TUGAS AKHIR – K141502

DETEKSI JENIS KENDARAAN di JALAN MENGGUNAKAN OPENCV

ALVIN LAZARO
NRP 5113100067

Dosen Pembimbing
Prof. Dr. Ir. JOKO LIANTO BULIALI, M.Sc.
BILQIS AMALIAH, S.Kom., M.Kom.

JURUSAN TEKNIK INFORMATIKA
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember
Surabaya 2017

[Halaman ini sengaja dikosongkan]

TUGAS AKHIR – K141502

DETEKSI JENIS KENDARAAN di JALAN MENGGUNAKAN OPENCV

ALVIN LAZARO
NRP 5113100067

Dosen Pembimbing
Prof. Dr. Ir. Joko Lianto Buliali, M.Sc.
Bilqis Amaliah, S.Kom, M.Kom.

JURUSAN TEKNIK INFORMATIKA
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember
Surabaya 2017

[Halaman ini sengaja dikosongkan]

FINAL PROJECT – K141502

VEHICLE TYPE DETECTION ON THE ROAD USING OPENCV

ALVIN LAZARO
NRP 5113100067

Supervisor
Prof. Dr. Ir. Joko Llanto Buliali, M.Sc.
Bilqis Amaliah, S.Kom, M.Kom.

Department of Informatics
Faculty of Information Technology
Institut Teknologi Sepuluh Nopember
Surabaya 2017

[Halaman ini sengaja dikosongkan]

LEMBAR PENGESAHAN

DETEKSI JENIS KENDARAAN di JALAN MENGGUNAKAN OPENCV

TUGAS AKHIR

Diajukan Guna Memenuhi Salah Satu Syarat
Memperoleh Gelar Sarjana Komputer
pada
Bidang Studi Dasar dan Terapan Komputasi
Program Studi S-1 Jurusan Teknik Informatika
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember

Oleh:
ALVIN LAZARO
NRP: 5113 100 067

Disetujui oleh Dosen Pembimbing Tugas Akhir:

Prof. Dr. Ir. Joko Lianto Buliali, M.Sc. ..... (pembimbing 1)
NIP: 19670727 199203 1 002

Bilqis Amaliah, S.Kom., M.Kom. .....
NIP: 19750914 200112 2 002 (pembimbing 2)

SURABAYA
JUNI 2017

[Halaman ini sengaja dikosongkan]

DETEKSI JENIS KENDARAAN di JALAN MENGGUNAKAN OPENCV

Nama Mahasiswa	:	Alvin Lazaro
NRP	:	5113 100 067
Jurusan	:	Teknik Informatika FTIf-ITS
Dosen Pembimbing 1	:	Prof. Dr. Ir. Joko Lianto Buliali, M.Sc.
Dosen Pembimbing 2	:	Bilqis Amaliah, S.Kom., M.Kom.

ABSTRAKSI

Saat ini, jalan raya merupakan komponen penting dalam lalu lintas. Dalam setiap kota, tidak semua jalan dapat dilalui oleh semua jenis kendaraan. Untuk membantu memantau kondisi jalan, diperlukan suatu program yang bisa mendeteksi jenis kendaraan yang melalui jalan tersebut. Sebelumnya, sudah ada program-program serupa yang mampu mendeteksi kendaraan di jalan raya. Namun, program-program tersebut hanya melakukan deteksi kendaraan saja. Program-program tersebut tidak mampu melakukan deteksi terhadap jenis kendaraan di jalan raya. Untuk itu, Tugas Akhir ini adalah pengembangan lebih lanjut dari deteksi kendaraan yaitu melakukan deteksi jenis kendaraan serta menghitung kendaraan yang terdeteksi berdasarkan jenisnya. Tujuan Tugas Akhir ini yaitu untuk membuat program yang mampu mengidentifikasi jenis kendaraan pada suatu input video dan menghitung jumlah kendaraan yang terdeteksi berdasarkan jenisnya.

Penulis menghasilkan file XML yang berisi fitur-fitur berbentuk angka-angka digital yang merepresentasikan karakteristik sebuah kendaraan. File XML ini dihasilkan melalui *data training* yang melibatkan gambar positif dan gambar negatif dari kendaraan. Kemudian, video lalu lintas dimasukkan ke program. Video yang digunakan adalah video dari kamera CCTV lalu lintas dengan sudut pandang dari tengah jalan. Lalu, program melakukan *capturing* pada *frame* video yang mengandung citra

kendaraan. Proses ini menggunakan library OpenCV. Citra kendaraan ini dicocokkan dengan file XML yang berisi fitur standar sebuah objek kendaraan. Lalu, program akan mengenali golongan dari objek kendaraan yang terdeteksi berdasarkan luas segi empat yang menjadi penanda objek kendaraan tersebut. Program mencatat jumlah kendaraan yang terdeteksi berdasarkan jenisnya. Hasil pencatatan ini dimasukkan ke dalam sebuah file .txt yang berisi ID kendaraan, jenis kendaraan, serta total kendaraan yang terdeteksi berdasarkan jenisnya. Dari hasil pengujian, program memiliki akurasi rata-rata 77.8% untuk kondisi jalan sepi, 47.5% untuk kondisi jalan normal, dan 28.2% untuk kondisi jalan padat. Hal-hal yang mempengaruhi akurasi deteksi adalah sudut pandang kendaraan terhadap kamera, jarak antar kendaraan, serta waktu tempuh kendaraan dalam memasuki hingga keluar dari wilayah deteksi.

Kata Kunci: Cascade, Classifier, Jenis kendaraan, Haar-like feature, OpenCV.

VEHICLE TYPE DETECTION ON THE ROAD USING OPENCV

Student Name	:	Alvin Lazaro
Student ID	:	5113 100 067
Major	:	Informatics Department FTIf-ITS
Advisor 1	:	Prof. Dr. Ir. Joko Lianto Buliali, M.Sc.
Advisor 2	:	Bilqis Amaliah, S.Kom, M.Kom.

ABSTRACT

Today, highways are an important component of traffic. In every city, not all roads can be passed by all types of vehicles. In order to help monitoring road conditions, a program that can detect the type of vehicle that passes through the road is needed. Previously, there are similar programs which can detect the vehicle on the road. However, these programs only do vehicle detection alone. These programs are unable to detect the type of vehicle on the highway. Therefore, this Final Project is the further development of vehicle detection, which is to detect the type of vehicle on the highway and calculate the number of vehicles detected based on its type. The purpose of this Final Project is to create a program which is able to identify the type of vehicle on a video input and calculate the number of vehicles detected based on its type.

The author generates an XML file which contain the features of digital numbers that represent the characteristics of a vehicle. Then, a traffic video is inserted to the program. The video used is a video of a CCTV camera traffic with a center point of view. The program is capturing the video frame containing the vehicle image. This process uses the OpenCV library. The vehicle image is matched to an XML file that contains standard features of a vehicle. The type of vehicle is identified based on the area of the rectangle that marks the object of the vehicle. The results of this identifications are written into a .txt file containing vehicle ID,

vehicle type, and total vehicle detected by type. From the test results, the program has 77.8% accuracy for deserted road conditions, 47.5% for normal road conditions, and 28.2% for crowded road conditions. Things that affect detection accuracy are the vehicle's viewing angle to the camera, the distance between vehicles, as well as the travel time of the vehicle in entering and getting out of the detection area.

Keywords: Cascade, Classifier, Vehicle type, Haar-like feature, OpenCV.

KATA PENGANTAR

Puji syukur kepada Bunda Maria dan Tuhan Yesus Kristus atas segala kasih karunia dan rahmat-Nya penulis dapat menyelesaikan Tugas Akhir yang berjudul:

DETEKSI JENIS KENDARAAN di JALAN MENGGUNAKAN OPENCV

Melalui lembar ini, penulis ingin menyampaikan ucapan terimakasih dan penghormatan yang sebesar-besarnya kepada:

1. Orangtua dan keluarga besar yang selalu memberikan dukungan penuh untuk menyelesaikan Tugas Akhir ini.
2. Prof. Dr. Ir. Joko Lianto Buliali, M.Sc., selaku dosen pembimbing 1 yang telah banyak membantu, membimbing, bahkan memotivasi penulis untuk menyelesaikan Tugas Akhir ini.
3. Ibu Bilqis Amaliah, S.Kom., M.Kom. selaku dosen pembimbing 2 yang telah banyak memberikan semangat, motivasi, serta arahan kepada penulis dalam menyelesaikan Tugas Akhir ini.
4. Andrew dan David, selaku tim yang membantu dalam pengerjaan Tugas Akhir ini.
5. Albert, Andre, Arianto, Billy, Johanes, Romario, dan Yosua selaku sahabat-sahabat karib yang banyak membantu penulis selama perkuliahan di Jurusan Teknik Informatika ITS.
6. Bapak dan Ibu dosen Jurusan Teknik Informatika ITS lainnya yang telah banyak menyampaikan ilmu dan bimbingan yang tak ternilai harganya bagi penulis.
7. Serta pihak-pihak lain yang namanya tidak dapat penulis sebutkan satu per satu.

Penulis menyadari masih terdapat banyak kekurangan baik dalam pelaksanaan Tugas Akhir maupun penyusunan buku laporan

ini, namun penulis berharap buku laporan ini dapat menambah wawasan pembaca dan dapat menjadi sumber referensi. Penulis mengharapkan kritik dan saran yang membangun untuk kesempurnaan penulisan buku Tugas Akhir ini.

Surabaya, April 2017

Alvin Lazaro

DAFTAR ISI

[Halaman ini sengaja dikosongkan]

DAFTAR GAMBAR

[Halaman ini sengaja dikosongkan]

DAFTAR TABEL

DAFTAR KODE SUMBER

[Halaman ini sengaja dikosongkan]

DAFTAR PERSAMAAN

[Halaman ini sengaja dikosongkan]

BAB I
PENDAHULUAN

Pada bab ini akan dipaparkan mengenai garis besar Tugas Akhir yang meliputi latar belakang, tujuan, rumusan dan batasan permasalahan, metodologi pembuatan Tugas Akhir, dan sistematika penulisan.

1.1. Latar Belakang

Dewasa ini, kondisi lalu lintas di jalan sudah mulai dipantau menggunakan CCTV. CCTV ini nantinya akan menghasilkan rekaman kondisi lalu lintas. Rekaman ini bisa dimanfaatkan untuk berbagai keperluan. Salah satunya adalah untuk mendeteksi jenis kendaraan yang melalui jalan tersebut.

Program akan mengenali objek kendaraan yang terdapat pada video rekaman. Objek kendaraan yang dikenali dapat digolongkan menjadi beberapa jenis. Misalkan, objek kendaraan yang terdeteksi digolongkan menjadi tiga jenis yaitu mobil kecil, mobil sedang, dan mobil besar. Nantinya, bisa dilakukan penghitungan terhadap setiap jenis kendaraan yang melewati jalan tersebut. Sehingga bisa diketahui, jumlah mobil berdasarkan jenisnya yang melalui jalan tersebut.

Pada kali ini, penulis akan menjawab implementasi kebutuhan mengenai deteksi jenis kendaraan di jalan. Input yang diperlukan adalah video. Nantinya, video ini diproses dengan bantuan *library* OpenCV. Proses yang dilakukan adalah menangkap *frame* video untuk mengenali kendaraan yang ada. Nantinya, kendaraan yang dikenali akan digolongkan menjadi tiga golongan yaitu mobil kecil, mobil sedang, dan mobil besar. Program ini dikembangkan menggunakan bahasa pemrograman Python. Dan juga, menggunakan *library* OpenCV untuk memproses inputan.

1.2. Tujuan

Tujuan dari pembuatan tugas akhir ini adalah membuat program dengan kemampuan sebagai berikut:

1. Mengidentifikasi jenis kendaraan pada suatu input video.
2. Menghitung jumlah kendaraan yang terdeteksi berdasarkan jenisnya.

1.3. Rumusan Permasalahan

Rumusan masalah yang diangkat dalam tugas akhir ini antara lain:

1. Metode apakah yang digunakan untuk mendeteksi objek kendaraan pada video?
2. Bagaimana menggolongkan objek kendaraan yang terdeteksi menjadi tiga jenis (mobil kecil, mobil sedang, mobil besar)?

1.4. Batasan Permasalahan

Permasalahan yang dibahas dalam tugas akhir ini memiliki beberapa batasan, antara lain:

1. Kendaraan yang terdeteksi digolongkan menjadi 3 golongan yaitu mobil kecil, mobil sedang, mobil besar.
2. Resolusi minimal kamera video adalah 3MP.
3. Video yang digunakan berformat .mp4.
4. Sudut pandang kendaraan terhadap kamera adalah tampak depan dari tengah jalan.
5. Kondisi cuaca cerah dan siang hari.
6. Tingkat fps video yang digunakan adalah 10-15 fps.

1.5. Metodologi

Langkah-langkah yang ditempuh dalam pengerjaan Tugas Akhir ini yaitu:

a. Penyusunan proposal tugas akhir

Proposal Tugas Akhir ini berisi sebagai gambaran penelitian yang akan dilakukan oleh penulis. Proposal dibuat selama semester 7 tahun ajaran 2016/2017.

Proposal Tugas Akhir ini terdiri dari deskripsi pendahuluan yang menjabarkan latar belakang dan rumusan masalah yang mendasari dibangunnya aplikasi ini, batasan masalah dalam pembangunan aplikasi ini, serta tujuan dan manfaat yang diharapkan dapat dicapai dengan dibangunnya aplikasi ini. Selain itu, pada proposal Tugas Akhir ini juga terdapat tinjauan pustaka yang menjelaskan teori-teori yang menjadi dasar pembuatan tugas akhir ini.

b. Studi literatur

Studi literatur yang dilakukan dalam pengerjaan Tugas Akhir ini adalah mengenai identifikasi kendaraan dari suatu video. Selain itu, juga dilakukan studi literatur mengenai cara penggunaan OpenCV untuk mendeteksi kendaraan pada video.

c. Analisis dan desain perangkat lunak

Rencana perangkat lunak yang digunakan adalah aplikasi berbasis desktop. Input yang diterima berupa video yang diproses oleh aplikasi dan akan melakukan deteksi kendaraan serta mengidentifikasi kendaraan yang telah dideteksi berdasarkan jenisnya. Kemudian, dilakukan penghitungan untuk menghitung jumlah kendaraan berdasarkan jenisnya.

d. Implementasi perangkat lunak

Implementasi untuk melakukan percobaan ini adalah menggunakan library OpenCV. Library ini digunakan untuk memproses input yang berupa video. Implementasi tool ini dibangun dengan menggunakan bahasa pemrograman Python.

e. Pengujian dan evaluasi

Pada tahapan ini dilakukan uji coba terhadap perangkat lunak yang telah dibuat. Pengujian yang dimaksud adalah pengujian fungsionalitas aplikasi yang dibangun. Pengujian dan evaluasi akan dilakukan dengan melihat kesesuaian dengan perencanaan. Tahap ini dimaksudkan juga untuk mengevaluasi jalannya aplikasi, mencari masalah yang mungkin timbul dan mengadakan perbaikan jika terdapat kesalahan.

f. Penyusunan Buku Tugas Akhir

Pada tahap ini dilakukan penyusunan laporan yang menjelaskan dasar teori dan metode yang digunakan dalam tugas akhir ini serta hasil dari implementasi aplikasi perangkat lunak yang telah dibuat. Sistematika penulisan buku tugas akhir secara garis besar antara lain:

1. Pendahuluan
    a. Latar Belakang
    b. Rumusan Masalah
    c. Batasan Tugas Akhir
    d. Tujuan
    e. Metodologi
    f. Sistematika Penulisan
2. Tinjauan Pustaka
3. Desain dan Implementasi
4. Pengujian dan Evaluasi
5. Kesimpulan dan Saran
6. Daftar Pustaka

1.6. Sistematika Penulisan

Buku Tugas Akhir ini bertujuan untuk mendapatkan gambaran dari pengerjaan Tugas Akhir ini. Selain itu, diharapkan dapat berguna untuk pembaca yang tertarik untuk melakukan pengembangan lebih lanjut. Secara garis besar, buku Tugas Akhir terdiri atas beberapa bagian seperti berikut ini.

Bab I Pendahuluan

Bab ini berisi latar belakang masalah, tujuan dan manfaat pembuatan Tugas Akhir, permasalahan, batasan masalah, metodologi yang digunakan, dan sistematika penyusunan Tugas Akhir.

Bab II Dasar Teori

Bab ini berisi beberapa teori penunjang yang berhubungan dengan pokok pembahasan dan mendasari pembuatan Tugas Akhir ini.

Bab III Analisis dan Perancangan Sistem

Bab ini berisi tentang analisis serta perancangan sistem. Perancangan meliputi perancangan data, arsitektur, dan proses yang dilakukan oleh aplikasi.

Bab IV Implementasi

Bab ini berisi implementasi dari perancangan dan implementasi fitur-fitur penunjang aplikasi.

Bab V Pengujian dan Evaluasi

Bab ini berisi lingkungan pengujian, skenario pengujian, dan evaluasi pengujian setelah aplikasi selesai dikembangkan.

Bab VI Kesimpulan dan Saran

Bab ini berisi kesimpulan dari hasil pengujian yang dilakukan serta saran-saran untuk pengembangan sistem lebih lanjut.

Daftar Pustaka

Merupakan daftar referensi yang digunakan untuk mengembangkan Tugas Akhir.

Lampiran

Merupakan bab tambahan yang berisi daftar istilah yang penting pada aplikasi ini.

BAB II
DASAR TEORI

Pada bab ini akan dibahas mengenai teori-teori yang menjadi dasar dari pembuatan Tugas Akhir.

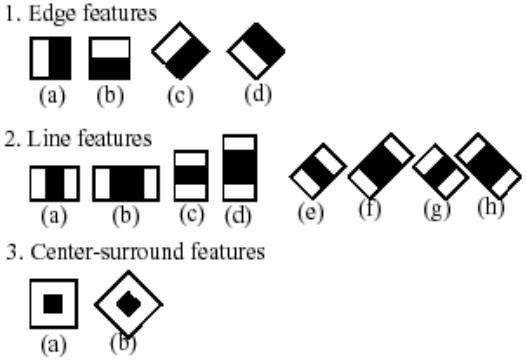
2.1. *OpenCV*

OpenCV (Open Source Computer Vision) adalah library dari fungsi pemrograman untuk realtime visi komputer [1]. OpenCV menggunakan lisensi BSD dan bersifat gratis baik untuk penggunaan akademis maupun komersial. OpenCV dapat digunakan dalam bahasa pemrograman C, C++, Python, Java, dan sebagainya [2]. OpenCV dapat digunakan pada sistem operasi Windows, Linux, Android, iOS dan Mac OS. OpenCV memiliki lebih dari 2500 algoritma yang telah dioptimalkan. Dalam penelitian ini, penulis menggunakan OpenCV 2.4.13.

2.2. *Haar-like Feature*

Haar-like feature pertama kali diusulkan oleh Paul Viola dan Michael Jones [3] untuk keperluan mendeteksi wajah manusia [4]. Metode ini kemudian diperbaharui kembali oleh Rainer Lienhart dan Jochen Maydt [5]. Ide dari Haar-like feature adalah sebuah *classifier* yang di-training dengan sejumlah sampel citra dari suatu objek. *Classifier* di-*training* menggunakan algoritma Adaboost. Dalam kasus ini, sampel citra yang digunakan adalah citra dari kendaraan, berupa tampak samping (kiri dan kanan), depan, dan belakang. Ukuran citra yang digunakan untuk training harus sama (misalkan 20x20), di mana nantinya ini akan menjadi sampel positif. Sampel negatif adalah citra dari objek yang berbeda-beda namun tetap memililki ukuran yang sama. Kumpulan citra ini akan menghasilkan kumpulan fitur objek yang disebut sebagai *cascade*. Classifier akan menghasilkan nilai “1” jika pada

citra yang dimasukkan terdapat objek yang dikenali dan nilai "0" jika tidak ada objek yang dikenali.



Gambar 2-1 Berbagai bentuk *Haar-like Feature*

Hasil penerapan masing-masing fitur pada suatu wilayah gambar tertentu dihasilkan melalui jumlah piksel yang terletak di dalam segi empat hitam dari fitur yang dikurangkan oleh jumlah piksel yang *overlap* dengan segi empat putih. Segi empat ini didefinisikan lewat koordinat kiri atas x, y, lebar w, dan tinggi h. Total piksel yang berada dalam area segi empat r_i direpresentasikan oleh RecSum(r_i).

$$feature_1 = \sum_{i=1}^{N} W_i \times RecSum(r_i)$$

$$= \sum_{i=1}^{N} W_i \times RecSum(x, y, w, h)$$

Persamaan (2-1)

Di mana dalam persamaan ini, nilai N, W_i, dan r_i dipilih secara acak, tergantung objek yang diidentifikasi.

Dalam mengenali suatu objek, *cascade* akan mengabaikan area citra yang tidak memiliki objek yang memnuhi kriteria. Ini sangat membantu dalam meningkatkan performa *classifier* tersebut. Efeknya, kemungkinan kesalahan deteksi objek berkurang dan akurasi deteksi menjadi meningkat.

2.3. *Frame Rate*

Frame rate adalah frekuensi di mana sebuah gambar ditampilkan dalam suatu video atau film. Satuan yang digunakan adalah frame per second (fps). Semakin tinggi nilai fps dari sebuah video maka semakin halus pergerakan gambarnya.

2.4. *Mean Squared Error* (MSE)

Mean squared error digunakan untuk mengetahui perbedaan antara estimator dengan hasil estimasi [6]. MSE digunakan sebagai tolok ukur suatu estimator. Hasil yang diperoleh selalu berupa angka positif. Semakin mendekati nol maka semakin baik kinerja estimator tersebut.

$$MSE = \frac{\sum_{i=0}^{n}(X_i - F_i)^2}{n}$$

Persamaan (2-2)

Dalam penelitian ini, MSE digunakan untuk mengecek apakah sebuah citra merupakan sebuah objek kendaraan atau bukan. Ini dilakukan dengan membandingkan kesimetrisan secara horizontal dan vertikal dari sebuah citra. Sebuah citra kendaraan bersifat simetris secara horizontal maupun vertikal. Kesimetrisan horizontal digunakan untuk mengetahui tinggi objek kendaraan sedangkan kesimetrisan vertikal digunakan untuk mengetahui apakah objek yang terdeteksi sisi kiri dan kanannya simetris atau tidak. Ini dapat mempermudah dalam menandai objek kendaraan yang terdeteksi. Jika perbedaan kesimetrisan terlalu besar, citra yang terdeteksi bukan merupakan objek kendaraan.

2.5. *Classifier Cascade*

Penelitian ini menggunakan fitur objek sebagai dasar dalam deteksi. Alasannya, konsep ini lebih cepat dibandingkan dengan menggunakan piksel. *Classifier* dikembangkan dengan menggunakan fitur-fitur objek yang akan dideteksi. Fitur-fitur ini merupakan turunan dari fungsi Haar seperti yang digunakan oleh Papageorgiu et al [7].

Dalam penelitian ini, penulis menggunakan konsep Two Rectangle Feature (TRF) [8]. Nilai TRF diperoleh melalui selisih antara jumlah piksel dalam dua wilayah segi empat pada suatu Region of Interest (ROI). ROI merupakan wilayah di mana objek yang akan dideteksi berada. Untuk menggunakan TRF, kedua wilayah segi empat harus berukuran sama dan bersifat simetris secara horizontal maupun vertikal.

2.6. *Kendaraan*

Menurut Kamus Besar Bahasa Indonesia (KBBI) kendaraan merupakan sesuatu yang digunakan untuk dikendarai atau dinaiki [9]. Dalam penelitian ini, penulis melakukan deteksi terhadap kendaraan mobil. Kemudian, penulis melakukan identifikasi terhadap jenis mobil yang terdeteksi. Jenis mobil yang terdeteksi akan dikelompokkan menjadi 3 golongan yaitu kendaraan kecil, kendaraan sedang, dan kendaraan besar. Hasil yang diperoleh dapat dimanfaatkan untuk mengetahui jumlah kendaraan berdasarkan jenisnya yang melewati jalan tersebut dalam suatu interval waktu tertentu. Program yang dikembangkan oleh penulis mampu mendeteksi jenis kendaraan serta menghitung jenis kendaraan yang terdeteksi berdasarkan jenisnya dalam suatu interval waktu tertentu, tergantung pada panjangnya video yang digunakan sebagai inputan.

BAB III
ANALISIS DAN PERANCANGAN SISTEM

Bab ini membahas tahap analisis dan perancangan program yang akan dibangun. Analisis membahas semua persiapan yang akan menjadi pokok pikiran pembuatan aplikasi ini. Mulai dari masalah yang melatarbelakangi, hingga analisis gambaran awal sistem yang akan dibuat. Perancangan sistem membahas hal-hal yang berkaitan dengan pondasi atau dasar pembuatan aplikasi.

3.1. Analisis

Tahap analisis meliputi analisis masalah, analisis kebutuhan, deskripsi umum sistem, dan fungsi-fungsi yang dimiliki sistem.

3.1.1. Penelitian Terkait

Sebenarnya sudah ada penelitian-penelitian terkait mengenai deteksi kendaraan di jalan raya. Contoh pertama adalah deteksi kendaraan secara otomatis di jalan raya oleh M. Oliveira dan V. Santos [10] pada tahun 2008. Contoh kedua adalah deteksi kendaraan di jalan raya secara *real time* menggunakan fitur Haar oleh Han et al [11] pada tahun 2009. Kedua penelitian tersebut mampu mendeteksi objek kendaraan dengan baik. Pada tahun 2010, Sayaman Sivaraman dan Mohan Manubhai Trivedi mengembangkan suatu *framework* [12] untuk mengenali dan melakukan *tracking* objek kendaraan. Penelitian ini menghasilkan sebuah *vehicle recognizer* yang mampu mengenali objek kendaraan. Hasil yang diperoleh dari penelitian ini menunjukkan bahwa *framework* yang dibangun mampu melakukan deteksi dan *tracking* kendaraan dengan akurasi yang baik.

Semua penelitian tersebut hanya melakukan deteksi terhadap objek kendaraan. Penelitian-penelitian tersebut belum melakukan klasifikasi atas jenis kendaraan yang terdeteksi. Sehingga terobosan dari penelitian yang dilakukan oleh penulis adalah klasifikasi jenis kendaraan di jalan raya.

3.1.2. Analisis Permasalahan

Oleh karena itu, aplikasi CarDetection ini dibuat untuk menangani permasalahan di atas. Pengguna dapat melakukan input video untuk mengetahui jenis kendaraan yang terdapat dalam video tersebut. Dari video tersebut, aplikasi dapat memberikan informasi mengenai jenis kendaraan yang terdeteksi serta jumlah kendaraan yang terdeteksi.

Dewasa ini, kondisi lalu lintas di jalan sudah mulai dipantau menggunakan CCTV. CCTV ini nantinya akan menghasilkan rekaman kondisi lalu lintas. Rekaman ini bisa dimanfaatkan untuk berbagai keperluan. Salah satunya adalah untuk mendeteksi jenis kendaraan yang melalui jalan tersebut. Program akan mengenali objek kendaraan yang terdapat pada video rekaman. Objek kendaraan yang dikenali digolongkan menjadi tiga jenis yaitu mobil kecil, mobil sedang, dan mobil besar. Kemudian dilakukan penghitungan terhadap setiap jenis kendaraan yang melewati jalan tersebut. Sehingga bisa diketahui, jumlah mobil berdasarkan jenisnya yang melalui jalan tersebut.

Pengguna dari program ini adalah pihak-pihak yang berhubungan dengan lalu lintas. Misalnya, Dinas Perhubungan atau Polantas. Kedua pihak ini dapat memanfaatkan data yang diperoleh untuk melakukan optimasi arus lalu lintas di jalan tersebut. Atau, untuk mengatur jenis kendaraan yang diperbolehkan melalui jalan tersebut.

3.1.3. Analisis Kebutuhan

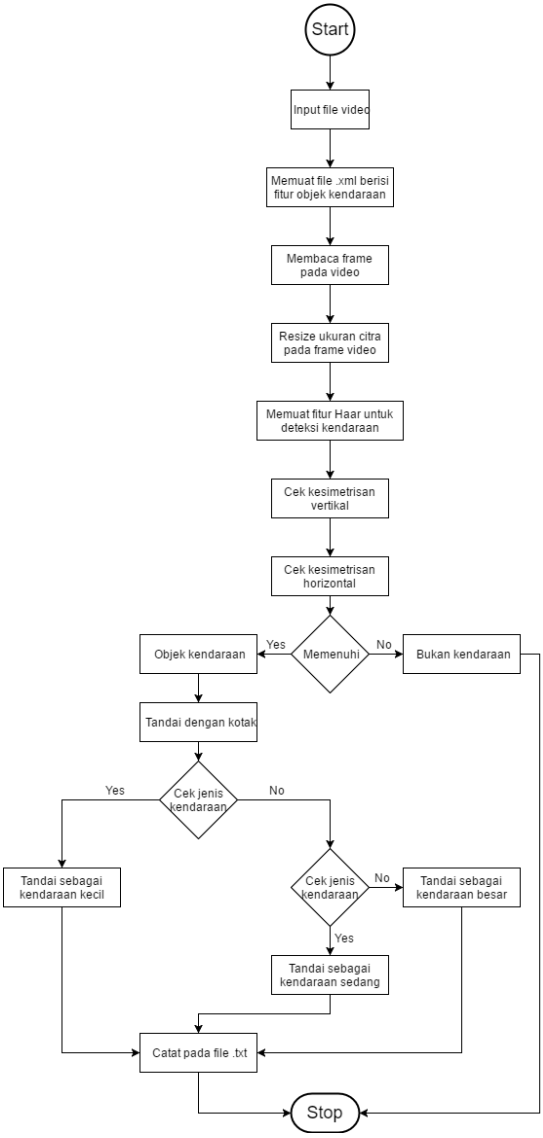
Kebutuhan utama dalam aplikasi ini difokuskan untuk mendeteksi jenis kendaraan dalam suatu video. Sebenarnya, sudah ada beberapa penelitian yang mampu melakukan deteksi kendaraan namun tidak dapat mendeteksi jenis kendaraan. Oleh karena itu, aplikasi ini dibangun dengan tujuan untuk mengatasi batasan tersebut. Tabel 3-1 merupakan daftar fungsi perangkat lunak yang dibangun dalam penelitian ini.

Tabel 3-1 Daftar Fungsi Sistem

Kode Fungsi	Nama Fungsi	Deskripsi
F-1	Mendeteksi objek kendaraan	Partisipan dapat mengenali objek kendaraan pada video.
F-2	Mendeteksi jenis kendaraan	Partisipan dapat mengenali jenis kendaraan yang terdapat pada video.
F-3	Mencatat hasil deteksi jenis kendaraan	Partisipan dapat mengetahui jumlah kendaraan yang terdeteksi berdasarkan jenisnya.

3.1.4. Deskripsi Umum Sistem

Aplikasi yang dibuat pada Tugas Akhir ini adalah program aplikasi untuk komputer. Gambar 3-1 adalah gambaran alur jalan sistem.



Gambar 3-1 Alur Jalan Sistem

Untuk penjelasan secara rinci akan dijelaskan di bawah mulai dari proses *input*, proses, hingga *output*.

3.1.4.1. *Input*

Langkah awal, sebuah video dimasukkan ke dalam program. Video dimasukkan dengan cara menuliskan nama file video beserta ekstensinya (misal center.mp4). Video menggunakan sudut pandang kamera tampak dari tengah jalan dan kondisi siang hari.

3.1.4.2. Proses

Pada tahap ini dijelaskan secara bertahap langkah-langkah yang dilakukan pada sistem dimulai dari proses ekstraksi fitur, pengenalan jenis kendaraan pada video, sampai menjadi hasil *output*.

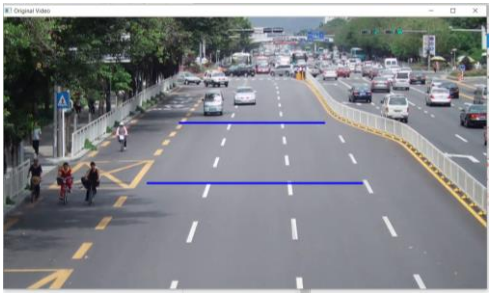
3.1.4.2.1 Memuat File XML

Setelah video dimasukkan, aplikasi akan memuat file XML yang berisi fitur-fitur objek kendaraan. Tujuan dari tahap ini adalah persiapan template untuk menetapkan apakah sebuah objek dianggap sebgai kendaraan atau bukan. File XML ini digunakan untuk pencocokan terhadap kandidat objek kendaraan pada video.

File XML berisi fitur-fitur yang merepresentasikan sebuah objek kendaraan. Setiap kandidat objek kendaraan dicocokkan dengan file XML tersebut. Jika memenuhi kriteria yang ditetapkan, objek tersebut akan dianggap sebagai objek kendaraan. Dan objek tersebut ditandai dengan segi empat berwarna merah.

3.1.4.2.2 *Processing*

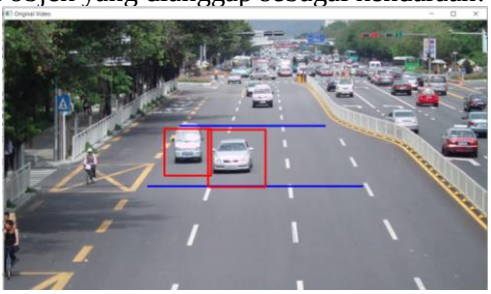
Pada tahap ini, aplikasi akan melakukan pembacaan video secara *frame by frame*. Pembacaan ini dilakukan pada area video yang sudah ditetapkan sebelumnya, yaitu area yang diberi garis batas berwarna biru. Di luar area tersebut, kendaraan tidak akan dideteksi.



Gambar 3-2 Batas area deteksi (antara dua garis biru)

Hal ini dilakukan untuk mengurangi risiko kesalahan deteksi. Dengan demikian, pengenalan objek kendaraan akan menjadi lebih fokus. Sehingga pengenalan jenis kendaraan yang dideteksi akan lebih akurat.

Kemudian dilanjutkan dengan mendeteksi objek kendaraan menggunakan *classifier*. *Classifier* akan mencocokkan objek kendaraan file XML yang telah dimuat sebelumnya. Jika memenuhi syarat, selanjutnya akan dicek kesimetrisan objek kendaraan yang terdeteksi. Objek kendaraan yang terdeteksi akan dicek apakah simetris secara vertikal dan horizontal atau tidak. Jika simeteris secara vertikal dan horizontal, objek tersebut dianggap sebagai kendaraan. Aplikasi akan memberikan tanda segi empat pada objek yang dianggap sebagai kendaraan.



Gambar 3-3 Penandaan objek yang terdeteksi sebagai kendaraan

Langkah berikutnya adalah pengenalan jenis kendaraan. Pengenalan jenis kendaraan dilakukan dengan mengukur luas segi empat yang digunakan untuk menandai objek kendaraan. Tabel 3-

2 memberikan detil ukuran luas segi empat untuk tiga jenis kendaraan yaitu kecil, sedang, dan besar.

Tabel 3-2 Luas Minimal Penanda Tiap Jenis Kendaraan

Jenis Kendaraan	Luas Segi Empat (piksel)
Kendaraan kecil	≤132
Kendaraan sedang	133 – 148
Kendaraan besar	≥149

Setelah dilakukan pengenalan jenis kendaraan, aplikasi akan mencatat ID kendaraan serta jenisnya pada file .txt. Ini untuk memudahkan pengguna dalam mengetahui jenis kendaraan yang telah terdeteksi dari video yang dimasukkan.

3.1.4.3. *Output*

Masuk tahap terakhir yaitu hasil. Hasil dari aplikasi ini adalah sebuah file .txt yang berisi jenis kendaraan yang terdeteksi dari video inputan. File .txt ini memudahkan pengguna untuk mengetahui jenis kendaraan yang terdeteksi dari video inputan.

3.1.5. Fungsi Utama Program

Program CarDetection memiliki 3 fungsi utama yaitu mendeteksi objek kendaraan pada video, mendeteksi jenis kendaraan pada video, serta mencatat hasil deteksi pada file .txt. Secara lebih jelas, ketiga fungsi tersebut dijelaskan pada subbab berikutnya.

3.1.5.1. Mengenali objek kendaraan

Fungsi diakses pengguna setelah memasukkan file video berformat .mp4 ke dalam aplikasi. Pada tahap ini pengguna akan memperoleh informasi mengenai objek kendaraan yang terdeteksi oleh aplikasi. Aplikasi akan menandai objek yang dianggap sebagai kendaraan dengan menggunakan segi empat. Nantinya, segi empat ini digunakan untuk mengenali jenis kendaraan tersebut.

3.1.5.2. Mengenali jenis kendaraan pada video

Fungsi dilakukan oleh sistem setelah melakukan pengenalan objek kendaraan. Jenis kendaraan diketahui berdasarkan luas segi empat yang menjadi penanda objek kendaraan.

3.1.5.3. Mencatat hasil deteksi pada file .txt

Fungsi dilakukan oleh sistem setelah melakukan pengenalan jenis kendararan yang terdeteksi. Pada tahap ini aplikasi akan menghasilkan sebuah file .txt yang berisi ID kendaraan yang terdeteksi beserta jenisnya.

3.2. Perancangan Sistem

Tahap ini meliputi perancangan basis data, tampilan antarmuka, dan perancangan alur proses penggunaan sistem yang diharapkan dapat memenuhi tujuan dari pengembangan aplikasi ini. Perlu diketahui bahwa aplikasi ini dibangun dalam kondisi lingkungan tertentu, dan dapat dioperasikan dalam lingkungan tertentu pula. Lingkungan pengembangan aplikasi adalah sebagai berikut.

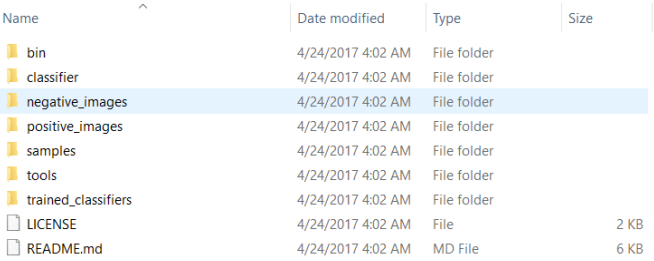
Komputer	: Prosesor Intel® Core™ i5-5200U CPU @ 2.20GHz, RAM 4 GB
Sistem operasi	: Windows 10
IDE	: Spyder 3.1.2
Bahasa Pemrograman	: Python 2.7.13
Versi OpenCV	: OpenCV 2.4.13

3.2.1. Perancangan Template Kendaraan

Pada subbab ini dijelaskan mengenai perancangan template kendaraan. Template kendaraan (berupa file XML) digunakan untuk mendeteksi objek kendaraan pada video. Gambaran perancangan template kendaraan adalah sebagai berikut.

3.2.1.1. Persiapan Direktori

Penulis menggunakan susunan direktori sebagai berikut. Susunan direktori ini menggunakan contoh dari *repository* milik github mrnugget [13] (https://github.com/mrnugget/opencv-haar-classifier-training).



Gambar 3-4 Susunan direktori

3.2.1.2. Menghasilkan Sampel

Tahap berikutnya adalah menghasilkan sampel positif dan negatif dari objek kendaraan. Berikut ini adalah langkah-langkah yang dijalankan.

1. Siapkan gambar objek yang diinginkan. Sampel terdiri dari 40 gambar positif dan 600 gambar negatif. Pastikan ukuran sampel tidak berbeda jauh satu sama lain.
2. Letakkan gambar positif pada folder *positive_images*. Jalankan perintah berikut di shell pada *root repository*.
   ```
   find ./positive_images -iname "*.jpg" > positives.txt
   ```
3. Letakkan gambar negatif pada direktori *negative_images*. Jalankan perintah berikut di shell pada *root repository*.
   ```
   find ./negative_images -iname "*.jpg" > negatives.txt
   ```
4. Gunakan *tools* dari OpenCV (opencv_createsamples) untuk menghasilkan sampel dari gambar positif yang dimasukkan. *Tools* ini melakukan transformasi dan distorsi pada gambar. Proses ini menghasilkan sebuah file .vec yang nantinya digunakan untuk proses *training classifier*. Jalankan perintah berikut di shell pada *root repository*.

```
perl bin/createsamples.pl positives.txt negatives.txt samples 1500\
"opencv_createsamples -bgcolor 0 -bgthresh 0 -maxxangle 1.1\
-maxyangle 1.1 maxzangle 0.5 -maxidev 40 -w 80 -h 40"
```

Proses ini menggunakan script dari Perl (createsamples.pl) untuk mendapatkan 1,500 sampel positif, dengan menggabungkan setiap gambar positif dengan gambar negatif acak dan kemudian menjalankannya melalui opencv_createsamples.

5. Hal berikutnya yang perlu dilakukan adalah menggabungkan file .vec yang sekarang ada di direktori *samples*. Untuk melakukannya, kita perlu mendapatkan daftar dari mereka dan kemudian menggunakan *tools* mergevec.cpp milik Naotoshi Seo [14] untuk menggabungkannya menjadi satu file .vec. Untuk menggunakannya, kita perlu menyalinnya ke dalam direktori sumber OpenCV dan mengkompilasinya di sana dengan file OpenCV lainnya.

```
cp src/mergevec.cpp ~/opencv-2.4.5/apps/haartraining
   cd ~/opencv-2.4.5/apps/haartraining
   g++ `pkg-config --libs --cflags opencv` -I. -o mergevec mergevec.cpp\
     cvboost.cpp cvcommon.cpp cvsamples.cpp cvhaarclassifier.cpp\
     cvhaartraining.cpp\
     -lopencv_core -lopencv_calib3d -lopencv_imgproc -lopencv_highgui -lopencv_objdetect
```

Kemudian, kembali ke *repository*, membawa mergevec yang bisa dieksekusi, dan menggunakannya.

```
find ./samples -name '*.vec' > samples.txt
./mergevec samples.txt samples.vec
```

6. Sekarang, sample.vec yang dihasilkan bisa digunakan untuk untuk memulai *training classifier*.

3.2.1.3. Melakukan *Training Classifier*

Tahap berikutnya adalah melakukan *training classifier*. Perlu diingat bahwa proses ini memakan waktu yang cukup lama (sekitar 2 hari), tergantung spesifikasi perangkat yang digunakan. Namun, proses ini bisa dihentikan sementara atau di-restart. Berikut adalah langkah-langkah yang dilakukan.

1. Arahkan opencv_traincascade ke sampel positif (samples.vec), gambar negatif.
2. Tulis hasilnya ke dalam direktori pengelompokan repositori kami dan ukuran sampel (-w dan -h). Variabel -numNeg

menentukan berapa banyak sampel negatif yang ada. Variabel -precalcValBufSize dan variabel -precalcIdxBufSize menentukan berapa banyak memori yang akan digunakan saat training. Perlu diingat, variabel -numPos harus lebih rendah dari sampel positif yang dihasilkan.

```
opencv_traincascade -data classifier -vec samples.vec -bg negatives.txt\
  -numStages 20 -minHitRate 0.999 -maxFalseAlarmRate 0.5 -numPos 1000\
  -numNeg 600 -w 80 -h 40 -mode ALL -precalcValBufSize 1024\
  -precalcIdxBufSize 1024
```

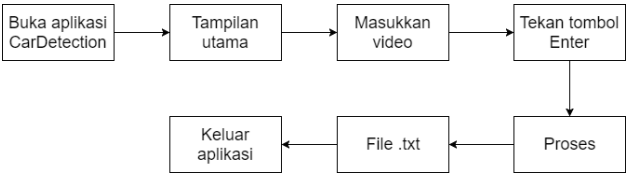
3. Setelah memulai proses *training*, proses akan mencetak kembali parameternya dan kemudian memulai proses training. Setiap tahap akan mencetak beberapa analisis.

```
===== TRAINING 0-stage =====
<BEGIN
POS count : consumed   1000 : 1000
NEG count : acceptanceRatio    600 : 1
Precalculation time: 11
+----+---------+---------+
|  N |    HR   |    FA   |
+----+---------+---------+
|   1|        1|        1|
+----+---------+---------+
|   2|        1|        1|
+----+---------+---------+
|   3|        1|        1|
+----+---------+---------+
|   4|        1|        1|
+----+---------+---------+
|   5|        1|        1|
+----+---------+---------+
|   6|        1|        1|
+----+---------+---------+
|   7|        1| 0.711667|
+----+---------+---------+
|   8|        1|     0.54|
+----+---------+---------+
|   9|        1|    0.305|
+----+---------+---------+
END>
Training until now has taken 0 days 3 hours 19 minutes 16 seconds.
```

   Setiap baris mewakili fitur yang di-train dan berisi beberapa keluaran tentang rasio HitRatio dan FalseAlarm-nya. Jika *training* hanya memilih beberapa fitur (misalnya N = 2), maka kemungkinan ada kesalahan pada data *training* yang digunakan.
4. Pada akhir setiap tahap, *classifier* disimpan ke file dan prosesnya bisa dihentikan dan di-restart (jika perangkat dimatikan).
5. Saat proses selesai kita akan menemukan sebuah file bernama classifier.xml di direktori *classifier*.

3.2.2. Perancangan Alur Proses Penggunaan Aplikasi

Pada tahap ini akan dijelaskan mengenai rancangan alur proses penggunaan aplikasi yang digunakan sebagai acuan dalam pembangunan aplikasi. Gambar 3-5 adalah penjelasan alur penggunaan aplikasi dalam bentuk *workflow.* Alur proses ini membantu memperlihatkan langkah demi langkah yang akan dilakukan pengguna sehingga terlihat jelas antara *input*, proses, dan *output.*



Gambar 3-5 Alur Penggunaan Aplikasi

Untuk alur proses penggunaan aplikasi, pengguna hanya perlu memasukkan video ke dalam aplikasi dan menjalankan aplikasi. Perlu diingat bahwa video yang digunakan harus direkam dengan sudut pandang tampak depan dari tengah jalan dan kondisi siang hari. Selama aplikasi melakukan deteksi, aplikasi akan memberikan tanda berupa segi empat merah pada objek yang dianggap sebagai kendaraan. Lalu, aplikasi akan menentukan jenis kendaraan yang terdeteksi berdasarkan luas segi empat yang digunakan. Ketika aplikasi sudah selesai melakukan deteksi jenis kendaraan, hasilnya akan dicatat dalam file .txt sehingga pengguna bisa mengetahui jenis kendaraan yang terdeteksi serta jumlah kendaraan yang terdeteksi.

BAB IV
IMPLEMENTASI

Bab ini membahas implementasi dari analisis dan perancangan sistem yang telah dibahas pada Bab III. Namun dalam penerapannya, rancangan tersebut dapat mengalami perubahan minor sewaktu-waktu apabila dibutuhkan.

4.1. Lingkungan Implementasi

Dalam implementasinya, lingkungan yang digunakan sama seperti yang dituliskan pada rancangan, yakni menggunakan beberapa perangkat pendukung sebagai berikut.

4.1.1. Lingkungan Implementasi Perangkat Keras

Perangkat keras yang digunakan dalam implementasi pengembangan aplikasi ini adalah Komputer. Spesifikasi komputer yang digunakan adalah laptop dengan prosesor Intel® Core™ i5-CPU dan RAM 4 GB.

4.1.2. Lingkungan Implementasi Perangkat Lunak

Penjelasan perangkat lunak yang digunakan dalam implementasi aplikasi ini adalah sebagai berikut:

- Microsoft Windows 10 Professional sebagai sistem operasi pada *notebook*.
- OpenCV 2.4.13 untuk memproses video inputan.

4.2. Implementasi OpenCV pada Python

Subbab ini membahas tentang implementasi OpenCV pada IDE Python. Proses implementasi dapat dilihat pada kode sumber 4-1. Penulis menggunakan OpenCV untuk melakukan input video dan proses deteksi jenis kendaraan pada video tersebut. Pada Python, library OpenCV terdapat pada cv2.

```
1.  import sys
2.  import cv2
3.  import numpy as np
```

Kode Sumber 4-1 Implementasi OpenCV pada IDE Python

4.3. Implementasi Wilayah Deteksi Objek

Subbab ini membahas tentang implementasi wilayah deteksi objek sebagai batas wilayah untuk mengenali objek kendaraan pada video. Selanjutnya akan dirinci dengan detail sebagai berikut.

4.3.1. Implementasi Deteksi ROI

Kode Sumber 4-2 merupakan implementasi untuk mendeteksi ROI. Proses ini dilakukan untuk menetapkan wilayah deteksi terhadap objek kendaraan pada video. Dalam proses ini, dilakukan pengecilan terhadap objek kendaraan dan penetapan batas wilayah deteksi objek kendaraan pada video. Setiap kali *classifier* mendeteksi objek kendaraan, objek tersebut akan diberi penanda batas waktu maksimal dengan nilai awal 0 (terdapat pada urutan keempat dalam *list* region, dijelaskan lebih lanjut pada subbab berikutnya). Nilai ini akan bertambah 1 selama objek kendaraan memasuki wilayah deteksi. Nilai ini sekaligus berguna sebagai parameter untuk mengetahui kecepatan objek kendaraan.

```
1.  def DetectROI(frame, car_cascade):
2.      scaledown = 2
3.      fheight, fwidth, fdepth = frame.shape
4.
5.      frame = cv2.resize(frame, (fwidth/scaledown, fh
    eight/scaledown))
6.      fheight, fwidth, fdepth = frame.shape
7.
8.      cars = car_cascade.detectMultiScale(frame, 1.2,
     1)
9.      newRegions = []
10.     minY = int(fheight*0.3)
11.
```

```
12.     #iterating ROI
13.     for (x,y,w,h) in cars:
14.             roi = [x,y,w,h]
15.             roiImage = frame[y:y+h, x:x+w]
16.             carwidth = roiImage.shape[0]
17.             if y > minY:
18.                 diffx = DiffLeftRight(roiImage)
19.                 diffy = round(DiffUpDown(roiImage))

20.
21.                 if diffx > 1600 and diffx < 3000 an
    d diffy > 12000:
22.                     rx,ry,rw,rh = roi
23.                     left = rx*scaledown
24.                     right = rx*scaledown + rw*scale
    down
25.                     top = ry*scaledown
26.                     bottom = ry*scaledown + rh*scal
    edown
27.
28.
29.                     if left >= 280 and left <= 1050
     and right >= 280 and right <= 1050 and top >= 280
    and top <= 500 and bottom >= 280 and bottom <= 500:

30.                         newRegions.append( [rx*scal
    edown,ry*scaledown,rw*scaledown,rh*scaledown,0] )
31.
32.     return newRegions
```

Kode Sumber 4-2 Deteksi ROI

4.3.2. Implementasi ROI Baru

Kode Sumber 4-3 adalah implementasi untuk menandai ROI baru. Tujuannya adalah untuk menandai objek kendaraan yang terdeteksi pada wilayah ROI. Jika jarak sebuah objek kendaraan dengan objek kendaraan lainnya kurang dari 75 piksel, kedua objek tersebut dianggap sebagai satu objek kendaraan.

```
1.  def NewROI(rx,ry,rw,rh,rect):
2.      for r in rect:
```

```
3.          if abs(r[0] - rx) < 75 and abs(r[1] - ry) <
    75:
4.              return False
5.      return True
```

Kode Sumber 4-3 Implementasi ROI Baru

4.4. Implementasi Deteksi Objek Kendaraan

Pada subbab ini dijelaskan implementasi proses deteksi objek kendaraan.

4.4.1. Implementasi Mengecek Kesimetrisan Horizontal

Kode Sumber 4-4 merupakan implementasi untuk mengecek kesimetrisan sebuah objek kendaraan. Sebuah objek dianggap sebagai kendaraan apabila bersifat secara simetris secara vertikal dan horizontal. Dalam mengecek kesimetrisan vertikal, digunakan rumus *mean squared error* (MSE) untuk mengetahui perbedaan sisi atas dan sisi bawah suatu objek.

```
1.  def DiffUpDown(car):
2.      height, width, depth = car.shape
3.      half = height/2
4.      top = car[0:half, 0:width]
5.      bottom = car[half:2*half, 0 : width]
6.      top = cv2.flip(top, 1)
7.      bottom = cv2.resize(bottom, (32, 64))
8.      top = cv2.resize(top, (32, 64))
9.      return (mse(top, bottom))
```

Kode Sumber 4-4 Mengecek Kesimetrisan Horizontal

4.4.2. Implementasi Mengecek Kesimetrisan Vertikal

Kode Sumber 4-5 merupakan implementasi untuk mengecek kesimetrisan sebuah objek kendaraan. Sebuah objek dianggap sebagai kendaraan apabila bersifat secara simetris secara vertikal dan horizontal. Dalam mengecek kesimetrisan horizontal,

digunakan rumus *mean squared error* (MSE) untuk mengetahui perbedaan sisi kiri dan sisi kanan suatu objek.

```
1.  def DiffLeftRight(car):
2.      height, width, depth = car.shape
3.      half = width/2
4.      left = car[0:height, 0:half]
5.      right = car[0:height, half:half + half-1]
6.      right = cv2.flip(right, 1)
7.      left = cv2.resize(left, (32, 64))
8.      right = cv2.resize(right, (32, 64))
9.      return (mse(left, right))
```

Kode Sumber 4-5 Mengecek Kesimetrisan Vertikal

4.4.3. Implementasi Rumus MSE

Kode Sumber 4-6 merupakan implementasi rumus MSE. MSE digunakan dalam proses mengecek kesimetrisan horizontal maupun vertikal suatu objek. Dalam hal ini, MSE membandingkan sisi kiri dan kanan (simetris horizontal) serta sisi atas dan bawah (simetris vertikal).

```
1.  def mse(car1, car2):
2.      err = np.sum((car1.astype("float") - car2.astyp
    e("float")) ** 2)
3.      err /= float(car1.shape[0] * car1.shape[1])
4.      return err
```

Kode Sumber 4-6 Implementasi MSE

4.5. Implementasi Program Utama

Pada subbab ini dijelaskan implementasi bagian utama program. Bagian ini meliputi memuat file XML, memasukkan video, serta menampilkan hasil deteksi pada jendela program dan file .txt.

4.5.1. Memuat file XML

Kode Sumber 4-7 merupakan kode untuk memuat file XML. File XML dimuat sebagai *template* untuk mencocokkan apakah sebuah objek memenuhi syarat sebagai kendaraan atau tidak. Di kode sumber ini, juga terdapat inisiasi sejumlah variable yang digunakan untuk ID kendaraan serta menghitung objek kendaraan yang terdeteksi berdasarkan jenisnya.

```
1.  def DetectCar(filename):
2.      rect = []
3.      rectCount = 0
4.      big = small= medium = 0
5.      biglist = []
6.      medlist = []
7.      smalist = []
8.      # XML classifier
9.      car_cascade = cv2.CascadeClassifier('cars.xml')
```

Kode Sumber 4-7 Memuat file XML

4.5.2. Memasukkan Video dan Memuat File .txt

Kode Sumber 4-8 merupakan implementasi untuk memasukkan video serta memuat file .txt. Video dimuat dengan menggunakan cv2.VideoCapture(). Untuk file .txt, file akan dimuat dengan menggunakan file = open(). File .txt digunakan untuk mencatat hasil deteksi jenis kendaraan serta total jumlah kendaraan yang terdeteksi.

```
1.  # insert video
2.  print("CarDetection Program\n")
3.  cap = cv2.VideoCapture(filename)
4.  if cap.isOpened():
5.      print("Reading video...")
6.      width = int(cap.get(cv2.cv.CV_CAP_PROP_FRAME_WI
    DTH))
7.      height = int(cap.get(cv2.cv.CV_CAP_PROP_FRAME_H
    EIGHT))
8.  else:
9.      print("Cannot read video. Check input")
```

```
    rval = False

try:
    file = open(filename + '_CarCount.txt','w')
    print("Writing to file...")

except:
     print("Cannot create or open file\n")
     sys.exit(0)

file.write("Number of vehicle(s) in video :\n")

roi = [0,0,0,0]
```

Kode Sumber 4-8 Memasukkan video dan Memuat File .txt

4.5.3. Membaca Video secara *Frame-by-frame*

Kode Sumber 4-9 merupakan implementasi untuk membaca video yang dimasukkan secara *frame-by-frame*. Kode sumber berikut juga meliputi pembuatan garis batas wilayah deteksi objek kendaraan. Sehingga *classifier* hanya melakukan deteksi objek kendaraan pada wilayah yang tersebut.

```
  while True:
        rval, frame = cap.read()
        cv2.line(frame, (width-815,height-
440), (width-430,height-440), (255, 0, 0), 3)
        cv2.line(frame, (width-990,height-
220), (width-290,height-220), (255, 0, 0), 3)
        fheight, fwidth, fdepth = frame.shape
```

Kode Sumber 4-9 Membaca Video secara *Frame-by-frame*

4.5.4. Mendeteksi Objek Kendaraan Beserta Jenisnya

Kode Sumber 4-10 merupakan implementasi untuk mendeteksi objek kendaraan serta mengenali jenisnya. Objek kendaraan yang terdeteksi akan diberi tanda berupa segi empat

berwarna merah. Kode sumber berikut juga meliputi penghitungan terhadap objek kendaraan yang dideteksi. Hasil penghitungan akan ditulis ke file .txt yang telah dimuat pada tahap sebelumnya.

```
1.          newRegions = DetectROI(frame, car_cascade)
2.          for region in newRegions:
3.              if NewROI(region[0],region[1],region[2],
    region[3],rect):
4.                  rect.append(region)
5.                  rectCount = rectCount + 1
6.                  if region[2] <= 132:
7.                      small = small + 1
8.                      print ("S%d. Small vehicle is de
    tected" % small)
9.                      print region[0]
10.                     print region[1]
11.                     print region[2]
12.                     print region[3]
13.                     print ("\n")
14.                     file.write("S%d. Small vehicle i
    s detected\n" % small)
15.                 elif region[2] <= 148 and region[2]
    >= 133:
16.                     medium = medium + 1
17.                     print ("M%d. Medium vehicle is d
    etected" % medium)
18.                     print region[0]
19.                     print region[1]
20.                     print region[2]
21.                     print region[3]
22.                     print ("\n")
23.                     file.write("M%d. Medium vehicle
    is detected\n" % medium)
24.                 else:
25.                     big = big + 1
26.                     print ("B%d. Big vehicle is dete
    cted" % big)
27.                     print region[0]
28.                     print region[1]
29.                     print region[2]
```

```
30.                     print region[3]
31.                     print ("\n")
32.                     file.write("B%d. Big vehicle is
    detected\n" % big)
33.
34.         for r in rect:
35.             cv2.rectangle(frame,(r[0],r[1]),(r[0]+r
    [2],r[1]+r[3]),(0,0,255),3)
```

Kode Sumber 4-10 Mendeteksi Objek Kendaraan Beserta Jenisnya

4.5.5. *Refresh* Batas Waktu Maksimal Objek Kendaraan

Kode Sumber 4-11 merupakan implementasi untuk melakukan *refresh* pada batas waktu maksimal objek kendaraan. Variabel ini terdapat pada urutan keempat dalam *list* region. Proses ini digunakan sebagai parameter untuk mengetahui kecepatan sebuah kendaraan. Dalam program ini, setiap kendaraan memiliki waktu maksimal sebesar 35 milidetik. Jika objek kendaraan melewati wilayah deteksi selama lebih dari 35 milidetik, objek tersebut akan dideteksi sebagai kendaraan untuk kedua kalinya. Setelah 35 milidetik, penanda segi empat akan dihilangkan dari jendela video.

```
1.  for region in rect:
2.       region[4] = region[4] + 1
3.       if region[4] > 35:
4.           rect.remove(region)
```

Kode Sumber 4-11 *Refresh* Batas Waktu Maksimal

4.5.6. Menampilkan Video pada Jendela Baru

Kode Sumber 4-12 merupakan implementasi untuk menampilkan video yang dimasukkan ke dalam aplikasi. Video ditampilkan pada sebuah jendela baru. Dalam jendela ini, pengguna dapat melihat proses penandaan objek yang dianggap sebagai kendaraan. Kode sumber ini juga memuat proses

pencatatan total kendaraan yang terdeteksi pada video ke dalam file .txt.

```
1.      # Display frames in a window
2.      cv2.imshow('CarDetection', frame)
3.
4.      # Wait for Esc key to stop
5.      if cv2.waitKey(33) == 27:
6.          break
7.
8.  file.write("\nTotal small vehicle(s) detected: %d"
    % small)
9.  file.write("\nTotal medium vehicle(s) detected: %d"
     % medium)
10. file.write("\nTotal big vehicle(s) detected: %d" %
    big)
11. file.write("\nTotal vehicle(s) detected: %d" % rect
    Count)
12. file.close()
13. cap.release()
14. cv2.destroyAllWindows()
```

Kode Sumber 4-12 Menampilkan Video pada Jendela Baru

BAB V
PENGUJIAN DAN EVALUASI

Bab ini membahas pengujian dan evaluasi pada aplikasi yang dikembangkan. Pengujian yang dilakukan adalah pengujian terhadap daftar fungsi aplikasi yang telah dijabarkan pada Bab III dan terhadap tujuan dibuatnya aplikasi ini, yakni melakukan deteksi jenis kendaraan dan menghitung objek kendaraan yang terdeteksi berdasarkan jenisnya.

5.1. Lingkungan Pengujian

Lingkungan pengujian sistem pada pengerjaan Tugas Akhir ini dilakukan pada lingkungan dan alat kakas sebagai berikut:

Prosesor	: Prosesor Intel® Core™ i5-CPU
RAM	: 4 GB
Jenis *Device*	: Personal Computer
Sistem Operasi	: Windows 10 Professional

5.2. Skenario Pengujian

Pada bagian ini akan dijelaskan tentang skenario pengujian yang dilakukan. Uji coba dilakukan untuk mengenali jenis kendaraan melalui input video. Terdapat beberapa skenario saat ujicoba, dikarenakan hasil *output* dipengaruhi skenario yang digunakan. Pada setiap skenario, penulis menggunakan 4 nilai batas waktu maksimal objek kendaraan untuk melewati wilayah deteksi yaitu 25, 30, 35, dan 40. Berikut ini adalah skenario pengujian yang diterapkan:

1. Skenario 1: Melakukan uji coba dengan kondisi jalan sepi.
   - Memasukkan video berformat .mp4 ke dalam aplikasi.
   - Aplikasi memuat file XML yang berisi fitur-fitur objek kendaraan.
   - Aplikasi membaca video secara *frame by frame*.
   - Aplikasi mendeteksi objek kendaraan menggunakan *classifier*.

- Hasil akan keluar sesuai dengan fitur objek kendaraan yang dikenali oleh aplikasi.
- Aplikasi memberikan tanda segi empat pada objek yang dianggap sebagai kendaraan.
- Aplikasi melakukan pengenalan jenis kendaraan.
- Hasil akan keluar sesuai dengan luas segi empat.
- Aplikasi mencatat ID dan jenis kendaraan pada file .txt.
- Hasil bisa berubah-ubah (tergantung jarak antar objek kendaraan dan nilai batas waktu maksimal).

2. Skenario 2: Melakukan uji coba dengan kondisi jalan normal.
   - Memasukkan video berformat .mp4 ke dalam aplikasi.
   - Aplikasi memuat file XML yang berisi fitur-fitur objek kendaraan.
   - Aplikasi membaca video secara *frame by frame*.
   - Aplikasi mendeteksi objek kendaraan menggunakan *classifier*.
   - Hasil akan keluar sesuai dengan fitur objek kendaraan yang dikenali oleh aplikasi.
   - Aplikasi memberikan tanda segi empat pada objek yang dianggap sebagai kendaraan.
   - Aplikasi melakukan pengenalan jenis kendaraan.
   - Hasil akan keluar sesuai dengan luas segi empat.
   - Aplikasi mencatat ID dan jenis kendaraan pada file .txt.
   - Hasil bisa berubah-ubah (tergantung jarak antar objek kendaraan dan nilai batas waktu maksimal).
3. Skenario 3: Melakukan uji coba dengan kondisi jalan padat.
   - Memasukkan video berformat .mp4 ke dalam aplikasi.
   - Aplikasi memuat file XML yang berisi fitur-fitur objek kendaraan.
   - Aplikasi membaca video secara *frame by frame*.
   - Aplikasi mendeteksi objek kendaraan menggunakan *classifier*.

- Hasil akan keluar sesuai dengan fitur objek kendaraan yang dikenali oleh aplikasi.
- Aplikasi memberikan tanda segi empat pada objek yang dianggap sebagai kendaraan.
- Aplikasi melakukan pengenalan jenis kendaraan.
- Hasil akan keluar sesuai dengan luas segi empat.
- Aplikasi mencatat ID dan jenis kendaraan pada file .txt.
- Hasil bisa berubah-ubah (tergantung jarak antar objek kendaraan dan nilai batas waktu maksimal).

Tabel 5-1 memperlihatkan skenario pengujian untuk tiga kondisi jalan.

Tabel 5-1 Skenario Pengujian

Kode Pengujian	Skenario Pengujian
SP-1	Pengujian dengan kondisi jalan sepi
SP-2	Pengujian dengan kondisi jalan normal
SP-3	Pengujian dengan kondisi jalan padat

5.2.1. Pengujian dengan Kondisi Jalan Sepi

Pengujian ini dimulai dengan pengguna melakukan input video ke aplikasi. Video yang digunakan adalah video rekaman lalu lintas yang direkam dengan kamera CCTV. Rincian skenario pengujian pada kasus penggunaan dapat dilihat pada Tabel 5-2.

Tabel 5-2 Tabel Pengujian dengan Kondisi Jalan Sepi

ID	SP-1
Nama	Mendeteksi jenis kendaraan dalam kondisi jalan sepi.
Tujuan Pengujian	Pengguna mengetahui jenis kendaraan pada video dan jumlah kendaraan yang terdeteksi.

Skenario	Mendeteksi jenis kendaraan pada video
Kondisi Awal	Pengguna membuka aplikasi CarDetection.
Data Uji	Video dengan durasi 20 detik.
Langkah Pengujian	1. Pengguna memasukkan video ke dalam aplikasi. 2. Pengguna menjalankan aplikasi CarDetection. 3. Aplikasi melakukan pengenalan jenis kendaraan pada video sesuai dengan durasi video. 4. Aplikasi mencatat hasil deteksi jenis kendaraan ke dalam file CarCount.txt. 5. Aplikasi perangkat bergerak akan melakukan proses. 6. Aplikasi akan menampilkan hasil berupa *pop-up.*
Hasil Yang Diharapkan	Tertampil detail jenis kendaraan serta jumlah kendaraan pada file CarCount.txt.
Hasil Yang Didapatkan	Detail jenis kendaraan dan jumlah kendaraan.
Hasil Pengujian	Berhasil.
Kondisi Akhir	Aktor mendapatkan file .txt yang berisi ID serta jenis kendaraan yang terdeteksi dari video.

5.2.2. Pengujian dengan Kondisi Jalan Normal

Pengujian ini dimulai dengan pengguna melakukan input video ke aplikasi. Video yang digunakan adalah video rekaman lalu lintas yang direkam dengan kamera CCTV. Rincian skenario pengujian pada kasus penggunaan dapat dilihat pada Tabel 5-3.

Tabel 5-3 Tabel Pengujian dengan Kondisi Jalan Normal

ID	**SP-2**
Nama	Mendeteksi jenis kendaraan dalam kondisi jalan padat.
Tujuan Pengujian	Pengguna mengetahui jenis kendaraan pada video dan jumlah kendaraan yang terdeteksi..
Skenario	Mendeteksi jenis kendaraan pada video.
Kondisi Awal	Pengguna membuka aplikasi CarDetection.
Data Uji	Video dengan durasi 27 detik.
Langkah Pengujian	1. Pengguna memasukkan video ke dalam aplikasi. 2. Pengguna menjalankan aplikasi CarDetection. 3. Aplikasi melakukan pengenalan jenis kendaraan pada video sesuai dengan durasi video. 4. Aplikasi mencatat hasil deteksi jenis kendaraan ke dalam file CarCount.txt. 5. Aplikasi perangkat bergerak akan melakukan proses. 6. Aplikasi akan menampilkan hasil berupa *pop-up.*

Hasil Yang Diharapkan	Tertampil detail jenis kendaraan serta jumlah kendaraan pada file CarCount.txt.
Hasil Yang Didapatkan	Detail jenis kendaraan dan jumlah kendaraan.
Hasil Pengujian	Berhasil.
Kondisi Akhir	Aktor mendapatkan file .txt yang berisi ID serta jenis kendaraan yang terdeteksi dari video.

5.2.3. Pengujian dengan Kondisi Jalan Padat

Pengujian ini dimulai dengan pengguna melakukan input video ke aplikasi. Video yang digunakan adalah video rekaman lalu lintas yang direkam dengan kamera CCTV. Rincian skenario pengujian pada kasus penggunaan dapat dilihat pada Tabel 5-4.

Tabel 5-4 Tabel Pengujian dengan Kondisi Jalan Padat

ID	**SP-3**
Nama	Mendeteksi jenis kendaraan dalam kondisi jalan sangat padat.
Tujuan Pengujian	Pengguna mengetahui jenis kendaraan pada video dan jumlah kendaraan yang terdeteksi..
Skenario	Mendeteksi jenis kendaraan pada video.
Kondisi Awal	Pengguna membuka aplikasi CarDetection.
Data Uji	Video dengan durasi 24 detik.
Langkah Pengujian	1. Pengguna memasukkan video ke dalam aplikasi. 2. Pengguna menjalankan aplikasi CarDetection. 3. Aplikasi melakukan pengenalan jenis kendaraan pada video sesuai dengan durasi video.

	4. Aplikasi mencatat hasil deteksi jenis kendaraan ke dalam file CarCount.txt. 5. Aplikasi perangkat bergerak akan melakukan proses. 6. Aplikasi akan menampilkan hasil berupa *pop-up*.
Hasil Yang Diharapkan	Tertampil detail jenis kendaraan serta jumlah kendaraan pada file CarCount.txt.
Hasil Yang Didapatkan	Detail jenis kendaraan dan jumlah kendaraan.
Hasil Pengujian	Berhasil.
Kondisi Akhir	Aktor mendapatkan file .txt yang berisi ID serta jenis kendaraan yang terdeteksi dari video.

5.3. Akurasi Pengujian Fungsionalitas

Pada subbab ini akan diberikan hasil evaluasi dari pengujian-pengujian yang telah dilakukan. Evaluasi yang diberikan meliputi evaluasi pengujian daftar fungsi sesuai dengan modul yang terdapat aplikasi.

Pada pengujian, terdapat tiga kondisi jalan yaitu sepi, padat, dan sangat padat. Ketiga kondisi ini mewakili kondisi nyata yang terjadi di jalan raya pada umumnya. Tabel 5-5 memperlihatkan hasil pengujian dari tiga kondisi jalan raya, jenis kendaraan yang terdeteksi, mana yang sesuai dengan definisi, mana yang tidak sesuai dengan definisi, dan mana yang tidak terdapat pada video inputan.

Tabel 5-5 Hasil Deteksi Jenis Kendaraan

Kondisi Jalan	Jenis Kendaraan		
	Kecil	**Sedang**	**Besar**
Sepi	V	V	V
Normal	V	V	V
Padat	V	V	V

Keterangan simbol:
V = Sesuai dan dipakai.
X = Tidak dipakai karena tidak ditemukan.

Akurasi pengenalan jenis kendaraan dihitung menggunakan Persamaan 5-1 yang merupakan perhitungan dengan cara sebagai berikut:

$$Acc = \frac{V_{detect}}{V_{real}} * 100\%$$

Persamaan (5-1)

Di mana V_{detect} adalah banyaknya suatu jenis kendaraan yang terdeteksi dan V_{real} adalah jumlah sebenarnya dari suatu jenis kendaraan. V_{real} diperoleh dengan cara menghitung secara manual setiap jenis kendaraan.

5.3.1. Hasil Pengujian Kondisi Jalan Sepi

Berikut ini adalah hasil pegujian yang diperoleh untuk kondisi jalan sepi. Dalam pengujian ini, penulis menggunakan 4 nilai batas waktu maksimal yaitu 25, 30, 35, dan 40. Jika waktu yang diperlukan objek untuk melewati wilayah deteksi melebihi nilai tersebut, objek tersebut akan terdeteksi sebagai kendaraan lebih dari sekali. Sehingga, penghitungan kendaraan akan menjadi tidak akurat. Hasil yang diperoleh dilampirkan pada tabel-tabel berikut.

Tabel 5-6 Hasil Pengujian Kondisi Jalan Sepi (batas waktu maksimal = 25)

Aktual	Pengujian			Akurasi
	Kendaraan Kecil	Kendaraan Sedang	Kendaraan Besar	
Kendaraaan Kecil	1	0	0	100%
Kendaraan Sedang	0	1	2	33%
Kendaraan Besar	0	0	1	100%

Tabel 5-7 Hasil Pengujian Kondisi Jalan Sepi (batas waktu maksimal = 30)

Aktual	Pengujian			Akurasi
	Kendaraan Kecil	Kendaraan Sedang	Kendaraan Besar	
Kendaraaan Kecil	1	0	0	100%
Kendaraan Sedang	0	1	2	33%
Kendaraan Besar	0	0	1	100%

Tabel 5-8 Hasil Pengujian Kondisi Jalan Sepi (batas waktu maksimal = 35)

Aktual	Pengujian			Akurasi
	Kendaraan Kecil	Kendaraan Sedang	Kendaraan Besar	
Kendaraaan Kecil	1	0	0	100%
Kendaraan Sedang	0	1	2	33%
Kendaraan Besar	0	0	1	100%

Tabel 5-9 Hasil Pengujian Kondisi Jalan Sepi (batas waktu maksimal = 40)

Aktual	Pengujian			Akurasi
	Kendaraan Kecil	Kendaraan Sedang	Kendaraan Besar	
Kendaraaan Kecil	1	0	0	100%
Kendaraan Sedang	0	1	2	33%
Kendaraan Besar	0	0	1	100%

Dari data di atas, maka dilakukan penghitungan akurasi dengan Persamaan 5-1. Jumlah asli kendaraan kecil, sedang, dan besar secara berurutan adalah 1, 3, dan 1. Dalam pengujian ini, terdapat kesalahan deteksi yaitu kendaraan sedang terdeteksi sebagai kendaraan besar. Hal ini disebabkan karena sudut pandang objek kendaraan terhadap kamera yang tidak pas. Sehingga objek terlihat lebih besar dari ukuran sebenarnya. Untuk kondisi jalan sepi, akurasi rata-rata adalah 77.8% dan nilai batas waktu maksimal yang ideal adalah 35 dan 40.

5.3.2. Hasil Pengujian Kondisi Jalan Normal

Berikut ini adalah hasil pegujian yang diperoleh untuk kondisi jalan padat. Dalam pengujian ini, penulis menggunakan 4 nilai batas waktu maksimal yaitu 25, 30, 35, dan 40. Jika waktu yang diperlukan objek untuk melewati wilayah deteksi melebihi nilai tersebut, objek tersebut akan terdeteksi sebagai kendaraan lebih dari sekali. Sehingga, penghitungan kendaraan akan menjadi tidak akurat. Sehingga, penghitungan kendaraan akan menjadi tidak akurat. Hasil yang diperoleh dilampirkan pada tabel-tabel berikut.

Tabel 5-10 Hasil Pengujian Kondisi Jalan Normal (batas waktu maksimal = 25)

Aktual	Pengujian			Akurasi
	Kendaraan Kecil	Kendaraan Sedang	Kendaraan Besar	
Kendaraaan Kecil	3	0	0	30%
Kendaraan Sedang	0	2	1	25%
Kendaraan Besar	0	0	3	75%

Tabel 5-11 Hasil Pengujian Kondisi Jalan Normal (batas waktu maksimal = 30)

Aktual	Pengujian			Akurasi
	Kendaraan Kecil	Kendaraan Sedang	Kendaraan Besar	
Kendaraaan Kecil	3	0	0	30%
Kendaraan Sedang	0	2	1	25%
Kendaraan Besar	0	0	3	75%

Tabel 5-12 Hasil Pengujian Kondisi Jalan Normal (batas waktu maksimal = 35)

Aktual	Pengujian			Akurasi
	Kendaraan Kecil	Kendaraan Sedang	Kendaraan Besar	
Kendaraaan Kecil	3	0	0	30%
Kendaraan Sedang	0	3	0	37.5%
Kendaraan Besar	0	0	3	75%

Tabel 5-13 Hasil Pengujian Kondisi Jalan Normal (batas waktu maksimal = 40)

Aktual	Pengujian			Akurasi
	Kendaraan Kecil	Kendaraan Sedang	Kendaraan Besar	
Kendaraaan Kecil	3	0	0	30%
Kendaraan Sedang	0	3	0	37.5%
Kendaraan Besar	0	0	3	75%

Dari data di atas, maka dilakukan penghitungan akurasi dengan Persamaan 5-1. Jumlah asli kendaraan kecil, sedang, dan besar secara berurutan adalah 10, 8, dan 4. Akurasi deteksi untuk kendaraan kecil, sedang, dan besar pada batas waktu maksimal 25 dan 30 adalah 30%, 25%, dan 75%. Akurasi rata-rata untuk batas waktu maksimal 25 dan 30 adalah 43.3%. Terdapat kesalahan deteksi pada batas waktu maksimal 25 dan 30 yaitu kendaraan sedang terdeteksi sebagai kendaraan besar. Ini disebabkan karena objek kendaraan bergerak lebih lambat dari nilai batas waktu maksimal yang ditetapkan. Dan juga, jarak antar objek kendaraan turut mempengaruhi akurasi deteksi. Pada pengujian ini, jarak antar kendaraan cenderung lebih rapat dibandingkan dengan pengujian kondisi jalan sepi. Untuk batas waktu maksimal 35 dan 40, hasil yang diperoleh untuk kendaraan kecil, sedang, dan besar secara berurutan yaitu 25%, 37.5%, dan 75%. Tidak ditemukan kesalahan kesalahan penggolongan jenis kendaraan pada batas waktu maksimal 35 dan 40. Untuk kondisi jalan normal, nilai batas waktu maksimal yang ideal adalah 35 dan 40 dengan akurasi rata-rata 47.5%.

5.3.3. Hasil Pengujian Kondisi Jalan Padat

Berikut ini adalah hasil pegujian yang diperoleh untuk kondisi jalan padat. Dalam pengujian ini, penulis menggunakan 4 nilai batas waktu maksimal yaitu 25, 30, 35, dan 40. Jika waktu yang diperlukan objek untuk melewati wilayah deteksi melebihi nilai tersebut, objek tersebut akan terdeteksi sebagai kendaraan

lebih dari sekali. Sehingga, penghitungan kendaraan akan menjadi tidak akurat. Hasil yang diperoleh dilampirkan pada tabel-tabel berikut.

Tabel 5-14 Hasil Pengujian Kondisi Jalan Padat (batas waktu maksimal = 25)

Aktual	Pengujian			Akurasi
	Kendaraan Kecil	Kendaraan Sedang	Kendaraan Besar	
Kendaraaan Kecil	1	0	1	11.1%
Kendaraan Sedang	0	2	1	25%
Kendaraan Besar	0	0	3	37.5%

Tabel 5-15 Hasil Pengujian Kondisi Jalan Padat (batas waktu maksimal=30)

Aktual	Pengujian			Akurasi
	Kendaraan Kecil	Kendaraan Sedang	Kendaraan Besar	
Kendaraaan Kecil	1	0	1	11.1%
Kendaraan Sedang	0	2	1	25%
Kendaraan Besar	0	0	3	37.5%

Tabel 5-16 Hasil Pengujian Kondisi Jalan Padat (batas waktu maksimal = 35)

Aktual	Pengujian			Akurasi
	Kendaraan Kecil	Kendaraan Sedang	Kendaraan Besar	
Kendaraaan Kecil	2	0	1	22.2%
Kendaraan Sedang	0	2	1	25%
Kendaraan Besar	0	0	3	37.5%

Tabel 5-17 Hasil Pengujian Kondisi Jalan Padat (batas waktu maksimal = 40)

Aktual	Pengujian			Akurasi
	Kendaraan Kecil	Kendaraan Sedang	Kendaraan Besar	
Kendaraaan Kecil	2	0	1	22.2%
Kendaraan Sedang	0	1	2	12.5%
Kendaraan Besar	0	0	3	37.5%

Dari data di atas, maka dilakukan penghitungan akurasi dengan Persamaan 5-1. Jumlah asli kendaraan kecil, sedang, dan besar secara berurutan adalah 9, 8, dan 8. Akurasi deteksi untuk kendaraan kecil, sedang, dan besar pada batas waktu maksimal 25 dan 30 adalah 11.1%, 25%, dan 37.5%. Akurasi rata-rata untuk batas waktu maksimal 25 dan 30 adalah 24.5%. Untuk batas waktu maksimal 35, akurasi deteksi kendaraan kecil, sedang, dan besar secara berurutan adalah 22.2%, 25%, dan 37.5%. Akurasi rata-rata untuk nilai batas waktu maksimal 35 adalah 28.2%. Untuk batas waktu maksimal 40, akurasi deteksi kendaraan kecil, sedang, dan besar secara berurutan adalah 22.2%, 12.5%, dan 37.5%. Akurasi rata-rata untuk nilai batas waktu maksimal 40 adalah 24.1%.

Dalam pengujian ini, terdapat kesalahan penggolongan jenis kendaraan yaitu kendaraan sedang terdeteksi sebagai kendaraan besar. Ada juga kendaraan kecil yang terdeteksi sebagai kendaraan besar. Dalam pengujian ini, kedua kejadian ini dipengaruhi oleh dua faktor. Faktor pertama adalah jarak antar objek kendaraan. Pada pengujian ini, jarak antar kendaraan cenderung lebih rapat dibandingkan dengan pengujian kondisi jalan sepi. Kemudian, faktor kedua adalah sudut pandang kamera. Dalam pengujian ini, terdapat mobil kecil yang terlihat seperti mobil besar. Ini disebabkan karena objek tersebut tersorot kamera dari sisi samping (tidak tepat dari tengah). Akibatnya, objek terlihat lebih besar dari ukuran sebenarnya. Untuk kondisi jalan padat, nilai

batas waktu maksimal yang ideal adalah 35 dengan akurasi rata-rata 28.2%.

5.4. Pengujian dalam Skenario Lain

Untuk skenario lainnya, penulis mencoba memperkecil besaran wilayah deteksi yang digunakan. Wilayah deteksi diperkecil menjadi seukuran 76400 piksel, dari ukuran semula sebesar 119350 piksel. Penulis menggunakan nilai batas waktu maksimal 35 dalam skenario ini. Berikut ini adalah hasil dari pengujian yang diperoleh.

Tabel 5-18 Kondisi Jalan Sepi

Aktual	**Pengujian**			**Akurasi**
	Kendaraan Kecil	Kendaraan Sedang	Kendaraan Besar	
Kendaraaan Kecil	1	0	0	100%
Kendaraan Sedang	0	0	1	0%
Kendaraan Besar	0	0	1	100%

Tabel 5-19 Kondisi Jalan Normal

Aktual	**Pengujian**			**Akurasi**
	Kendaraan Kecil	Kendaraan Sedang	Kendaraan Besar	
Kendaraaan Kecil	1	1	1	10%
Kendaraan Sedang	1	1	1	12.5%
Kendaraan Besar	0	2	1	25%

Tabel 5-20 Kondisi Jalan Padat

Aktual	Pengujian			Akurasi
	Kendaraan Kecil	Kendaraan Sedang	Kendaraan Besar	
Kendaraaan Kecil	0	0	0	0%
Kendaraan Sedang	0	1	2	12.5%
Kendaraan Besar	0	0	2	25%

Untuk kondisi jalan sepi, akurasi rata-rata yang diperoleh adalah 66.7%. Untuk kondisi jalan normal, akurasi rata-rata yang diperoleh adalah 15.8%. Sedangkan untuk kondisi jalan padat, akurasi rata-rata yang diperoleh adalah 12.5%. Dalam pengujian pada skenario ini, terdapat lebih banyak kesalahan deteksi dibandingkan dengan pengujian dalam skenario-skenario sebelumnya. Nilai akurasi rata-rata mengalami penurunan, jika dibandingkan dengan nilai akurasi rata-rata dengan batas waktu yang sama namun dengan besaran wilayah deteksi yang lebih besar. Dapat dilihat bahwa akurasi yang diperoleh relatif kurang baik jika dibandingkan dengan besaran wilayah deteksi senilai 76400 piksel. Sehingga, penulis memutuskan untuk menggunakan besaran wilayah deteksi senilai 119350 piksel dalam pengembangan program CarDetection ini.

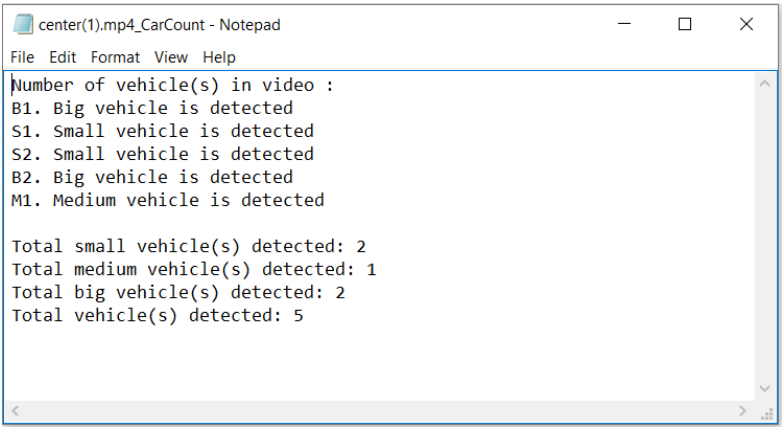
5.5. Evaluasi Pengujian Fungsionalitas

Rangkuman mengenai hasil pengujian fungsionalitas dapat dilihat pada Tabel 5-18. Berdasarkan data pada tabel tersebut, semua skenario pengujian berhasil dan program berjalan dengan baik. Sehingga bisa ditarik kesimpulan bahwa fungsionalitas dari aplikasi telah dapat bekerja sesuai dengan yang diharapkan.

Tabel 5-21 Rangkuman Hasil Pengujian

Kode Pengujian	Skenario Pengujian	Hasil
SP-F1	Mendeteksi objek kendaraan pada video	Berhasil
SP-F2	Mendeteksi jenis kendaraan pada video	Berhasil
SP-F3	Mencatat hasil deteksi pada file .txt	Berhasil

Gambar 5-1 adalah contoh hasil tampilan *output* berupa file .txt yang dibuka oleh pengguna. Pada file tersebut, terdapat ID kendaraan, jenis kendaraan yang terdeteksi, serta total jumlah kendaraan yang terdeteksi berdasarkan jenisnya.



Gambar 5-1 Contoh isi file .txt yang dihasilkan

[Halaman ini sengaja dikosongkan]

BAB VI
KESIMPULAN DAN SARAN

Pada bab ini akan diberikan kesimpulan yang diperoleh selama pengerjaan Tugas Akhir dan saran mengenai pengembangan yang dapat dilakukan terhadap Tugas Akhir ini di masa yang akan datang.

6.1. Kesimpulan

Dari hasil pengamatan selama proses perancangan, implementasi, dan pengujian perangkat lunak yang dilakukan, dapat diambil kesimpulan sebagai berikut:

1. Haar-like feature dapat digunakan untuk melakukan deteksi objek kendaraan.
2. Jenis kendaraan dapat digolongkan berdasarkan luas segi empat penanda objek kendaraan.
3. Batas waktu maksimal yang optimal untuk tiap objek kendaraan dalam melewati wilayah deteksi adalah 35 milidetik. Sehingga mengurangi risiko objek kendaraan yang sama terdeteksi lebih dari sekali.
4. Tingkat akurasi rata-rata untuk tiga kondisi jalan yang berbeda (sepi, normal, padat) adalah 77.8%, 47.5%, dan 28.2%.

6.2. Saran

Berikut merupakan beberapa saran untuk pengembangan sistem di masa yang akan datang. Saran-saran ini didasarkan pada hasil perancangan, implementasi dan pengujian yang telah dilakukan:

1. Memperkaya fitur jenis kendaraan lain pada file XML yang digunakan sebagai *template* untuk mendeteksi objek kendaraan. Sehingga program tetap bisa mengenali objek kendaraan jenis baru di masa depan.
2. Penambahan kamera dari sudut pandang lain. Sehingga akurasi deteksi dapat meningkat.

3. Menambahkan fitur untuk menyambungkan program dengan kamera. Sehingga program mampu melakukan deteksi jenis kendaraan secara *real-time*.
4. Penambahan fitur untuk menuliskan hasil deteksi ke dalam file dalam format lain (misalkan PDF atau spreadsheet).

DAFTAR PUSTAKA

[1] Intel Corporation, "OpenCV," Itseez, June 2000. [Online]. Available: http://opencv.org/. [Accessed 4 December 2016].

[2] Intel Corporation, "OpenCV," Itseez, June 2000. [Online]. Available: http://opencv.org/platforms/. [Accessed 5 December 2016].

[3] P. Viola and M. Jones, "Rapid Object Detection using A Boosted Cascade of Simple Features," in *Conference on Computer Vision and Pattern Recognition CVPR 2001*, Hawaii, 2001.

[4] P. Viola and M. J. Jones, "Robust Real-time Face Detection," *International Journal of Computer Vision*, vol. 57, no. 2, pp. 137-154, 2004.

[5] R. Lienhart and J. Maydt, "An Extended Set of Haar-like Features for Rapid Object Detection," in *International Conference on Image Processing 2002*, New York, 2002.

[6] Office for Mathematics, Science, and Technology Education University of Illinois, "Mean Square Error," Office for Mathematics, Science, and Technology Education (MSTE), Champaign, 2004.

[7] C. Papageorgiou and T. Poggio, "A Trainable System for Object Detection," *International Journal of Computer Vision*, vol. 38, no. 1, pp. 15-33, 2000.

[8] V. K. Narayanan, "Vision Based Robust Vehicle Detection and Tracking VIA Active Learning," University of Florida, Gainesville, 2013.

[9] Badan Pengembangan dan Pembinaan Bahasa, "KBBI Daring," Kementerian Pendidikan dan Kebudayaan Indonesia, 2016. [Online]. Available:

https://kbbi.kemdikbud.go.id/entri/kendaraan. [Accessed 9 May 2017].

[10] M. Oliveira and V. Santos, "Automatic Detection of Cars in Real Roads using Haar-like Features," *Department of Mechanical Engineering, University of Aveiro,* vol. 3810, 2008.

[11] S. Han, Y. Han and H. Hahn, "Vehicle Detection Method using Haar-like Feature on Real Time System," *World Academy of Science, Engineering and Technology,* vol. 59, pp. 455-459, 2009.

[12] S. Sivaraman and M. M. Trivedi, "A General Active-Learning Framework for on-road Vehicle Recognition and Tracking," *IEEE Transactions on Intelligent Transportation Systems,* vol. 11, no. 2, pp. 267-276, 2010.

[13] T. Ball, "Coding Robin," Coding Robin, 22 July 2013. [Online]. Available: http://coding-robin.de/2013/07/22/train-your-own-opencv-haar-classifier.html. [Accessed 23 May 2017].

[14] N. Seo, "Tutorial: OpenCV haartraining (Rapid Object Detection With A Cascade of Boosted Classifiers Based on Haar-like Features) - Naotoshi Seo," Sonots, 16 October 2008. [Online]. Available: http://note.sonots.com/SciSoftware/haartraining.html. [Accessed 23 May 2017].

LAMPIRAN A

LAMPIRAN KODE SUMBER

```
1.  import sys
2.  import cv2
3.  import numpy as np
4.
5.  x = y = w = h = 0
6.
7.  def DiffUpDown(car):
8.      height, width, depth = car.shape
9.      half = height/2
10.     top = car[0:half, 0:width]
11.     bottom = car[half:2*half, 0 : width]
12.     top = cv2.flip(top, 1)
13.     bottom = cv2.resize(bottom, (32, 64))
14.     top = cv2.resize(top, (32, 64))
15.     return (mse(top, bottom))
16.
17. def DiffLeftRight(car):
18.     height, width, depth = car.shape
19.     half = width/2
20.     left = car[0:height, 0:half]
21.     right = car[0:height, half:half+half-1]
22.     right = cv2.flip(right, 1)
23.     left = cv2.resize(left, (32, 64))
24.     right = cv2.resize(right, (32, 64))
25.     return (mse(left, right))
26.
27. def mse(car1, car2):
28.     err = np.sum((car1.astype("float") - car2.astyp
    e("float")) ** 2)
29.     err /= float(car1.shape[0] * car1.shape[1])
30.     return err
31.
32. def NewROI(rx,ry,rw,rh,rect):
33.     for r in rect:
34.         if abs(r[0] - rx) < 75 and abs(r[1] - ry) <
     75:
35.             return False
36.     return True
```

```
37.
38. def DetectROI(frame, car_cascade):
39.     scaledown = 2
40.     fheight, fwidth, fdepth = frame.shape
41.
42.     #resize
43.     frame = cv2.resize(frame, (fwidth/scaledown, fh
    eight/scaledown))
44.     fheight, fwidth, fdepth = frame.shape
45.
46.     #haar
47.     cars = car_cascade.detectMultiScale(frame, 1.2,
     1)
48.     newRegions = []
49.     minY = int(fheight*0.3)
50.
51.     #iterating ROI
52.     for (x,y,w,h) in cars:
53.             roi = [x,y,w,h]
54.             roiImage = frame[y:y+h, x:x+w]
55.             carwidth = roiImage.shape[0]
56.             if y > minY:
57.                 diffx = DiffLeftRight(roiImage)
58.                 diffy = round(DiffUpDown(roiImage))

59.
60.                 if diffx > 1600 and diffx < 3000 an
    d diffy > 12000:
61.                     rx,ry,rw,rh = roi
62.                     left = rx*scaledown
63.                     right = rx*scaledown + rw*scale
    down
64.                     top = ry*scaledown
65.                     bottom = ry*scaledown + rh*scal
    edown
66.
67.
68.                     if left >= 280 and left <= 1050
     and right >= 280 and right <= 1050 and top >= 280
    and top <= 500 and bottom >= 280 and bottom <= 500:
     #center.mp4
```

```
69.                         newRegions.append( [rx*scaledown,ry*scaledown,rw*scaledown,rh*scaledown,0] ) # 0 utk batas waktu max tiap kendaraan
70.
71.     return newRegions
72.
73. def DetectCar(filename):
74.     rect = []
75.     rectCount = 0
76.     big = small= medium = 0
77.
78.     # XML classifier
79.     car_cascade = cv2.CascadeClassifier('cars.xml')

80.     # insert video
81.     cap = cv2.VideoCapture(filename)
82.     if cap.isOpened():
83.         print("Reading video...")
84.         width = int(cap.get(cv2.cv.CV_CAP_PROP_FRAME_WIDTH))
85.         height = int(cap.get(cv2.cv.CV_CAP_PROP_FRAME_HEIGHT))
86.     else:
87.         print("Cannot read video. Check input")
88.         rval = False
89.
90.     try:
91.         file = open(filename + '_CarCount.txt','w')
92.         print("Writing to file...")
93.
94.     except:
95.          print("Cannot create or open file\n")
96.          sys.exit(0)
97.
98.     file.write("Number of vehicle(s) in video :\n")

99.
100.          roi = [0,0,0,0]
101.
102.          while True:
103.              # reads frames from a video
104.              rval, frame = cap.read()
```

```
105.                cv2.line(frame, (width-815,height-
    440), (width-430,height-440), (255, 0, 0), 3)
106.                cv2.line(frame, (width-990,height-
    220), (width-290,height-220), (255, 0, 0), 3)
107.
108.                fheight, fwidth, fdepth = frame.shape

109.                newRegions = DetectROI(frame, car_cas
    cade)
110.                for region in newRegions:
111.                    if NewROI(region[0],region[1],reg
    ion[2],region[3],rect):
112.                        rect.append(region)
113.                        rectCount = rectCount + 1
114.                        #identify vehicle type
115.                        if region[2] <= 132:
116.                            small = small + 1
117.                            print ("S%d. Small vehicl
    e is detected" % small)
118.                            print region[0]
119.                            print region[1]
120.                            print region[2]
121.                            print region[3]
122.                            print ("\n")
123.                            file.write("S%d. Small ve
    hicle is detected\n" % small)
124.                        elif region[2] <= 148 and reg
    ion[2] >= 133:
125.                            medium = medium + 1
126.                            print ("M%d. Medium vehic
    le is detected" % medium)
127.                            print region[0]
128.                            print region[1]
129.                            print region[2]
130.                            print region[3]
131.                            print ("\n")
132.                            file.write("M%d. Medium v
    ehicle is detected\n" % medium)
133.                        else:
134.                            big = big + 1
135.                            print ("B%d. Big vehicle
    is detected" % big)
136.                            print region[0]
```

```
137.                             print region[1]
138.                             print region[2]
139.                             print region[3]
140.                             print ("\n")
141.                             file.write("B%d. Big vehi
    cle is detected\n" % big)
142.
143.                 for r in rect:
144.                     cv2.rectangle(frame,(r[0],r[1]),
    (r[0]+r[2],r[1]+r[3]),(0,0,255),3)
145.
146.                 for region in rect: #max time utk tia
    p mobil
147.                     region[4] = region[4] + 1
148.                     if region[4] > 35:
149.                         rect.remove(region)
150.
151.                 # Display frames in a window
152.                 cv2.imshow('CarDetection', frame)
153.
154.                 # Wait for Esc key to stop
155.                 if cv2.waitKey(33) == 27:
156.                     break
157.
158.             file.write("\nTotal small vehicle(s) dete
    cted: %d" % small)
159.             file.write("\nTotal medium vehicle(s) det
    ected: %d" % medium)
160.             file.write("\nTotal big vehicle(s) detect
    ed: %d" % big)
161.             file.write("\nTotal vehicle(s) detected:
    %d" % rectCount)
162.             file.close()
163.             cap.release()
164.             cv2.destroyAllWindows()
165.
166.         print("CarDetection Program\n")
167.         filename = raw_input("File name (with extens
    ion): ")
168.         DetectCar(filename)
```

Kode Sumber A.0-1 Program CarDetection

[Halaman ini sengaja dikosongkan]

BIODATA PENULIS

Alvin Lazaro, lahir pada tanggal 21 Desember 1995 di Jakarta. Penulis menempuh pendidikan perguruan tinggi di Institut Teknologi Sepuluh Nopember Surabaya di Jurusan Teknik Informatika Fakultas Teknologi Informasi pada tahun 2013. Penulis terlibat aktif dalam Unit Kegiatan Mahasiswa (UKM) badminton ITS sejak tahun 2013 hingga 2014. Dalam melakukan pengerjaan Tugas Akhir, penulis memiliki ketertarikan pada bidang Dasar dan Terapan Komputasi (DTK). Untuk menghubungi penulis dapat melalui *email*: lazaro.alvin13@mhs.if.its.ac.id.