan proses *resize* menjadi lebar 500 *pixel* seperti pada Gambar 5.4.

Gambar 5.4 Citra kendaraan berukuran lebar 500 *pixel*

Selanjutnya dilakukan konversi warna dari RGB menjadi *grayscale*, bertujuan untuk efisiensi proses komputasi saat ekstraksi itur HOG. Hasil proses konversi warna *grayscale* seperti pada Gambar 5.5.

Gambar 5.5 Citra kendaraan setelah dilakukan konversi warna *grayscale*

Selanjutnya dilakukan proses *gaussian blurring* untuk mengurangi *noise* yang terdapat pada gambar. Hasil proses *gaussian blurring* terlihat seperti pada Gambar 5.6.

Gambar 5.6 Citra kendaraan setelah dilakukan proses *gaussian blurring*

Selanjutnya dilakukan proses *background substraction (absolute difference)*, yaitu dengan mengurangi *frame background* dengan *frame foreground* (*frame* video yang terdapat kendaraan). Pada proses ini jika terdapat perbedaan nilai *pixel* antara *frame background* dengan *frame foreground* pada *pixel* tersebut akan memiliki perbedaan derajat warna keabuan, semakin besar nilai perbedaan maka warna *pixel* semakin berwarna putih. Hasil proses *background substraction* seperti pada Gambar 5.7.

Gambar 5.7 Citra kendaraan setelah dilakukan proses *background substraction*

Selanjutnya dilakukan proses dilasi untuk membuat objek kendaraan yang terdeteksi terlihat menyatu, agar saat proses *bounding* pada kendaraan dapat mencakup seluruh badan kendaraan. Hasil proses dilasi seperti pada Gambar 5.8.

Gambar 5.8 Citra kendaraan setelah dilakukan proses dilasi

Kemudian citra kendaraan dilakukan proses *bounding* untuk menyeleksi *region of interest* berupa badan kendaraan. Hasil proses *bounding* pada citra kendaraan seperti pada Gambar 5.9.

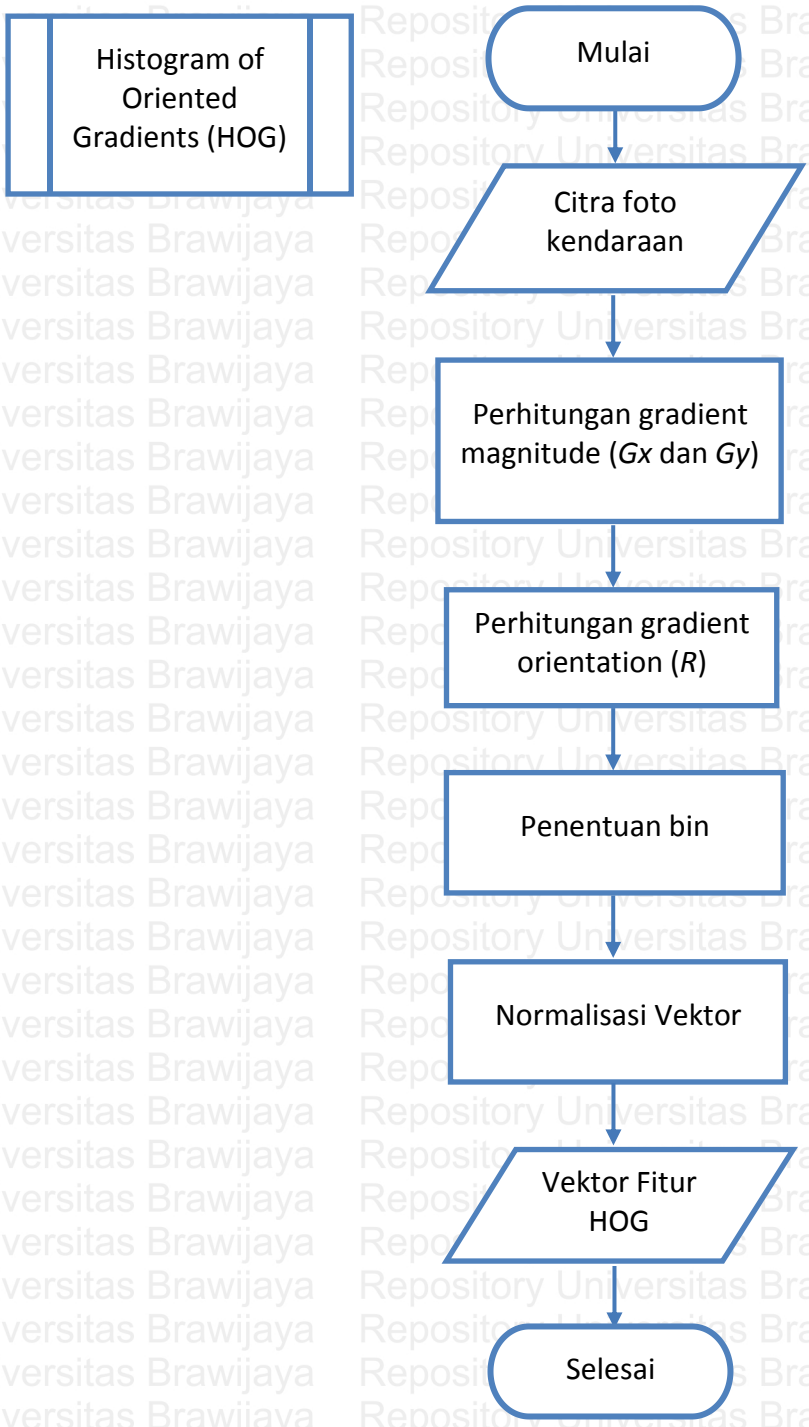
Gambar 5.9 Citra kendaraan setelah dilakukan *bounding*

Setelah itu dilakukan *cropping* pada area dalam *bounding* untuk mendapatkan hanya citra dari badan kendaran. Hasil *cropping* seperti pada Gambar 5.10.

Gambar 5.10 Citra kendaraan setelah dilakukan *cropping*

b. Proses *Histogram of Oriented Gradients*

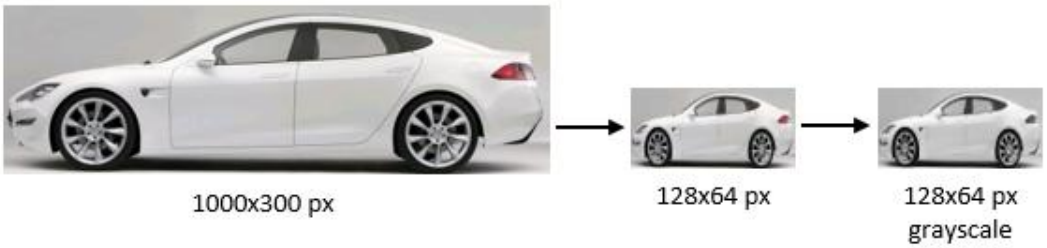
Diagram alir dari proses ekstraksi fitur *histogram of oriented gradients* ditunjukkan pada Gambar 5.11.



Gambar 5.11 Diagram alir *histogram of oriented gradients*

Contoh proses manualisasi perhitungan fitur *histogram of oriented gradients*:

Sebelum dilakukan ekstraksi fitur HOG citra kendaraan dilakukan proses *resize* menjadi ukuran 200 x 100 *pixel*, dan dilakukuan konversi warna dari RGB menjadi *grayscale* seperti pada Gambar 5.12.



Gambar 5.12 Hasil *resize* dan konversi warna terhadap citra kendaraan

Setelah dilakukan *resize* dan konversi warna maka didapatkan citra kendaraan seperti pada Gambar 5.13.

Gambar 5.13 Citra kendaraan *grayscale* 128 x 64 *pixel*

Dari citra kendaraan seperti pada gambar 5.14 didapatkan cuplikan matriks seperti pada Tabel 5.1.

Tabel 5.1 Cuplikan matriks dari citra kendaraan

24	102	31	29	49	66	61	63
62	26	29	34	48	32	48	44
49	25	33	49	84	115	56	52
40	24	37	33	71	67	41	47
27	24	24	78	42	33	25	23
22	95	26	30	36	37	95	87
64	53	29	34	45	41	50	45
24	109	33	58	23	30	39	36

Setelah itu dilakukan perhitungan *gradient magnitude* setiap *pixel* pada tiap sel pada kedua arah sumbu *horizontal* dan *vertical* menggunakan *kernel filter* operator Sobel dengan cara, matriks citra kendaraan dikonvolusikan dengan matriks operator Sobel sumbu *horizontal (Gx)* dan *vertical (Gy)* seperti pada persamaan 2.4. Berikut perhitungan konvolusi cuplikan matriks citra kendaraan berukuran 4 x 4 dengan matriks operator Sobel sumbu *horizontal* (*Gx*):

$$\begin{bmatrix}24 & 102 & 31 & 29\\62 & 26 & 29 & 34\\49 & 25 & 33 & 49\\40 & 24 & 37 & 33\end{bmatrix} * \begin{bmatrix}-1 & 0 & 1\\-2 & 0 & 2\\-1 & 0 & 1\end{bmatrix}$$

$$(1,1) = 24(-1) + 102(0) + 31(1) + 62(-2) + 26(0) + 29(2) + 49(-1) + 25(0) + 33(1) = 75$$

$$(1,2) = 102(-1) + 31(0) + 29(1) + 26(-2) + 29(0) + 34(2) + 25(-1) + 33(0) + 49(1) = 33$$

$$(2,1) = 62(-1) + 26(0) + 29(1) + 49(-2) + 25(0) + 33(2) + 40(-1) + 24(0) + 37(1) = 68$$

$$(2,2) = 26(-1) + 29(0) + 34(1) + 25(-2) + 33(0) + 49(2) + 24(-1) + 37(0) + 33(1) = 66$$

Maka didapatkan matriks sumbu *horizontal* (*Gx*) yaitu:

$$\begin{bmatrix}75 & 33\\68 & 66\end{bmatrix}$$

Selanjutnya perhitungan konvolusi cuplikan matriks citra kendaraan berukuran 4 x 4 dengan matriks operator Sobel sumbu *vertical* (*Gy*):

$$\begin{bmatrix}24 & 102 & 31 & 29\\62 & 26 & 29 & 34\\49 & 25 & 33 & 49\\40 & 24 & 37 & 33\end{bmatrix} * \begin{bmatrix}1 & 2 & 1\\0 & 0 & 0\\-1 & -2 & -1\end{bmatrix}$$

$$(1,1) = 24(1) + 102(2) + 31(1) + 62(0) + 26(0) + 29(0) + 49(-1) + 25(-2) + 33(-1) = 127$$

$$(1,2) = 102(1) + 31(2) + 29(1) + 26(0) + 29(0) + 34(0) + 25(-1) + 33(-2) + 49(-1) = 53$$

$$(2,1) = 62(1) + 26(2) + 29(1) + 49(0) + 25(0) + 33(0) + 40(-1) + 24(-2) + 37(-1) = 18$$

$$(2,2) = 26(1) + 29(2) + 34(1) + 25(0) + 33(0) + 49(0) + 24(-1) + 37(-2) + 33(-1) = 13$$

Maka didapatkan matriks sumbu *horizontal* (*Gy*) yaitu:

$$\begin{bmatrix}127 & 53\\18 & 13\end{bmatrix}$$

Setelah didapatkan nilai masing-masing sumbu *x* dan *y* pada tiap *pixel*, selanjutnya dilakukan perhitungan untuk mencari *gradient magnitude* menggunakan persamaan 2.5. Berikut perhitungan untuk mencari *gradient magnitude*:

$$g(1,1) = \sqrt{(127)^2 + (75)^2} = 147$$

$$g(1,2) = \sqrt{(53)^2 + (33)^2} = 91$$

$$g(2,1) = \sqrt{(18)^2 + (68)^2} = 70$$

$$g(2,2) = \sqrt{(13)^2 + (66)^2} = 67$$

Maka didapatkan matriks *gradient magnitude(G)* yaitu:

$$\begin{bmatrix} 147 & 91 \\ 70 & 67 \end{bmatrix}$$

Selanjutnya dilakukan juga perhitungan untuk mencari *gradient orientation (R)* dengan menggunakan persamaan 2.6. Berikut perhitungan untuk menentukan gradient orientation:

$$R(1,1) = arctan\frac{75}{127} = 59$$

$$R(1,1) = arctan\frac{53}{33} = 58$$

$$R(1,1) = arctan\frac{18}{68} = 14$$

$$R(1,1) = arctan\frac{13}{66} = 11$$

Maka didapatkan matriks *gradient orientation (R)* yaitu:

$$\begin{bmatrix} 59 & 58 \\ 14 & 11 \end{bmatrix}$$

Langkah selanjutnya melakukan proses *orientation binning* pada tiap *pixel* didalam sel menggunakan persamaan 2.7, dengan menentukan sebanyak 9 *bin* dari 180° maka diperoleh rentang sebesar 20°. Pada dasarnya *histogram* merupakan vektor atau sebuah *array* dari 9 *bin* yang tiap *bin* bertanggung jawab terhadap 0°, 20°, 40°, 60°, 80°, 100°, 120°, 140°, 160°. Berikut hasil dari dari tahap *orientation binning* terhadap matriks *gradient magnitude* $[g]$ dan matriks *gradient orientation* $[\theta]$:

$$Bin\ (1,1) = \frac{(60-59)}{20} * 147 = 7.35; Bin\ (1,1) = \frac{(59-40)}{20} * 147 = 139.65$$

$$Bin\ (1,2) = \frac{(60-58)}{20} * 91 = 9.1; Bin\ (1,2) = \frac{(58-40)}{20} * 91 = 81.9$$

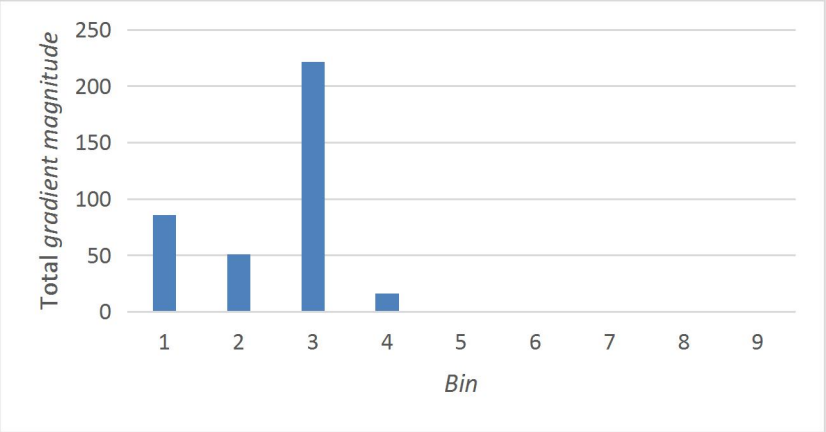
$$Bin\ (2,1) = \frac{(20-14)}{20} * 70 = 21; Bin\ (2,1) = \frac{(14-0)}{20} * 70 = 49$$

$$Bin\ (2,2) = \frac{(20-11)}{20} * 67 = 30.15; Bin\ (2,2) = \frac{(11-0)}{20} * 67 = 36.85$$

Pada Tabel 5.2 dan grafik pada Gambar 5.14 dapat dilihat hasil dari tahap *orientation binning* terhadap matriks *gradient magnitude* $[g]$ dan matriks *gradient orientation* $[\theta]$.

Tabel 5.2 Hasil *orientation binning* terhadap matriks *gradient magnitude* $[g]$ dan matriks *gradient orientation* $[\theta]$.

36.85	30.15	81.9	9.1					
49	21	139.65	7.35					
0	20	40	60	80	100	120	140	160



Gambar 5.14 Grafik *histogram of oriented gradients* dari matriks *gradient magnitude* $[g]$ dan matriks *gradient orientation* $[\theta]$.

Selanjutnya dilakukan proses normalisasi terhadap fitur HOG tiap sel pada blok dengan menggunakan Persamaan 2.8. Berikut perhitungan normalisasi untuk fitur HOG pada tiap sel:

$$\|x\|_2 := \sqrt{85.85^2 + 51.16^2 + 221.55^2 + 16.45^2} = 243.6$$

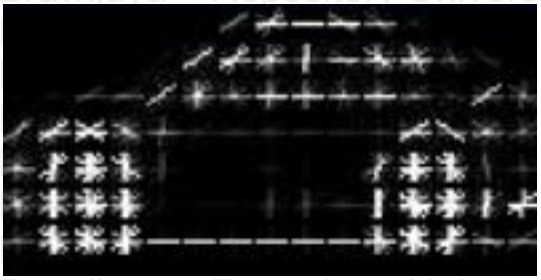
Kemudian dilakukan pembagian tiap elemen pada vektor dengan 243.6, maka didapatkan hasil sebagai berikut:

$$\left[\left(\frac{85.85}{243.6}\right), \left(\frac{52.16}{243.6}\right), \left(\frac{221.55}{243.6}\right), \left(\frac{16.45}{243.6}\right)\right] = [0.35, 0.21, 0.9, 0.06]$$

Untuk mendapatkan fitur vektor keseluruhan gambar, vektor 36 x 1 per blok digabungkan menjadi dari keseluruhan blok, terdapat 15 posisi horizontal, dan 7 posisi vertical dari blok, tiap blok terdiri dari 36 x 1 vektor, sehingga menghasilkan vektor akhir sejumlah:

$$7 * 15 * 36 = 3780 \text{ vektor}$$

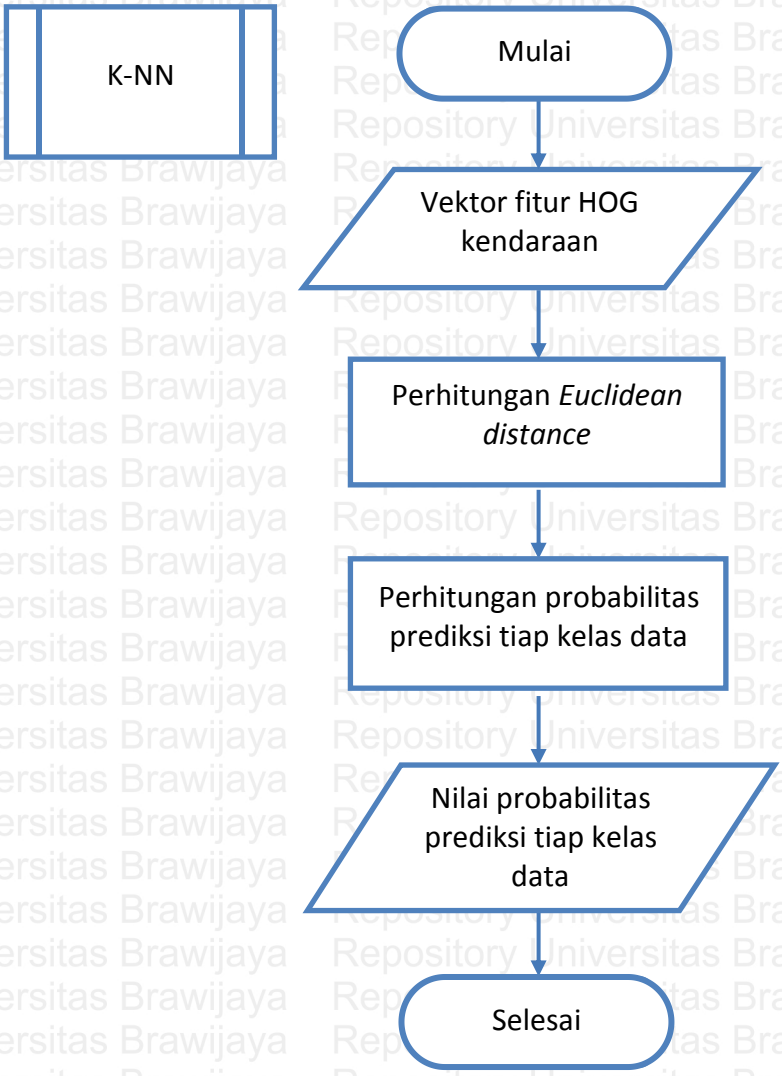
Hasil akhir dari visualisasi fitur HOG kendaraan terlihat seperti pada Gambar 5.15.



Gambar 5.15 Visualisasi fitur HOG kendaraan

c. Proses k-*Nearest Neighbors*

Diagram alir dari proses k-*nearest neighbors* dapat ditunjukkan seperti pada Gambar 5.16.



Gambar 5.16 Diagram alir proses k-NN

Contoh proses manualisasi perhitungan k-*nearest neighbors*:

Pada proses k-*nearest neighbors* vektor fitur HOG dilakukan perhitungan *Euclidean distance* menggunakan Persamaan 2.8 untuk mengetahui jarak terdekat antara data uji dengan data latih. Berikut contoh perhitungan *Euclidean distance* pada 5 data latih:

Tabel 5.3 Nilai vektor fitur HOG dari 5 kendaraan

No.	Jenis Golongan	Nilai vektor fitur HOG
1	Golongan 1	0.05, 0.09, 0.12, ... 0.008
2	Golongan 4	0.004, 0.0004, 0.001, ... 0.0004
3	Golongan 3	0.04, 0.01, 0.02, ... 0.002
4	Golongan 2	0.06, 0.01, 0.04, ... 0.02
5	Golongan 5	0.02, 0.01, 0.02, ... 0.02

$$d_1 = \sqrt{(0.05-0.35)^2+(0.09-0.21)^2+(0.12-0.9)^2+...(0.008-0.03)^2} = 0{,}278$$

$$d_2 = \sqrt{(0.004-0.35)^2+(0.0004-0.21)^2+(0.001-0.9)^2+...(0.0004-0.03)^2} = 0{,}594$$

$$d_3 = \sqrt{(0.04-0.35)^2+(0.01-0.21)^2+(0.02-0.9)^2+...(0.002-0.03)^2} = 0{,}315$$

$$d_4 = \sqrt{(0.06-0.35)^2+(0.01-0.21)^2+(0.04-0.9)^2+...(0.02-0.03)^2} = 0{,}224$$

$$d_5 = \sqrt{(0.02-0.35)^2+(0.01-0.21)^2+(0.02-0.9)^2+...(0.02-0.03)^2} = 0{,}653$$

Dari perhitungan jarak Euclidean antara data uji dengan data latih yang sudah dilakukan, dengan menerapkan nilai $k = 1$ maka diambil sebanyak 1 data yang memiliki jarak *Euclidean* terdekat dengan data uji. Karena nilai $k = 1$ maka 1 data terdekat berkontribusi sepenuhnya untuk menentukan prediksi nama kelas dari data uji. Dari contoh perhitungan jarak Euclidean terhadap 5 data latih pada Tabel 5.3 maka hasil dari nilai probabilitas kontribusi tiap kelas seperti pada Tabel 5.4.

Tabel 5.4 Nilai probabilitas kontribusi tiap kelas

No.	Nama kelas	Nilai probabilitas kontribusi
1	Golongan 1	1
2	Golongan 4	0
3	Golongan 3	0
4	Golongan 2	0
5	Golongan 5	0

Dengan kontribusi nilai probabilitas kelas Golongan 1 sebesar 1, maka klasifikasi untuk data uji memiliki nama kelas sebagai Golongan 1.

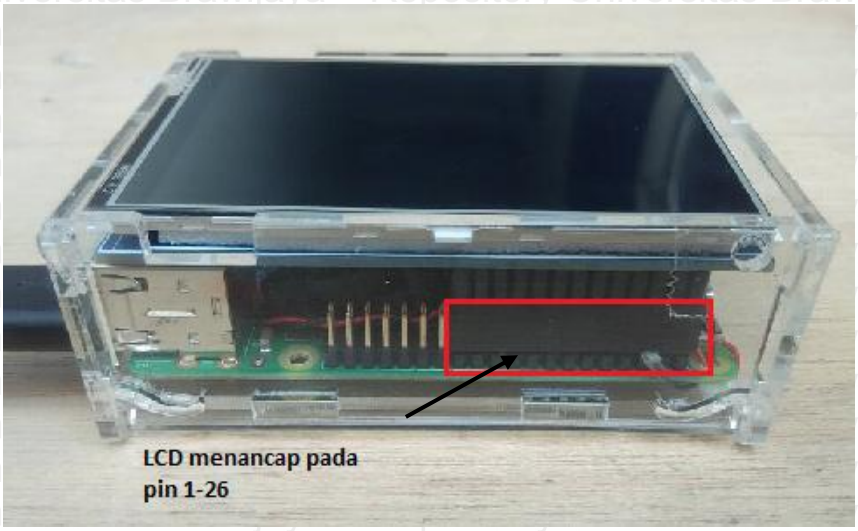
5.2 Implementasi Sistem

Implementasi sistem menjelaskan secara teknis langkah pengimplementasian sistem sesuai dengan rancangan yang sudah dibuat sebelumnya. Implementasi sistem dijelaskan menjadi dua *sub* bab, yaitu implementasi perangkat keras, dan implementasi perangkat lunak.

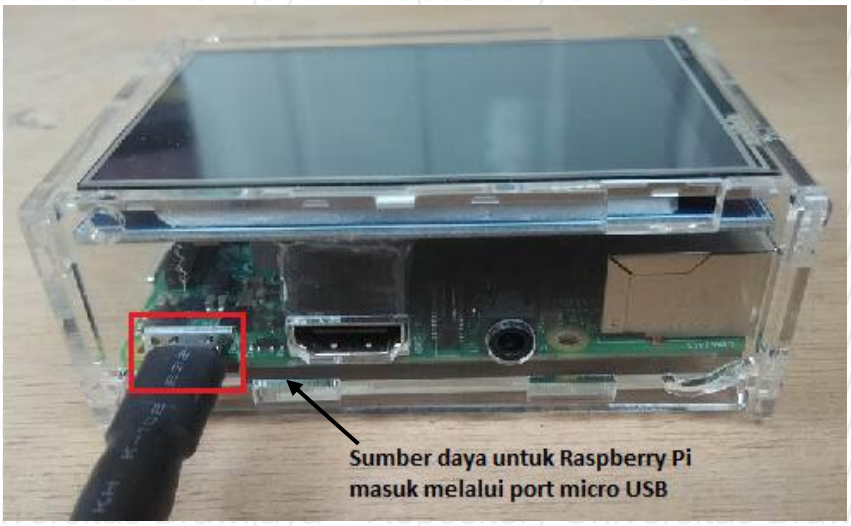
5.2.1 Implementasi Perangkat Keras

Dalam sistem ini menggunakan tiga komponen perangkat keras, yaitu kamera Logitech C270, Raspberry Pi 3, dan layar LCD 3,5 *inch*. Pada Gambar 5.17 ditunjukkan pemasangan kamera pada *port* USB Raspberry Pi 3, dan pada Gambar 5.18 ditunjukkan pemasangan layar LCD pada GPIO pin 1-26 Raspberry Pi 3. Untuk sumber daya Raspberry Pi 3 menggunakan tegangan arus 5V 2A pada *port micro* USB seperti pada Gambar 5.19.

Gambar 5.17 Pemasangan kamera pada *port* USB Raspberry Pi 3



Gambar 5.18 Pemasangan LCD pada *pin* GPIO 1 - 26 Raspberry Pi 3



Gambar 5.19 Pemasangan sumber daya Raspberry Pi 3 pada *port micro* USB

5.2.2 Implementasi Perangkat Lunak

Implementasi yang dilakukan yaitu menerapkan metode-metode yang sudah dirancang ke dalam bentuk kode program. Implementasi akan dijelaskan menjadi tiga bagian yaitu, implementasi proses ekstraksi fitur HOG dan *training model* k-NN, implementasi *preprocessing* citra, implementasi proses ekstraksi fitur HOG dan klasifikasi k-NN.

a. Implementasi Proses Ektraksi Fitur HOG dan *Training Model* k-NN

Algoritma 1: Proses Ekstraksi Fitur HOG dan *Training* Model k-NN

```
1  for imagePath in paths.list_images(args["training"]):
2        # extract class of the vehicle
3        make = imagePath.split("/")[-2]
```

```
4         # load the image, convert it to grayscale
5         image = cv2.imread(imagePath)
6         gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
7         gray = cv2.resize(gray, (128, 64))
8         # extract Histogram of Oriented Gradients from the vehicle
9         H = feature.hog(gray, orientations=9, pixels_per_cell=(8,
10  8), cells_per_block=(2, 2), transform_sqrt=True, block_norm="L2")
11        # update the data and labels
12        data.append(H)
13        labels.append(make)
14        # "train" the nearest neighbors classifier
15        model = KNeighborsClassifier(n_neighbors=1)
16        model.fit(data, labels)
```

Pada Algoritma 1 merupakan proses *training* model k-NN dengan membaca data citra dan label kendaraan pada *dataset*. Program diawali dengan melakukan pengulangan untuk membaca tiap gambar pada *folder* data latih, dan membaca nama *folder* data latih tiap golongan untuk memberi label pada gambar. Kemudian dilakukan konversi warna pada gambar dari RGB menjadi *grayscale*, mengubah ukuran gambar menjadi 128 x 64 *pixel*, selanjutnya dilakukan ekstraksi fitur HOG. Setelah didapatkan nilai fitur HOG, nilai disimpan pada variabel *data* beserta label golongan ke variabel *label*. Kemudian dilakukan inisialisasi nilai k pada metode k-NN dengan nilai 1, dan dilakukan pemodelan data sesuai dengan label sebagai data latih k-NN.

b. Implementasi *Preprocessing* Citra

Algoritma 2: *Preprocessing* Citra

```
1   background = None
2   while True:
3         # Capture frame-by-frame
4         return_value, frame = video_capture.read()
5         #check for empty frames
6         if(frame is None):
7               break
8         frame = imutils.resize(frame, width=300)
9         gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
10        gray = cv2.GaussianBlur(gray, (21, 21), 0)
11        # if the background is None, initialize it
12        if background is None:
13              background = gray
14              continue
15        frameDelta = cv2.absdiff(background, gray)
```

```
16          thresh = cv2.threshold(frameDelta, 20, 255,
17    cv2.THRESH_BINARY)[1]
18          # dilate the thresholded image to fill in holes, then find
19    contours on thresholded image
20          thresh = cv2.dilate(thresh, None, iterations=10)
21          cnts = cv2.findContours(thresh.copy(), cv2.RETR_EXTERNAL,
22    cv2.CHAIN_APPROX_SIMPLE)
23          cnts = imutils.grab_contours(cnts)
24          # show thresh, frameDelta on the windows
25          #cv2.imshow('Thresh', thresh)
26          #cv2.imshow('Frame Delta', frameDelta)
27          # loop over the contours
28          for c in cnts:
29                # if the contour is too small, ignore it
30                if cv2.contourArea(c) < args["min_area"]:
31                      continue
32                # compute the bounding box for the contour, draw it
33    on the frame, and update the text
34                (x, y, w, h) = cv2.boundingRect(c)
35                cropped = frame[y+10:y+h-10, x+10:x+w-10]
36                carB = cv2.resize(cropped, (128, 64))
```

Pada Algoritma 2, merupakan *preprocessing* citra sebelum dilakukan ekstraksi HOG. Program diawali dengan deklarasi variabel *background*, dan menggunakan pengulangan untuk memproses tiap *frame* video, *frame* diambil dari masukan kamera lalu disimpan pada variabel *frame*, kemudian frame diubah ukurannya menjadi lebar 500 *pixel*, dikonversi warnanya dari RGB menjadi *grayscale*, dilakukan proses *blur* untuk menghilangkan *noise* pada gambar. Selanjutnya dilakukan pengecekan, jika variabel *background* kosong maka diisi dengan nilai variabel *blur*, jika tidak kosong maka langsung dilewati, selanjutnya dihitung nilai *absolut difference* antara *frame* video pada variabel *background* dan variabel *blur*, lalu dikonversi warnanya dari *grayscale* menjadi *binary* dengan memberi *treshold* sebesar 20, nilai *pixel* dibawah *treshold* akan menjadi warna hitam, dan nilai di atas *treshold* menjadi warna putih, lalu dilakukan proses dilasi untuk menyatukan bagian objek yang terpisah pada gambar. Selanjutnya dicari kontur pada gambar, jika kontur yang dideteksi >= *treshold* akan dilakukan *bounding* pada area kontur dan dilakukan *cropping*, selanjutnya gambar pada variabel *cropped* diubah ukurannya menjadi 128 x 64 *pixel*, diubah warnanya dari RGB menjadi *grayscale*, lalu disimpan pada variabel *carB.*

c. Implementasi Proses Ekstraksi Fitur HOG dan Klasifikasi k-NN

Algoritma 3: Proses Ekstraksi Fitur HOG dan Klasifikasi k-NN

```
1   # extract Histogram of Oriented Gradients from the test
2   image and predict the make of the car
3   (H, hogImage) = feature.hog(carB, orientations=9,
4   pixels_per_cell=(8, 8), cells_per_block=(2, 2),
5   transform_sqrt=True,
6   block_norm="L2", visualize=True)
7   pred = model.predict(H.reshape(1, -1))[0]
8   cv2.rectangle(frame, (x+10, y+10), (x+w-10, y+h-10), [0,
9   255, 0], 2)
10  # draw the prediction ox,n the test image and display it
11  cv2.putText(frame, pred.title(), (x, y-10),
12  cv2.FONT_HERSHEY_DUPLEX, 0.8, (0, 0, 255), 2)
13  # show frame on the windows
14  cv2.imshow('Video (press Q to quit)', frame)
```

Pada Algoritma 3 merupakan proses ekstraksi fitur HOG dari citra kendaraan yang selanjutnya dilakukan proses klasifikasi k-NN. Program diawali dengan mengambil data gambar pada variabel *carB* diekstrak untuk mendapatkan fitur HOG, setelah itu dapat dilakukan prediksi pada nilai fitur HOG yang dihasilkan untuk menentukan golongan kendaraan apa. Selanjutnya objek berupa kendaraan pada *frame* video diberi *bounding* persegi dan diberi label golongan kendaraan. Setelah itu hasil dari klasifikasi beserta *bounding* persegi terhadap objek ditampilkan pada video yang ditangkap kamera secara *real time* pada layar LCD Raspberry Pi.

BAB 6 PENGUJIAN

Bab ini menjelaskan tujuan dan prosedur dalam melakukan pengujian dari sistem yang sudah dirancang dan diimpelementasi, serta ditunjukkan hasil dan analisis dari pengujian yang sudah dilakukan. Pengujian dilakukan untuk mengetahui apakah sistem telah dapat bekerja dan memenuhi kebutuhan fungsionalitas yang sudah ditentukan. Pengujian dari sistem ini meliputi pengujian apakah sistem dapat mengolah citra video kendaraan dari tampak samping kemudian diproses menggunakan Raspberry Pi 3 untuk menghasilkan klasifikasi berupa golongan kendaraan, dan menguji tingkat akurasi hasil klasifikasi dari sistem ini. Kemudian dilakukan analisis untuk mengukur keberhasilan dari sistem yang sudah dibuat.

6.1 Pengujian Untuk Mencari Nilai *k* Paling Optimal Dalam Melakukan Klasifikasi

Pengujian untuk mencari nilai *k* paling optimal dalam melakukan klasifikasi dilakukan untuk mencari parameter nilai *k* dalam algoritma k-NN yang paling optimal dalam melakukan klasifikasi untuk mendapatkan hasil akurasi terbaik sistem dalam melakukan klasifikasi golongan kendaraan.

6.1.1 Prosedur Pengujian Untuk Mencari Nilai *k* Paling Optimal Dalam Melakukan Klasifikasi

Pengujian dilakukan dengan menggunakan data latih sebanyak 200 data, tiap kelas golongan kendaraan sebanyak 40 data latih, kemudian dilakukan pengujian menggunakan data uji sebanyak 25 data, dengan tiap kelas golongan kendaraan sebanyak 5 data uji. Selanjutnya dilakukan perhitungan akurasi tiap parameter nilai *k*=1, *k*=3, *k*=5, *k*=7, *k*=9, *k*=11, *k*=13, dan *k*=15.

6.1.2 Hasil Pengujian Akurasi Sistem Dengan Nilai k berbeda

Tabel 6.1 Hasil pengujian akurasi sistem dengan nilai *k* berbeda

Nilai parameter *k*	Hasil akurasi (%)
1	**92**
3	80
5	84
7	**92**
9	88
11	76
13	76
15	84

6.1.3 Analisis Hasil Pengujian Akurasi Sistem Dengan Nilai *k Berbeda*

Dari hasil yang didapat dari pengujian, nilai *k* = 1 dan *k* = 7 mendapatkan akurasi tertinggi dengan nilai 92% dalam melakukan klasifikasi golongan kendaraan, dari hasil tersebut dipilih nilai *k* = 1 karena menghitung 1 data tetangga terdekat lebih efisien dalam sisi komputasi daripada menghitung 7 data tetangga terdekat.

6.2 Pengujian Sistem Dalam Melakukan Klasifikasi Golongan Kendaraan

Pengujian sistem dalam melakukan klasifikasi golongan kendaraan dilakukan untuk mengetahui sistem yang sudah dibuat apakah sudah dapat menghasilkan klasifikasi golongan kendaraan terhadap kendaraan tersebut.

6.2.1 Prosedur Pengujian Sistem Dalam Melakukan Klasifikasi Golongan Kendaraan

Pengujian dilakukan dengan menempatkan kamera pada sisi jalan Tol, kemudian saat program dijalankan sistem akan menangkap video dari kamera untuk kemudian diproses untuk menghasilkan klasifikasi golongan terhadap kendaraan yang lewat. Kemudian hasil klasifikasi kendaraan ditampilkan pada layar LCD Raspberry Pi 3.

6.2.2 Hasil Pengujian Sistem Dalam Melakukan Klasifikasi Golongan Kendaraan

Tabel 6.2 Hasil pengujian sistem dalam mengklasifikasi golongan kendaraan

No.	Nilai fitur HOG	Prediksi kemunculan data tiap kelas golongan kendaraan	Hasil klasifikasi golongan	Hasil tampilan pada layar
1	0.006312, 0.002621, 0.002966, 0.006695, 0.013420, 0.006645, 0.005200, 0.002665, 0.001094, ... 0.010210	Golongan 1 = 1 Golongan 2 = 0 Golongan 3 = 0 Golongan 4 = 0 Golongan 5 = 0	Golongan 1	

Tabel 6.2 Hasil pengujian sistem dalam mengklasifikasi golongan kendaraan (lanjutan)

No.	Nilai fitur HOG	Prediksi kemunculan data tiap kelas golongan kendaraan	Hasil klasifikasi golongan	Hasil tampilan pada layar
2	0.001732, 0.005307, 0.006161, 0.008654, 0.004284, 0.000375, 0.002909, 0.006608, 0.001531, ... 0.000862	Golongan 1 = 0 Golongan 2 = 1 Golongan 3 = 0 Golongan 4 = 0 Golongan 5 = 0	Golongan 2	
3	0.012219, 0.006619, 0.123790, 0.010647, 0.016836, 0.005058, 0.001763, 0.098049, 0.054057, ... 0.032019	Golongan 1 = 0 Golongan 2 = 0 Golongan 3 = 1 Golongan 4 = 0 Golongan 5 = 0	Golongan 3	
4	0.041877, 0.333808, 0.033485, 0.026387, 0.048365, 0.058765, 0.004349, 0.000738, 0.002261, ... 0.002624	Golongan 1 = 0 Golongan 2 = 0 Golongan 3 = 0 Golongan 4 = 1 Golongan 5 = 0	Golongan 4	

Tabel 6.2 Hasil pengujian sistem dalam mengklasifikasi golongan kendaraan (lanjutan)

No.	Nilai fitur HOG	Prediksi kemunculan data tiap kelas golongan kendaraan	Hasil klasifikasi golongan	Hasil tampilan pada layar
5	0.003686, 0.001825, 0.000160, 0.001239, 0.002815, 0.025561, 0.006251, 0.007659, 0.007604, ... 0.047790	Golongan 1 = 0 Golongan 2 = 0 Golongan 3 = 0 Golongan 4 = 0 Golongan 5 = 1	Golongan 5	

6.2.3 Analisis Hasil Pengujian Sistem Dalam Melakukan Klasifikasi Golongan Kendaraan

Dari hasil yang didapat dari pengujian, sistem sudah mampu dalam melakukan klasifikasi terhadap kendaraan pada rekaman video, namun hasil klasifikasi yang ditampilkan pada layar LCD memiliki *frame rate* rendah, ini dikarenakan proses deteksi objek dan klasifikasi dilakukan tiap *frame* pada video sehingga membuat proses komputasi yang berat.

6.3 Pengujian Akurasi Sistem Dalam Melakukan Klasifikasi

Pengujian akurasi sistem bertujuan untuk mengetahui seberapa besar tingkat akurasi sistem dalam melakukan klasifikasi golongan kendaraan terhadap data uji.

6.3.1 Prosedur Pengujian Akurasi Sistem Dalam Melakukan Klasifikasi

Pengujian dilakukan dengan menguji tiap kelas golongan kendaraan sejumlah 5 data uji, kemudian dari hasil klasifikasi tiap *frame* dicari kelas golongan kendaraan yang paling sering muncul, kemudian kelas golongan kendaraan yang paling sering muncul digunakan untuk mencari persentase akurasi dari sistem dengan menggunakan Persamaan 6.1.

$$Akurasi = \frac{Total\ data\ uji - Total\ data\ uji\ tidak\ sesuai}{Total\ data\ uji} * 100\% \quad (6.1)$$

6.3.2 Hasil Pengujian Akurasi Sistem Dalam Melakukan Klasifikasi

Tabel 6.3 Tingkat akurasi sistem

Pengujian ke-	Kondisi Aktual	Frame Total	Hasil Deteksi Sistem (frame)					Hasil Akhir	Akurasi (%)
			Golongan 1	Golongan 2	Golongan 3	Golongan 4	Golongan 5		
1	Golongan 1	33	20	8	5	3	0	Golongan 1	
2	Golongan 1	33	27	7	0	0	0	Golongan 1	
3	Golongan 1	28	20	8	0	0	0	Golongan 1	**80**
4	Golongan 1	27	17	9	0	1	0	Golongan 1	
5	Golongan 1	22	0	1	21	0	0	**Golongan 3**	
6	Golongan 2	48	0	33	5	0	1	Golongan 2	
7	Golongan 2	45	1	38	3	3	0	Golongan 2	
8	Golongan 2	47	4	14	16	6	7	**Golongan 3**	**80**
9	Golongan 2	49	0	32	6	1	10	Golongan 2	
10	Golongan 2	49	1	43	5	0	1	Golongan 2	
11	Golongan 3	55	2	7	46	0	0	Golongan 3	
12	Golongan 3	52	7	8	37	0	0	Golongan 3	
13	Golongan 3	56	0	27	25	0	4	**Golongan 2**	**60**
14	Golongan 3	56	1	21	34	0	0	Golongan 3	
15	Golongan 3	57	0	29	25	1	2	**Golongan 2**	

Tabel 6.3 Tingkat akurasi sistem (lanjutan)

Pengujian ke-	Kondisi Aktual	Frame Total	Hasil Deteksi Sistem (frame)					Hasil Akhir	Akurasi (%)
			Golongan 1	Golongan 2	Golongan 3	Golongan 4	Golongan 5		
16	Golongan 4	43	0	3	8	24	8	Golongan 4	
17	Golongan 4	53	1	0	0	52	0	Golongan 4	
18	Golongan 4	55	0	4	8	31	12	Golongan 4	**60**
19	Golongan 4	43	0	12	16	13	2	**Golongan 3**	
20	Golongan 4	55	0	42	11	1	1	**Golongan 2**	
21	Golongan 5	35	0	6	8	3	18	Golongan 5	
22	Golongan 5	41	4	4	15	10	8	**Golongan 3**	
23	Golongan 5	41	0	4	6	1	30	Golongan 5	**60**
24	Golongan 5	45	0	6	2	2	35	Golongan 5	
25	Golongan 5	62	0	16	31	1	14	**Golongan 3**	
Akurasi Total (%)									**68**

6.3.3 Analisis Pengujian Akurasi Sistem Dalam Melakukan Klasifikasi

Dari hasil yang didapat dari pengujian menunjukkan sistem masih terdapat kesalahan saat mengklasifikasi golongan kendaraan, hal itu dapat terjadi karena kurangnya variasi jenis kendaraan pada data latih, pengambilan foto kendaraan pada data latih kurang ideal, seperti pencahayaan yang buruk, dan terdapat *background* dari foto kendaraan yang ikut masuk dalam data latih maupun data uji, sehingga *background* yang seharusnya tidak menunjukkan fitur dari kendaraan ikut terklasifikasi.

BAB 7 PENUTUP

Bab ini menjelaskan kesimpulan dari analisis terhadap penelitian yang sudah dilakukan dan penyampaian saran yang dapat diterapkan untuk pengembangan sistem selanjutnya.

7.1 Kesimpulan

Berdasarkan hasil yang diperoleh dari pengujian yang sudah dilakukan dapat diambil kesimpulan sebagai berikut:

1. Sistem dapat melakukan klasifikasi golongan kendaraan dari masukan berupa video kendaraan, dan kemudian hasil klasifikasi ditampilkan pada layar LCD meskipun dengan *frame rate* yang tergolong rendah.
2. Nilai parameter *k* tetangga terdekat yang paling optimal dalam melakukan klasifikasi golongan kedaraan yaitu $k = 1$.
3. Tingkat akurasi sistem dalam melakukan klasifikasi kendaraan Golongan 1 sebesar 80%, Golongan 2 sebesar 80%, Golongan 3 sebesar 60%, Golongan 4 sebesar 60%, Golongan 5 sebesar 60%, sehingga secara keseluruhan akurasi sistem sebesar 68% yang didapatkan dari 25 data pengujian, dengan tiap golongan kendaraan sebanyak 5 data pengujian.

7.2 Saran

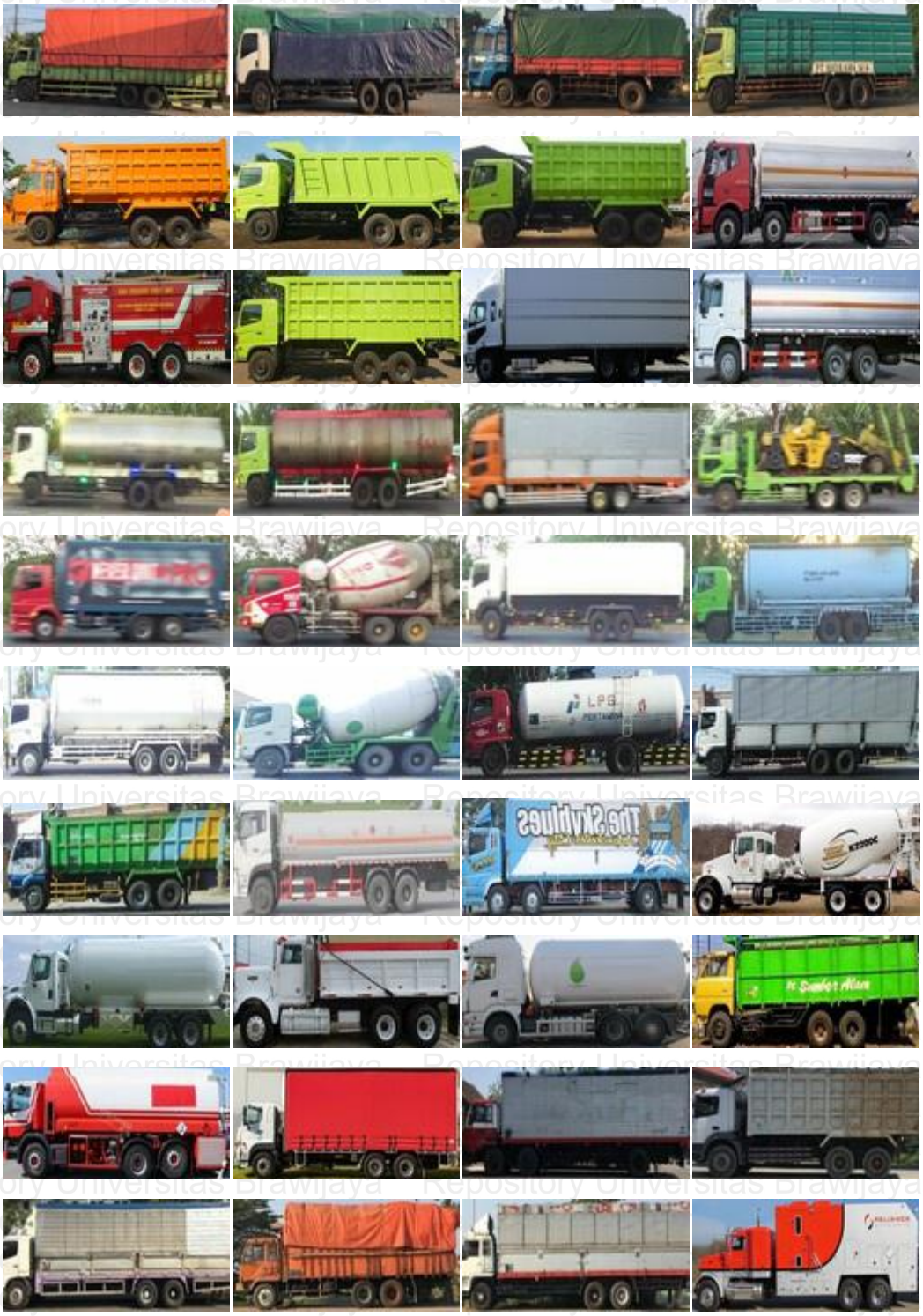
Berdasarkan hasil yang diperoleh dari pengujian yang sudah dilakukan dapat diberikan saran untuk pengembangan sistem sebagai berikut:

1. Untuk meningkatkan *frame rate* dari hasil klasifikasi yang ditampilkan pada layar dapat menggunakan layar LCD yang memiliki *interface* HDMI, dan menggunakan Raspberry Pi seri terbaru yang memiliki spesifikasi *hardware* lebih tinggi.
2. Untuk meningkatkan akurasi sistem dalam melakukan klasifikasi diperlukan penggunaan data latih dan data uji yang ideal, dalam arti lokasi pengambilan foto untuk data latih harus sama dengan lokasi pengambilan video untuk data uji, tingkat pencahayaan yang digunakan tidak jauh berbeda antara foto data latih dengan video data uji, jarak antara objek dengan kamera harus sama saat pengambilan foto untuk data latih dan video untuk data uji, tiap jenis kendaraan harus memiliki foto data uji dari berbagai sudut samping kendaraan.

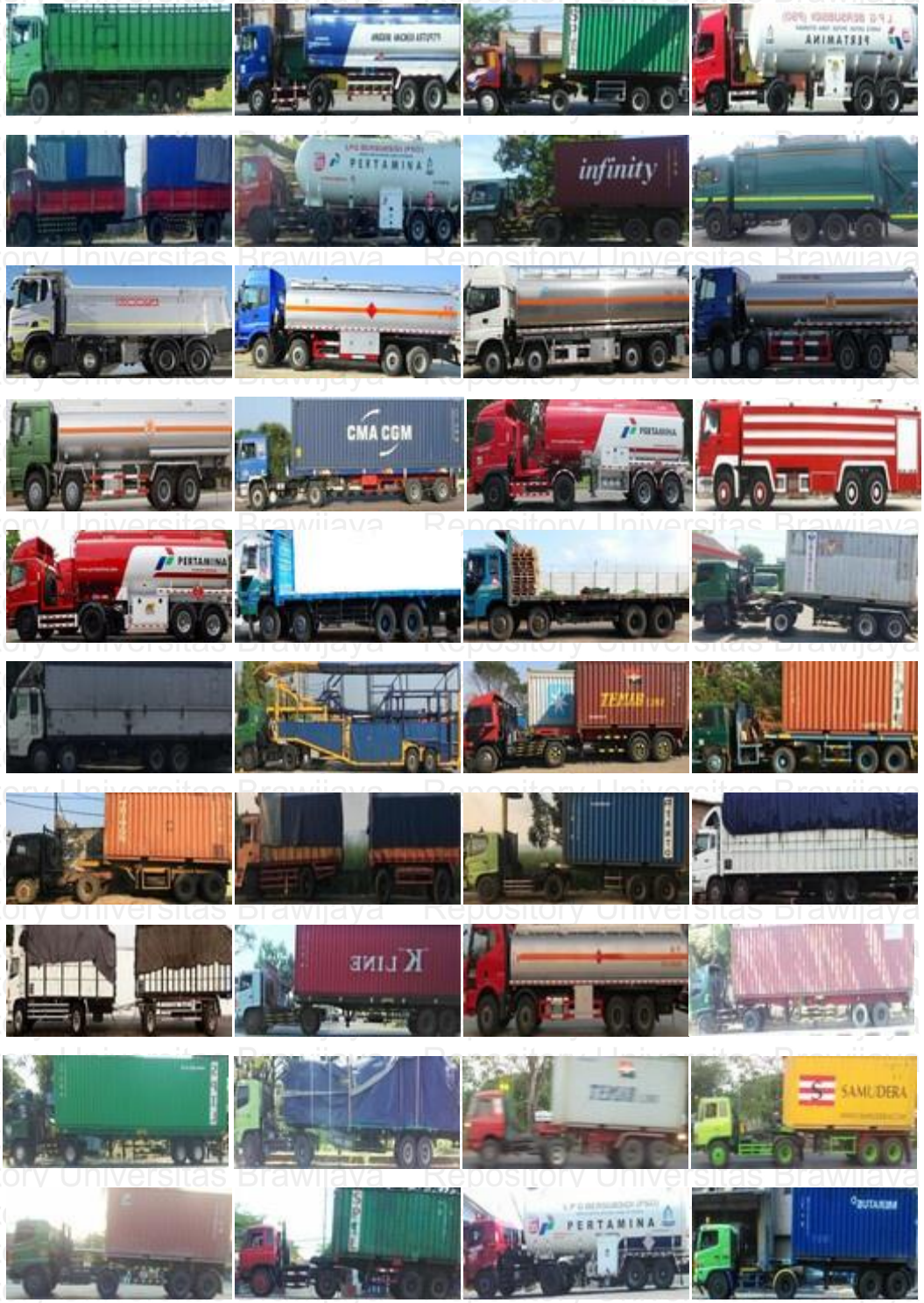
LAMPIRAN A DATA LATIH GOLONGAN 1

LAMPIRAN B DATA LATIH GOLONGAN 2

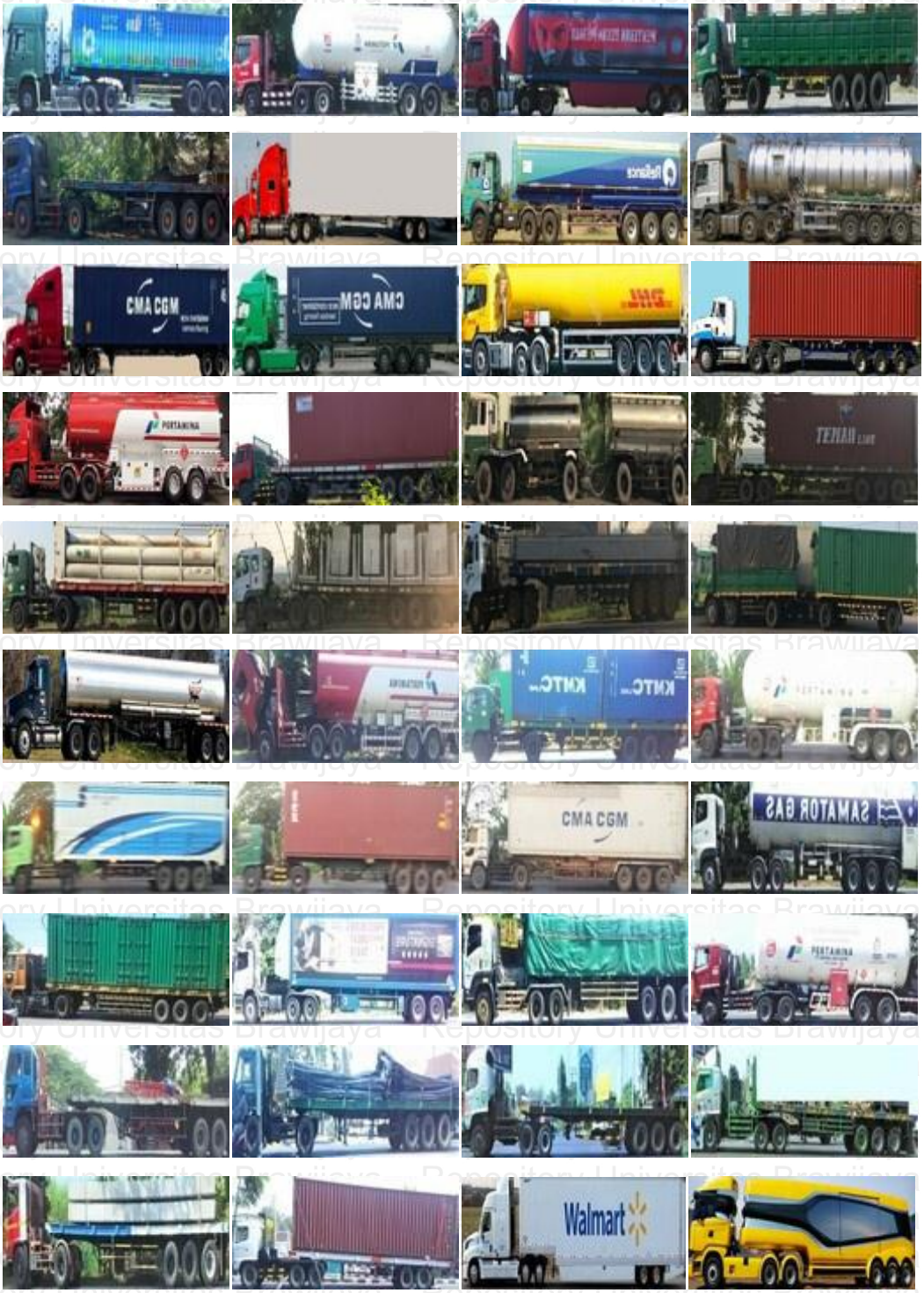
LAMPIRAN C DATA LATIH GOLONGAN 3



LAMPIRAN D DATA LATIH GOLONGAN 4



LAMPIRAN E DATA LATIH GOLONGAN 5



DAFTAR REFERENSI

Adistya, R., Muslim, M. A., 2016. Deteksi dan Klasifikasi Kendaraan menggunakan Algoritma Backpropagation dan Sobel. *Journal of Mechanical Engineering and Mechatronics* [online] Tersedia di: <http://e-journal.president.ac.id/presunivojs/index.php/JMEM/article/view/94/67> [Diakses 19 Januari 2019]

ArcGIS, 2019. Using spatial analytics to study spatio-temporal patterns in sport. [Online] Tersedia di: https://www.esri.com/arcgis-blog/products/arcgis-desktop/3d-gis/using-spatial-analytics-to-study-spatio-temporal-patterns-in-sport/?rmedium=blogs_esri_com&rsource=/esri/arcgis/2013/02/19/using-spatial-analytics-to-study-spatio-temporal-patterns-in-sport/ [Diakses 10 Juli 2019].

Bhatia, M., Vandana., 2010. Survey of nearest neighbor techniques. International Journal of Computer Science and Information Security 8, 1947-5500.

Bramer, M., 2007. Principles of Data Mining. London: Springer.

Budisanjaya, I. P. G. & Kumara, I. N. S., 2013. Perangkat Lunak Pengolahan Citra Untuk Segmentasi dan Cropping Daun Sawi Hijau. *Prosiding Conference on Smart-Green Technology in Electrical and Information System,* p. 217.

Febriliyanti, Y., Faradila, L. R., Taqwa, A., 2017. *Implementasi pengolahan citra dengan metode histogram of oriented gradient (hog) untuk pengaturan waktu pada traffic light berdasarkan deteksi kepadatan kendaraan*. Prosiding SNATIF Ke-4 Tahun 2017, [online] Tersedia di:< https://jurnal.umk.ac.id/index.php/SNA/article/view/1311 > [Diakses 19 Januari 2019]

Kachouane, M., Sahki, S., Lakrouf, M., Quadah, N., 2012. HOG Based fast Human Detection. In: Université Saad Dahlab de Blida, Centre de Développement des Technologies Avancées, 2012 24th International Conference on Microelectronics (ICM). Algiers, Algeria, 16-20 December 2012. IEEE

Kumparan, 2016. Mengenang Mereka yang 'Gugur' di Jalur Brexit. *kumparanNEWS* [online] Tersedia di: <https://kumparan.com/@kumparannews/mengenang-mereka-yang-gugur-di-jalur-brexit> [Diakses 19 Januari 2019]

Kusrini dan Lutfhi, E.T., 2009. Algoritma Data Mining, Andi Publishing, Yogyakarta.

Kusumanto, R. D., Tompunu, A. N., 2011. Pengolahan citra digital untuk mendeteksi obyek menggunakan pengolahan warna model normalisasi rgb. Seminar Nasional Teknologi Informasi & Komunikasi Terapan 2011 (Semantik 2011) [online] Tersedia di: <http://publikasi.dinus.ac.id/index.php/semantik/article/view/153/116>[Diakses 19 Januari 2019]

Leidiyana, H. 2013. Penerapan algoritma k-nearest neighbor untuk penentuan resiko kredit kepemilikan kendaraan bemotor. *System Embedded & Logic*, [online] Tersedia di: <https://media.neliti.com/media/publications/155541-ID-penerapan-algoritma-k-nearest-neighbor-u.pdf> [Diakses 19 Januari 2019]

Mallick, S., 2018. *Learn OpenCV.* [Online] Tersedia di: https://www.learnopencv.com/histogram-of-oriented-gradients/

Opencv, 2019. *Background Substraction.* [Online] Tersedia di: https://docs.opencv.org/3.3.0/db/d5c/tutorial_py_bg_subtraction.html

Opencv, 2019. *Countours Begin*. [Online] Tersedia di: https://docs.opencv.org/3.3.1/d4/d73/tutorial_py_contours_begin.html

Opencv, 2019. *Color Conversions*. [Online] Tesedia di: https://docs.opencv.org/3.1.0/de/d25/imgproc_color_conversions.html

Opencv, 2019. *Countour Features.* [Online] Tersedia di: https://docs.opencv.org/3.1.0/dd/d49/tutorial_py_contour_features.html

Opencv, 2019. *Erosion Dilation*. [Online] Tersedia di: https://docs.opencv.org/2.4/doc/tutorials/imgproc/erosion_dilatation/erosion_dilatation.html

Opencv, 2019. *Gaussian Median Blur Bilateral Filter*. [Online] Tersedia di: https://docs.opencv.org/2.4/doc/tutorials/imgproc/gausian_median_blur_bilateral_filter/gausian_median_blur_bilateral_filter.html

Opencv, 2019. *Resize Image*. [Online] Tersedia di: https://www.tutorialkart.com/opencv/python/opencv-python-resize-image/

Opencv, 2019. *Tresholding*. [Online] Tersedia di: https://docs.opencv.org/trunk/d7/d4d/tutorial_py_thresholding.html

Advernesia, 2019. Pengertian dan Cara Kerja Algoritma k-Nearest Neighbors . [Online] Tersedia di: https://www.advernesia.com/blog/data-science/pengertian-dan-cara-kerja-algoritma-k-nearest-neighbours-knn/

Raspberry Pi Foundation, 2016. *Raspberry Pi.* [Online] Tersedia di: https://www.raspberrypi.org/products/raspberry-pi-3-model-b/ [Diakses 10Juli 2019].

Logitech, 2019. Logitech C270 HD Webcam. [Online] Tersedia di: https://www.logitech.com/en-us/product/hd-webcam-c270 [Diakses 10 Juli 2019]

Robotix, 2018. *Robotix - Technology Robotix Society.* [Online] Tersedia di: https://2018.robotix.in/tutorial/imageprocessing/imagetypes/ [Diakses10Juli 2019].

Rosebrock, 2015. Basic motion detection and tracking with Python and OpenCV. [online] Tersedia di: https://www.pyimagesearch.com/2015/05/25/basic-motion-detection-and-tracking-with-python-and-opencv/ [Diakses 23 Mei 2019].

Waveshare, 2019. 3,5 inch RPi LCD (A). [Online] Tersedia di: https://www.waveshare.com/wiki/3.5inch_RPi_LCD_(A) [Diakses 10 Juli 2019].