TUGAS AKHIR – K141502

DETEKSI KECEPATAN KENDARAAN BERJALAN DI JALAN MENGGUNAKAN OPENCV

ANDREW
NRP 5113100032

Dosen Pembimbing
Prof. Ir. Joko Lianto Buliali, M.Sc., Ph.D.
Arya Yudhi Wijaya, S.Kom., M.Kom.

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

[Halaman ini sengaja dikosongkan]

TUGAS AKHIR – K141502

DETEKSI KECEPATAN KENDARAAN BERJALAN DI JALAN MENGGUNAKAN OPENCV

ANDREW
NRP 5113100032

Dosen Pembimbing
Prof. Ir. Joko Lianto Buliali, M.Sc., Ph.D.
Arya Yudhi Wijaya, S.Kom., M.Kom.

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

[Halaman ini sengaja dikosongkan]

FINAL PROJECT – K141502

VEHICLE SPEED DETECTION ON THE ROAD USING OPENCV

ANDREW
NRP 5113100032

Supervisor
Prof. Ir. Joko Lianto Buliali, M.Sc., Ph.D.
Arya Yudhi Wijaya, S.Kom., M.Kom.

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

[Halaman ini sengaja dikosongkan]

LEMBAR PENGESAHAN

DETEKSI KECEPATAN KENDARAAN BERJALAN 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:
ANDREW
NRP: 5113 100 032

Disetujui oleh Dosen Pembimbing Tugas Akhir:

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

Arya Yudhi Wijaya, S.Kom, M.Kom.
NIP: 19840904 201012 1 002 ..... (pembimbing 2)

SURABAYA
JULI 2017

[Halaman ini sengaja dikosongkan]

DETEKSI KECEPATAN KENDARAAN BERJALAN DI JALAN MENGGUNAKAN OPENCV

Nama Mahasiswa	:	Andrew
NRP	:	5113 100 032
Jurusan	:	Teknik Informatika FTIf-ITS
Dosen Pembimbing 1	:	Prof. Ir. Joko Lianto, M.Sc., Ph.D.
Dosen Pembimbing 2	:	Arya Yudhi Wijaya, S.Kom., M.Kom.

ABSTRAKSI

Saat ini, di berbagai kota telah dipasang CCTV pada setiap ruas jalan. Dari CCTV, dapat diketahui kondisi lalu lintas, namun tidak dapat diketahui kecepatan setiap kendaraaan. Oleh karena itu, dibuat perangkat lunak yang dapat mendeteksi kecepatan kendaraan di ruas jalan dari video yang diambil oleh CCTV. Tujuan lainnya adalah untuk mengetahui perbedaan hasil deteksi kecepatan dengan berbagai nilai FPS (*Frame Per Second*).

Input untuk aplikasi ini adalah video (.avi). Pertama, sistem mengambil *Region of Interest* (ROI). Selanjutnya, sistem melakukan *background subtraction*, membuat garis awal dan akhir, memperbarui posisi kendaraan, dan menyimpan hasil kecepatan rata-rata kendaraan ke berkas Excel (.xls).

Skenario uji coba dilakukan berdasarkan nilai FPS pada video (30 FPS, 27 FPS, 25 FPS, dan 20 FPS). Setiap skenario terdapat sub-skenario berdasarkan posisi koordinat garis akhir {(296,0); (282,0); (270,0); dan (248,0)}. Pengujian dilakukan 5 kali setiap skenario, lalu dibandingkan dengan hasil sebenarnya untuk mendapatkan nilai *error* pada sistem. *Error* terkecil yang dihasilkan sistem sebesar 2,75% dengan posisi koordinat garis akhir di (282,0) pada skenario 30 FPS.

Kata Kunci: CCTV, OpenCV, ROI, FPS, Deteksi Kecepatan.

[Halaman ini sengaja dikosongkan]

VEHICLE SPEED DETECTION ON THE ROAD USING OPENCV

Student Name : Andrew
Student ID : 5113 100 032
Major : Informatics Department FTIf-ITS
Advisor 1 : Prof. Ir. Joko Lianto Buliali, M.Sc., Ph.D.
Advisor 2 : Arya Yudhi Wijaya, S.Kom., M.Kom.

ABSTRACT

Nowadays, CCTV camera has been installed on every road segment in many cities. From those cameras, people are able to know the traffic condition, but unable to know the speed of each vehicle. Therefore, the author decided to develop a software that can detect vehicle speed on the road from the video taken by CCTV. Another goal is to identify the difference in speed detection results with various FPS values.

The input for this application is a video with .avi format. First, the system take ROI from the video. For the next step, it performs background subtraction, creates the initial and end line, updates the vehicle's position and stores its average speed to an Excel file (.xls).

The trial scenarios are based on the FPS value of the video (20, 25, 27, and 30 FPS). Each scenario has sub-scenarios based on the position of the end line {(248, 0); (270, 0); (282, 0); and (296, 0)}. Testing is done for each scenario as much as 5 times, then its result is compared with actual result in order to get the system's error value. The smallest error generated by the system is 2.75% with the end line coordinate position at (282, 0) in the 30 FPS scenario.

Keywords: CCTV, OpenCV, ROI, FPS, Speed Detection.

[Halaman ini sengaja dikosongkan]

KATA PENGANTAR

Puji syukur kepada Tuhan Yang Maha Esa atas segala kasih karunia dan rahmat-Nya penulis dapat menyelesaikan Tugas Akhir yang berjudul:

DETEKSI KECEPATAN KENDARAAN BERJALAN DI JALAN MENGGUNAKAN OPENCV

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

1. Bapak, Ibu, kakak dan keluarga besar yang selalu memberikan dukungan penuh untuk menyelesaikan Tugas Akhir ini.
2. Prof. Ir. Joko Lianto Buliali, M.Sc., Ph.D., selaku dosen pembimbing 1 yang telah banyak membantu, membimbing, bahkan memotivasi penulis untuk menyelesaikan Tugas Akhir ini.
3. Bapak Arya Yudhi Wijaya selaku dosen pembimbing 2 yang telah banyak memberikan semangat, motivasi, serta arahan kepada penulis dalam menyelesaikan Tugas Akhir ini.
4. Bapak dan Ibu mahasiswa S-2 dan S-3 sekaligus anak bimbing Prof. Joko Lianto yang telah banyak membimbing dan memberikan arahan kepada penulis dalam menyelesaikan Tugas Akhir ini.
5. David, Jandre, Albert, Alvin, Andre, Freddy, Arianto, Billy, 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, Juli 2017

Andrew

DAFTAR ISI

DAFTAR GAMBAR

[Halaman ini sengaja dikosongkan]

DAFTAR TABEL

[Halaman ini sengaja dikosongkan]

DAFTAR KODE SUMBER

[Halaman ini sengaja dikosongkan]

DAFTAR FORMULA

[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, dan sistematika penulisan Tugas Akhir.

1.1. Latar Belakang

Saat ini, di berbagai kota telah dipasang CCTV pada setiap ruas jalan, terutama pada kota Surabaya. Namun, petugas lalu lintas belum sepenuhnya memanfaatkan kegunaan CCTV. Petugas lalu lintas hanya dapat mengetahui kondisi pada suatu jalan, seperti kepadatan, kecelakaan dan lain-lain. Kecepatan kendaraan sangat berguna untuk diketahui, karena dapat ditentukan apakah kecepatan suatu kendaraan diatas hukum batas kecepatan yang berlaku [1].

Tugas Akhir ini merupakan implementasi dari kebutuhan mengenai deteksi kecepatan kendaraan di jalan dari video. Penelitian yang dilakukan terkait dengan implementasi *library* OpenCV pada aplikasi yang akan dibuat.

Konsep yang digunakan adalah pengambilan *region of interest* (ROI) dari video, dilakukan *background subtraction* untuk mendapatkan latar depan, deteksi kendaraan berjalan, menggambar kotak pada kendaraan yang terdeteksi, dan memperbarui posisi kendaraan yang terdeteksi [2].

1.2. Tujuan

Tujuan dari pembuatan Tugas Akhir ini adalah:

1. Membuat perangkat lunak yang dapat mendeteksi kecepatan kendaraan di ruas jalan menggunakan OpenCV dari video.
2. Mengetahui dampak pengurangan FPS pada video terhadap hasil deteksi kecepatan.

1.3. Rumusan Permasalahan

Rumusan masalah yang diangkat dalam Tugas Akhir ini antara lain:

1. Bagaimana cara implementasi *region of interest* pada video?
2. Bagaimana cara implementasi *background subtraction*?
3. Bagaimana cara mencari bentuk kendaraan yang terdapat pada video?
4. Bagaimana cara memperbarui posisi mobil setiap *frame* pada video?
5. Bagaimana cara mengetahui jarak sebenarnya (dalam meter) yang ditempuh setiap kendaraan?
6. Berapa kecepatan suatu kendaraan yang bisa dideteksi dalam berbagai FPS (*Frame Per Second*)?
7. Apakah besar FPS pada video dapat mempengaruhi hasil dari deteksi kecepatan suatu kendaraan?
8. Berapa besar FPS pada video untuk mendapatkan *error* terkecil yang dihasilkan oleh sistem?
9. Bagaimana peletakan posisi garis akhir analisa untuk mendapatkan *error* terkecil yang dihasilkan oleh sistem?

1.4. Batasan Permasalahan

Permasalahan yang dibahas dalam Tugas Akhir ini memiliki beberapa batasan, antara lain:

1. Video yang digunakan berekstensi .avi, beresolusi 320x176, dan berorientasi horizontal (kendaraan berjalan dari kiri ke kanan)
2. Kendaraan yang dapat dideteksi adalah mobil.
3. Bahasa pemrograman yang digunakan adalah Native Java dengan JavaFX sebagai GUI.
4. *Library* yang digunakan adalah OpenCV, jxl.

1.5. Metodologi

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

a. Penyusunan proposal Tugas Akhir

Proposal Tugas Akhir ini berisi latar belakang pembuatan Tugas Akhir, rumusan masalah, batasan masalah, tujuan pembuatan, manfaat, metodologi hingga jadwal kegiatan pembuatan Tugas Akhir. Selain itu, proposal Tugas Akhir ini memberikan ringkasan dari Tugas Akhir. Proposal Tugas Akhir juga berisi tinjauan pustaka yang digunakan sebagai referensi pembuatan tugas akhir ini.

b. Studi literatur

Pada studi literatur ini, akan dipelajari sejumlah referensi yang diperlukan dalam pembuatan aplikasi yaitu mengenai OpenCV, *frame rate*, *background subtraction*, *region of interest*, pencarian bentuk mobil, dan formula yang digunakan untuk deteksi kecepatan kendaraan pada video.

c. Analisis dan desain perangkat lunak

Tahap ini meliputi perancangan sistem berdasarkan studi literatur dan pembelajaran konsep teknologi dari perangkat lunak yang ada. Tahap ini mendefinisikan alur dari implementasi. Langkah-langkah yang dikerjakan juga didefinisikan pada tahap ini. Pada tahap ini dibuat *prototype* sistem, yang merupakan rancangan dasar dari sistem yang akan dibuat. Serta dilakukan desain suatu sistem dan desain proses-proses yang ada.

d. Implementasi perangkat lunak

Implementasi perangkat lunak ini dibangun dengan bahasa pemrograman Native Java dan *library* yang digunakan adalah OpenCV.

e. Pengujian dan evaluasi

Pada tahap ini dilakukan uji coba terhadap perangkat lunak yang telah dibuat. Pengujian yang dimaksud adalah pengujian *error* yang terdapat pada sistem dengan membandingkan hasil kecepatan uji coba dengan kecepatan sebenarnya. 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 serta hasil dari aplikasi 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. Dasar Teori
3. Analisis dan Perancangan
4. Implementasi
5. Pengujian dan Evaluasi
6. Kesimpulan dan Saran
7. Daftar Pustaka

1.6. Sistematika Penulisan

Buku Tugas Akhir ini bertujuan untuk mendapatkan gambaran dari pengerjaan Tugas Akhir ini. Selain itu, dapat berguna untuk pembaca yang tertarik untuk melakukan

pengembangan lebih lanjut. Secara garis besar, buku Tugas Akhir terdiri atas beberapa bagian seperti berikut.

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 membahas beberapa teori penunjang yang berhubungan dengan pokok pembahasan dan dasar pembuatan Tugas Akhir ini. Teori yang terkait adalah OpenCV, *frame rate*, *background subtraction*, *region of interest*, pencarian bentuk mobil, dan formula untuk deteksi kecepatan kendaraan pada video.

Bab III Analisis dan Perancangan Sistem

Bab ini membahas perancangan perangkat lunak. Perancangan perangkat lunak meliputi jenis data, arsitektur, proses dan perancangan antarmuka aplikasi.

Bab IV Implementasi

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

Bab V Pengujian dan Evaluasi

Membahas tentang 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. Bab ini membahas saran-saran untuk pengembangan sistem lebih lanjut.

Daftar Pustaka

Merupakan daftar referensi yang digunakan untuk mengembangkan Tugas Akhir.

[Halaman ini sengaja dikosongkan]

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 visi komputer. OpenCV menggunakan lisensi BSD, sehingga OpenCV gratis, baik untuk penggunaan akademis maupun komersial. OpenCV dapat digunakan pada bahasa pemrograman C, C++, Python dan Java. OpenCV mendukung Windows, Linux, Android, iOS dan Mac OS. OpenCV dibuat dalam bahasa pemrograman C / C++ dengan OpenCL, dan memiliki lebih dari 2500 algoritma yang telah dioptimalkan. *Library* ini digunakan oleh pengguna di seluruh dunia. OpenCV memiliki lebih dari 47 ribu orang dari komunitas pengguna dan perkiraan jumlah pengunduhan lebih dari 14 juta kali [3], [4].

Open Computing Language (OpenCL) adalah suatu kerangka kerja untuk membuat aplikasi yang mengeksekusi seluruh platform heterogen, seperti *Central Processing Units* (CPU), *Graphics Processing Units* (GPU), *Digital Signal Processors* (DSPs), *Field-Programmable Gate Arrays* (FPGAs) dan prosesor lainnya [5].

2.2. Frame Rate

Frame rate (dinyatakan dalam *Frame Per Second* atau FPS) adalah frekuensi dimana gambar (*frame*) berturut-turut ditampilkan. Istilah ini berlaku sama untuk film, video kamera, komputer grafis, dan sistem *motion capture* [6].

2.3. Background Subtraction

Background subtraction adalah teknik di bidang pengolahan citra untuk pengambilan latar depan gambar. *Background subtraction* merupakan teknik untuk mendeteksi benda bergerak dalam video. Namun, *background subtraction* memiliki kekurangan, yaitu pada video dengan kondisi diluar ruangan yang tidak stabil, seperti hujan, angin, dan perubahan pencahayaan [7], [8]. Algoritma yang akan digunakan berdasarkan penelitian oleh Z.Zivkovic [9], [10] yang berjudul "*Improved adaptive Gausian mixture model for background subtraction*" pada tahun 2004 dan "*Efficient Adaptive Density Estimation per Image Pixel for the Task of Background Subtraction*" pada tahun 2006. Terdapat metode berdasarkan *Gaussian Mixture Model* (GMM) yang tercantum pada penelitian tersebut. *Background subtraction* melakukan analisa pada setiap piksel, apakah suatu piksel merupakan latar belakang atau latar depan yang sesuai dengan definisi pada (2.1). Gambar sebelum dan sesudah penerapan *background subtraction* dapat dilihat pada Gambar 2.1.

$$P(x_t) > C_{thr} \tag{2.1}$$

dimana:

- $P(x_t)$: probabilitas nilai piksel tertentu
- x_t : titik atau piksel pada waktu t
- c_{thr} : nilai threshold

Estimasi model pada *Gaussian mixture model* (GMM) dapat dilihat pada formula (2.2), (2.3), dan (2.4)

$$P(x_t) = \sum_{m=1}^{K} \omega_m . N\left(x_t \mid \mu_m, \sigma_m^2 I\right) \tag{2.2}$$

$$N\left(x_t \mid \mu_m, \sigma_m^2 I\right) = \frac{1}{(2\pi)^{D/2}} \cdot \frac{1}{\left|\sigma_m^2 . I\right|^{1/2}} . e^z \quad (2.3)$$

$$z = -\frac{1}{2}\left(x_t - \mu_m\right)^T . \sigma_m^2 . I^{-1} . \left(x_t - \mu_m\right) \quad (2.4)$$

dimana:

- K : distribusi K Gaussian yang menggambarkan salah satu latar belakang atau latar depan. Biasanya, nilai K diantara 3 dan 5.
- ω_m : estimasi bobot campuran
- μ_m : estimasi means
- σ_m^2 : estimasi variance
- I : matriks identitas
- D : ukuran dimensi
- e : bilangan Euler = 2.718281828… [11]
- T : periode

(a) (b)

Gambar 2.1 (a) sebelum dan (b) sesudah *background subtraction*

2.4. Region of Interest

Region of Interest (ROI) adalah sampel yang diambil dari satu set data untuk tujuan tertentu. Contoh penggunaan ROI

adalah mengetahui batas-batas tumor pada gambar untuk mengukur ukurannya. Dalam sistem informasi geografis (GIS), ROI dapat dipahami sebagai pilihan poligonal dari peta 2D [12]. Untuk melihat perbedaan gambar sebelum dan sesudah pemilihan ROI dapat dilihat pada Gambar 2.2.

(a) (b)

Gambar 2.2 (a) sebelum ROI dan (b) setelah ROI

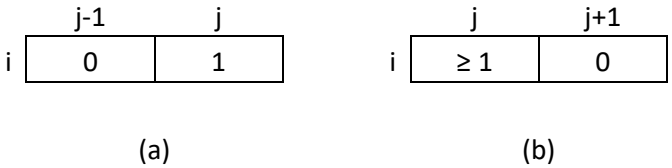
2.5. Pencarian Bentuk Mobil

Algoritma pencarian bentuk mobil didapat dari penelitian oleh S.Suzuki [13] yang berjudul "*Topological Structural Analysis of Digitized Binary Images by Border Following*" pada tahun 1985. Peneliti mengusulkan 2 algoritma untuk mendeteksi objek pada gambar biner (hitam-putih).

Algoritma 1:

- *Input* algoritma adalah suatu gambar biner dan lakukan analisa setiap piksel (i,j). Hentikan analisa jika terdapat suatu piksel (i,j) yang merupakan batas luar (Gambar 2.3a) dan batas lubang (Gambar 2.3b). Jika piksel (i,j) memenuhi kedua kondisi tersebut, maka piksel (i,j) dianggap sebagai titik awal dari batas luar. Tetapkan suatu angka yang unik pada perbatasan yang baru ditemukan, disebut sebagai NBD.
- Tentukan *parent* perbatasan dari perbatasan yang baru ditemukan. Selama proses analisa, simpanlah suatu nomor urut perbatasan, disebut sebagai LNBD, dari perbatasan yang baru ditemukan. LNBD ini merupakan *parent* perbatasan dari

perbatasan sebelumnya (NBD). Oleh karena itu, dapat ditentukan urutan nomor *parent* perbatasan (LNBD) dari perbatasan sebelumnya (NBD) dengan jenis perbatasan menurut Tabel 2.1.



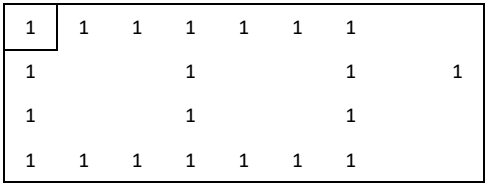
Gambar 2.3 Kondisi perbatasan titik awal (i,j) dari (a) batas luar dan (b) batas lubang

Tabel 2.1 *Decision rule* untuk parent perbatasan dari perbatasan B

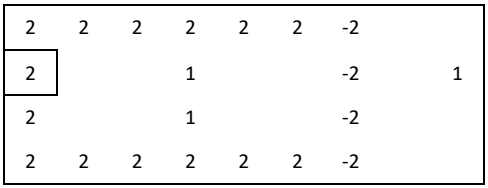
Type of B	**Type of border B' with sequential number LNBD**	
	Outer border	**Hole border**
Outer border	The parent border of the border B'	The border B'
Hole border	The border B'	The parent border of the border B'

- Ikuti perbatasan yang ditemukan dari titik awal dengan menandai piksel. Ketentuan penandaan setiap piksel sebagai berikut
  - Jika perbatasan sekarang adalah diantara 0-komponen yang berisi piksel (p,q+1) dan 1-komponen yang berisi piksel (p,q), maka ubah nilai piksel (p,q) dengan -NBD.
  - Jika tidak, tetapkan nilai piksel (p,q) dan piksel (p,q+1) dengan NBD kecuali piksel berada pada perbatasan yang sudah diikuti.
- Penandaan ini untuk mencegah piksel (p,q) menjadi perbatasan titik awal. Jika tanda suatu piksel bernilai positif maka piksel tersebut merupakan batas terluar gambar. Lakukan algoritma ini hingga pengamatan setiap piksel

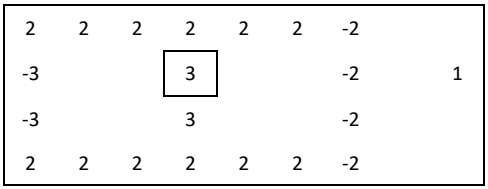
mencapai sudut kanan bawah gambar. Berikut ini merupakan ilustrasi algoritma 1 yang menunjukkan nilai piksel suatu gambar. Piksel di dalam kotak adalah titik awal proses ((a) s/d (e) berurutan):



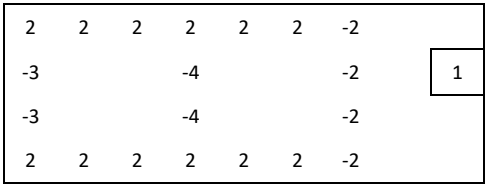
(a)



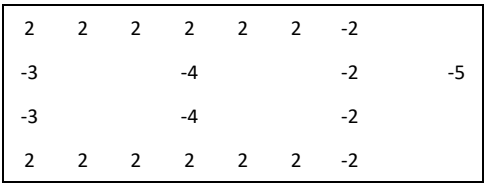
(b)



(c)



(d)



(e)

Algoritma 2 hampir sama dengan algoritma 1, namun terdapat satu perbedaannya yaitu pengamatan hanya pada perbatasaan dengan titik awal batas luar saja atau *outer border* (Gambar 2.3a), sehingga waktu komputasi algoritma 2 lebih cepat daripada algoritma 1.

2.6. Kecepatan Kendaraan pada Video

Formula kecepatan pada umumnya adalah membagi jarak tempuh suatu kendaraan dengan waktu tempuh seperti pada (2.5) dan (2.7). Formula jarak tempuh kendaraan (2.6) didapat dari penelitian oleh Chan Chia Y [14].

$$Speed = \frac{Dist}{Time} \tag{2.5}$$

$$Dist = Df \times \left(\frac{D}{Dx} \right) \times \left(P_n - P_0 \right) \tag{2.6}$$

$$Time = Tf \times \left(t(n) - t(0) \right) \tag{2.7}$$

dimana,

- D*f* : konversi jarak dari meter ke kilometer (0.001)
- D : jarak sesungguhnya (penulis menggunakan perkiraan)
- Dx : jarak antara dua garis area deteksi (awal dan akhir) dalam piksel
- P_n : posisi ujung kanan kendaraan pada t_n

- P_o : posisi ujung kanan kendaraan pada t_0
- T*f* : konversi waktu dari millidetik ke jam (1/1000*60*60)
- t(n) : waktu akhir
- t(0) : waktu awal

BAB III
ANALISIS DAN PERANCANGAN SISTEM

Bab ini terdapat analisis dan perancangan sistem 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 dasar pembuatan aplikasi, yang meliputi tampilan aplikasi, hingga perancangan alur proses yang akan diimplementasikan ke dalam aplikasi.

3.1. Analisis

Tahap analisis meliputi analisis masalah, analisis kebutuhan, deskripsi umum sistem, dan perancangan sistem yang dibuat.

3.1.1. Kontribusi Penelitian

Deteksi kecepatan suatu kendaraan telah diimplementasi pada penelitian sebelumnya, seperti penelitian oleh Hardy Santosa S. [2] dan Chan Chia Y. [14]. Namun, kedua penelitian ini memiliki kekurangan, yaitu akurasi sistem pada penelitian oleh Hardy lebih kecil daripada akurasi pada penelitian oleh Chan, dan algoritma deteksi kendaraan pada penelitian oleh Chan tergolong susah untuk diimplementasikan. Oleh karena itu, penulis ingin mendeteksi kecepatan suatu kendaraan dengan algoritma yang mudah diimplementasi dan menggunakan formula yang digunakan pada penelitian oleh Chan.

3.1.2. Analisis Permasalahan

Saat ini, di berbagai kota telah dipasang CCTV pada setiap lampu lalu lintas. Namun, kegunaan CCTV tersebut hanya untuk melihat situasi lalu lintas saja, seperti kepadatan, dan kecelakaan lalu lintas. Petugas lalu lintas saat ini hanya menebak untuk mengetahui kecepatan suatu kendaraan.

Oleh karena itu, dibuat aplikasi untuk menangani permasalahan tersebut. Ketika petugas lalu lintas ingin mengetahui kecepatan kendaraan pada video yang terekam dari CCTV, petugas lalu lintas cukup membuka aplikasi ini. Setelah itu, petugas lalu lintas memilih ROI yang akan diambil, memasang tiga garis (dua garis awal dan garis akhir), dan menekan tombol "*Start Video*". *Output* aplikasi ini berupa berkas berekstensi .xls (Excel) yang berisi kumpulan hasil deteksi kecepatan mobil dengan *id*, waktu tempuh, posisi awal, dan posisi akhir mobil.

3.1.3. Deskripsi Umum Sistem

Aplikasi yang akan dibuat adalah aplikasi desktop. Gambar 3.1 adalah *flowchart* sistem.

3.1.3.1. Input

Input sistem berupa berkas berekstensi *Audio Video Interleaved* (AVI). Terdapat empat video yang digunakan sebagai *input* untuk dijadikan skenario, yaitu video.avi, video27.avi, video25.avi dan video20.avi. Empat video tersebut diambil dari sumber yang sama, perbedaannya hanya pada FPS setiap video.

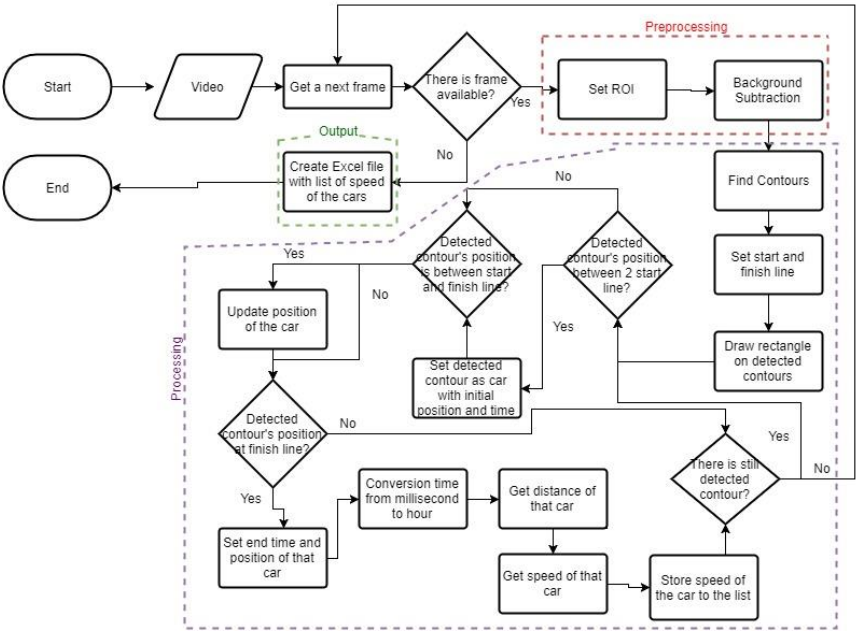
3.1.3.2. Proses

Pada tahap ini, dijelaskan secara bertahap langkah-langkah yang dilakukan sistem, dimulai dari tahap *preprocessing*, *processing*, sampai menjadi hasil *output*.

3.1.3.2.1. Preprocessing

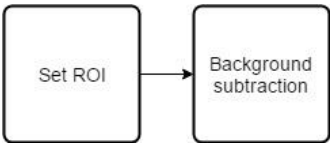
Gambar 3.2 merupakan gambaran umum untuk *preprocessing* yang diambil dari *flowchart* sistem pada Gambar 3.1. Setelah video masuk dalam sistem, sistem akan mengambil ROI yang sudah ditentukan. Tahap pengambilan ROI ini bertujuan untuk lebih fokus menganalisis area, dan menghindari *error* pada sistem. *Error* terjadi pada saat mobil terpotong oleh

batas resolusi video. Ilustrasi pengambilan ROI dapat dilihat pada Gambar 3.3.



Gambar 3.1 *Flowchart* Sistem

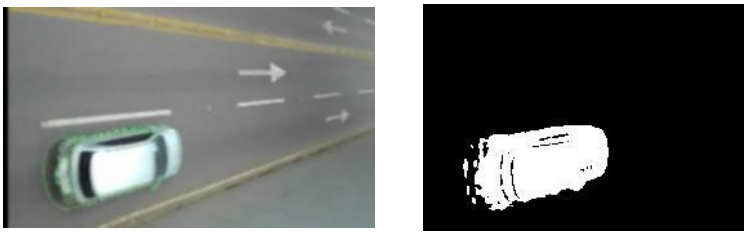
Setelah pengambilan ROI, tahap selanjutnya adalah *background subtraction* untuk mendapatkan benda bergerak pada video. Ilustrasi *background subtraction* dapat dilihat pada Gambar 3.4.



Gambar 3.2 *Preprocessing*

(a) (b)

Gambar 3.3 Ilustrasi (a) sebelum mengambil ROI dan (b) hasil pengambilan ROI

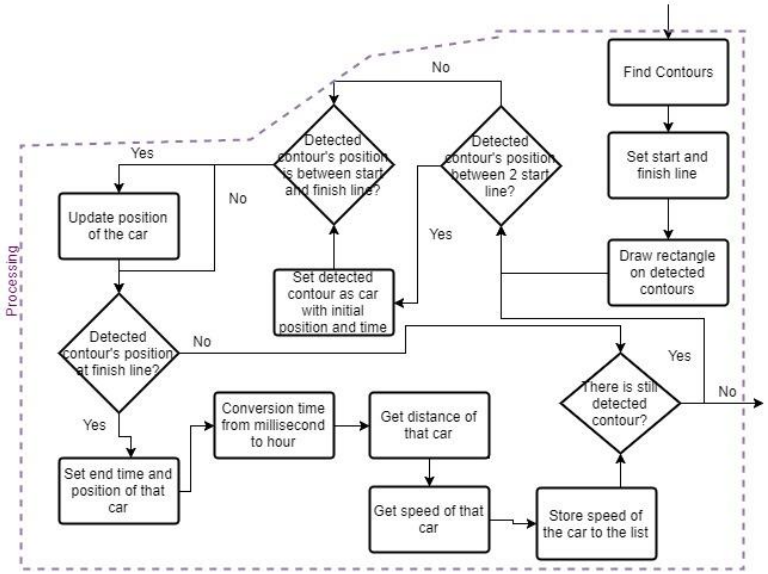


(a) (b)

Gambar 3.4 Ilustrasi (a) sebelum *background subtraction* dan (b) setelah *background subtraction*

3.1.3.2.2. Processing

Gambar 3.5 merupakan *flowchart* untuk *processing* yang diambil dari *flowchart* sistem pada Gambar 3.1. Langkah pertama adalah pencarian bentuk mobil dari gambar hasil *preprocessing* dengan menggunakan algoritma pada penelitian oleh S.Suzuki [13]. Langkah selanjutnya adalah pemasangan 3 garis (dua garis awal dan satu garis akhir), dan pembuatan kotak pada mobil yang terdeteksi. Ilustrasi pembuatan kotak pada mobil yang terdeteksi dapat dilihat pada Gambar 3.6. Sistem akan melakukan pengecekan apakah sisi kanan kotak (ujung mobil) berada diantara 2 garis awal. Jika iya, maka informasi mobil (*id*, posisi awal dan waktu awal mobil) tersebut akan disimpan.

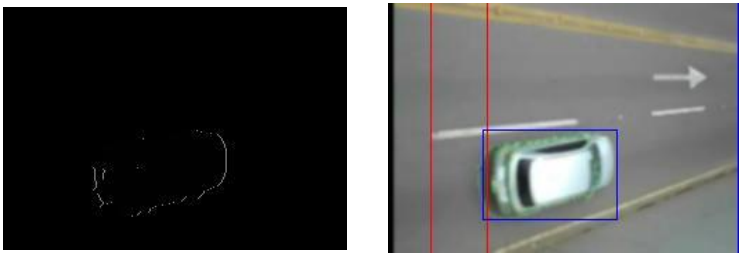


Gambar 3.5 *Processing*

Ilustrasi kondisi ujung mobil berada di antara 2 garis awal dapat dilihat pada Gambar 3.7(a). Posisi mobil akan selalu diperbarui selama posisi mobil tersebut berada diantara garis awal dan garis akhir. Pada saat posisi mobil berada di garis akhir, waktu akhir dan posisi akhir mobil tersebut akan disimpan. Ilustrasi kondisi ujung mobil pada posisi garis akhir dapat dilihat pada Gambar 3.7(b). Sistem akan mengkonversi waktu tempuh mobil dari millidetik ke jam. Jarak tempuh dan kecepatan suatu mobil didapat dengan menggunakan (2.6), dan (2.5).

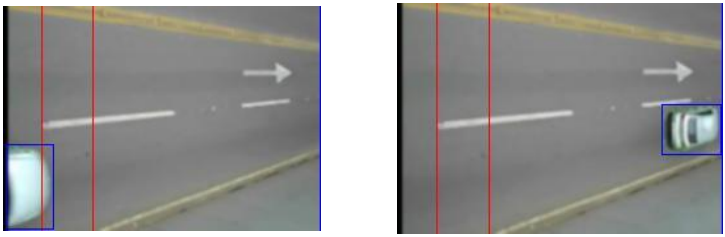
3.1.3.2.2.1. Jarak sebenarnya dalam meter

Pada video, tidak terdapat informasi mengenai panjang jalan sebenarnya. Oleh karena itu, dibutuhkan analisa untuk menentukan jarak sebenarnya yang ditempuh oleh mobil. Penulis memperkirakan panjang jalan sebenarnya dengan menambahkan panjang marka dan panjang pemisah antar marka.



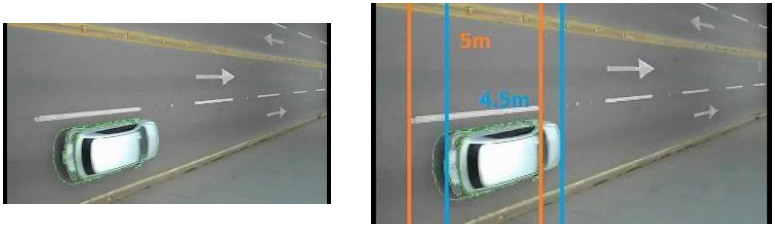
(a) (b)

Gambar 3.6 Ilustrasi (a) penggunaan algoritma pencarian bentuk mobil dan (b) pembuatan kotak pada gambar setelah *background subtraction*



(a) (b)

Gambar 3.7 (a) posisi sisi kanan kotak diantara 2 garis awal, dan (b) Posisi sisi kanan kotak pada garis akhir



(a) (b)

Gambar 3.8 (a) original dan (b) perbandingan panjang mobil dengan panjang marka jalan

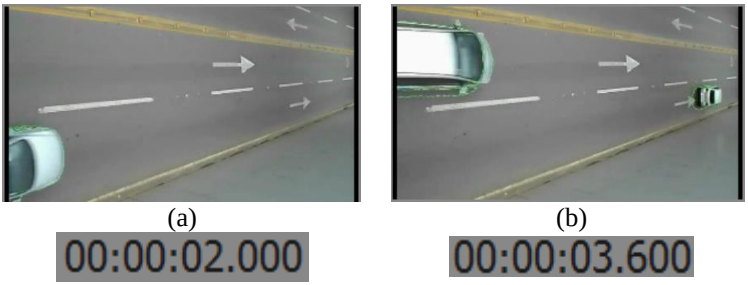
Panjang marka dan panjang pemisah antar marka dibandingkan dengan rata-rata panjang mobil sedan pada umumnya, yaitu sekitar 4,5 meter. Oleh karena itu, perkiraan

panjang marka adalah 5 meter, dan perkiraan panjang pemisah antar marka adalah 4 meter. Untuk ilustrasi perbandingan panjang mobil dengan panjang marka dapat dilihat pada Gambar 3.8. Untuk ilustrasi perbandingan panjang mobil dengan panjang pemisah antar marka dapat dilihat pada Gambar 3.9.

(a) (b)

Gambar 3.9 (a) original dan (b) perbandingan panjang mobil dengan panjang pemisah antar marka jalan

3.1.3.2.2.2. Estimasi kecepatan sebenarnya



(a) (b)

(c) (d)

Gambar 3.10 Kondisi video (a) pada saat waktu 2 detik (c) dan (b) pada saat waktu 3,6 detik (d)

Estimasi kecepatan mobil sebenarnya berguna untuk menghitung *error* pada sistem. Untuk mencari waktu tempuh mobil sebenarnya, dilakukan analisa menggunakan aplikasi video *editing* yang dapat menunjukkan waktu video dalam millidetik.

Setelah jarak sebenarnya dan waktu tempuh sebenarnya diketahui, maka dapat dicari estimasi kecepatan mobil sebenarnya dengan menggunakan (2.5). Untuk ilustrasi waktu pada video dapat dilihat pada Gambar 3.10.

3.1.3.3. Output

Tahap terakhir adalah pembuatan berkas Excel dengan kumpulan kecepatan mobil yang dihasilkan pada tahap *processing*. Berkas Excel akan dibuat pada saat pengguna menekan tombol "*Stop Video*" atau durasi video telah habis.

3.2. Perancangan Sistem

Tahap ini meliputi perancangan tampilan aplikasi, dan perancangan alur proses penggunaan aplikasi. Aplikasi ini dibangun dan dioperasikan pada lingkungan tertentu.

Lingkungan pengembangan aplikasi ini adalah sebagai berikut.

Perangkat keras

- Komputer : Prosesor Intel® Core™ i7-CPU (2.50GHz), RAM 12 GB, Graphic Intel ® IvyBridge Laptop

Perangkat lunak

- Sistem operasi : Windows 10
- IDE : Eclipse SDK Neon I20160606-1100
- *Library* : OpenCV 3.1.0, JavaFX, jxl

3.2.1. Jenis Video Input

Pada subbab ini, dijelaskan mengenai jenis video yang digunakan. Seperti pada Tabel 3.1, beberapa video direkam pada lokasi yang sama dengan FPS yang berbeda-beda. Contoh gambar pada video ini dapat dilihat pada Gambar 3.11.

Tabel 3.1 Daftar video yang menjadi *input* sistem

Nama Video	Resolusi Video	FPS
video.avi	320x176	30
video27.avi	320x176	27
video25.avi	320x176	25
video20.avi	320x176	20

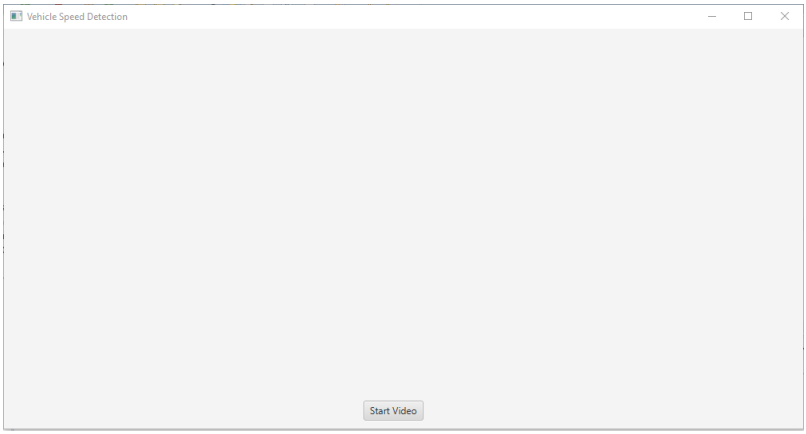
Gambar 3.11 *Input* video

3.2.2. Perancangan Antar Muka

Pada subbab ini, dijelaskan bagaimana rancangan antarmuka yang akan berinteraksi secara langsung dengan pengguna.

3.2.2.1. Perancangan Tampilan Awal Aplikasi Dibuka

Pada saat pengguna menjalankan aplikasi dari IDE Eclipse, terdapat tampilan dengan judul "*Vehicle Speed Detection*" dan tombol "*Start Video*". Tombol ini berguna untuk memulai proses analisa. Perancangan tampilan dapat dilihat pada Gambar 3.12.



Gambar 3.12 Tampilan awal aplikasi

3.2.2.2. Perancangan Tampilan Saat Proses Analisa Berjalan

Gambar 3.13 Tampilan pada saat proses analisa berjalan

Tampilan aplikasi pada saat proses analisa berjalan hampir sama dengan tampilan awal aplikasi. Dapat dilihat pada Gambar 3.13, terdapat dua *ImageView* yang digunakan untuk menampilkan video. Video dengan garis awal (berwarna merah) dan garis akhir (berwarna biru) ditampilkan pada *ImageView* sebelah kiri. Pengambilan ROI dari video dengan garis awal dan garis akhir ditampilkan pada *ImageView* sebelah kanan. Tombol "*Stop Video*" berguna untuk memberhentikan proses analisa.

3.2.2.3. Perancangan Tampilan Hasil

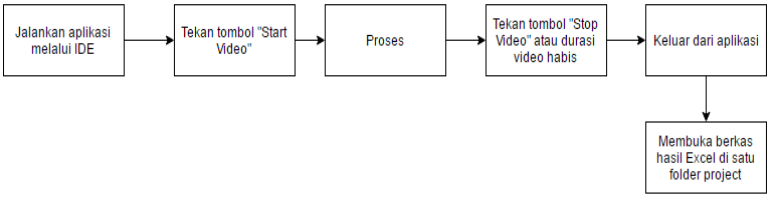
Hasil dari aplikasi ini adalah berkas Excel yang berisi kumpulan hasil deteksi kecepatan mobil. Hasil berkas tersebut dibuat pada saat pengguna menekan tombol "*Stop Video*" atau durasi video telah habis. Pada Gambar 3.14 terlihat tampilan hasil berkas Excel. Pada berkas tersebut terdapat label *CarID* (*id* mobil), *Speed* [km/h] (kecepatan rata-rata mobil), *StartTime* (waktu awal), *EndTime* (waktu akhir mobil), *StartPos* (posisi mobil awal), dan *FinishPos* (posisi mobil akhir).

CarID.	Speed [km/h]	StartTime	FinishTime	StartPos	FinishPos
1	76.76532745	25/4/2017 2:37:24:586	25/4/2017 2:37:25:812	38	282
2	77.20884705	25/4/2017 2:37:25:992	25/4/2017 2:37:27:161	48	282
3	81.62876129	25/4/2017 2:37:26:478	25/4/2017 2:37:27:527	60	282
4	84.01328278	25/4/2017 2:37:28:806	25/4/2017 2:37:29:839	57	282
5	80.32698822	25/4/2017 2:37:31:647	25/4/2017 2:37:32:713	60	282

Gambar 3.14 Tampilan hasil melalui berkas Excel

3.2.3. Perancangan Alur Proses Penggunaan Aplikasi

Gambar 3.15 adalah penjelasan alur penggunaan aplikasi dalam bentuk *workflow*. Alur proses ini memperlihatkan langkah demi langkah yang akan dilakukan pengguna untuk menggunakan aplikasi.



Gambar 3.15 *Workflow* penggunaan aplikasi

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 sewaktu-waktu apabila dibutuhkan.

4.1. Lingkungan Implementasi

Lingkungan yang digunakan untuk implementasi sama seperti yang dituliskan pada rancangan, yaitu menggunakan beberapa perangkat pendukung.

4.1.1. Lingkungan Implementasi Perangkat Keras

Perangkat yang digunakan untuk pengembangan aplikasi ini adalah laptop. Spesifikasi laptop yang digunakan adalah prosesor Intel Core i7 3537U dengan Intel HD Graphics 4000 dan RAM 12 GB.

4.1.2. Lingkungan Implementasi Perangkat Lunak

Perangkat lunak yang digunakan untuk pengembangan aplikasi ini adalah sebagai berikut:

- Microsoft Windows 10 sebagai sistem operasi pada laptop.
- Eclipse SDK Neon I20160606-1100 untuk IDE perancangan sistem.
- OpenCV 3.1.0, JavaFX, jxl untuk *library* pendukung dalam perancangan sistem.

4.2. Implementasi Kode

Subbab ini membahas tentang implementasi analisis sistem yang telah dirancang dan dibahas pada Bab III.

4.2.1. Implementasi Input dan Penampilan Video

Implementasi ini diambil dari *flowchart* Gambar 3.1, yaitu pada langkah video yang digunakan sebagai *input*, dan pada

langkah *get a next frame* (gambar ditampilkan pada tampilan aplikasi yang telah dibuat).

Kode 4.1 merupakan implementasi *input* video dan penampilan video yang sudah dianalisis. Analisis video (*preprocessing* dan *processing*) dilakukan pada fungsi *grabFrame()* dan menghasilkan gambar dengan tipe data *Mat*. Hasil tersebut akan dikonversi menjadi tipe data *Image*, lalu ditampilkan pada *ImageView* yang ada di tampilan aplikasi.

```
private VideoCapture capture = new VideoCapture();
this.capture.open("C:/Users/andre.DESKTOP-
OPOUQEH/EclipseNeon/workspace/VehicleTS with
JavaFX/src/resources/road_traffic.avi");
if (this.capture.isOpened())
{
        Runnable frameGrabber = new Runnable() {
        @Override
        public void run()
        {
                // effectively grab and process a single frame
                Mat editedFrame = grabFrame();
                // convert and show the frame
                if(!frame.empty()){
                        Image imageEditedToShow =
Utils.mat2Image(editedFrame, (long) videoFPS);
                        updateImageView(editFrame,
imageEditedToShow);
                }
        }
        };
```

Kode 4.1 Inisialisasi untuk *input* video dan *output* video

4.2.2. Implementasi Preprocessing Video

Setelah video diambil sebagai *input* sistem, tahap selanjutnya adalah *preprocessing* seperti yang tercantum pada *flowchart* Gambar 3.1, dimana tahap *preprocessing* ini berupa pengambilan ROI (langkah *set ROI*) dan eliminasi latar belakang (langkah *background subtraction*).

Dapat dilihat pada Kode 4.2, pengambilan ROI dimulai pada fungsi *Rect* dengan parameter koordinat *x* awal, koordinat *y* awal, lebar dan tinggi. Setelah itu, sistem melakukan *mapping* pada *frame* dengan variabel *rectRoi* yang sudah ditentukan dan disimpan di variabel *roi*.

Setelah pengambilan ROI, sistem akan melakukan *background subtraction*. Sistem akan melakukan inisialisasi fungsi *background subtraction* dengan parameter panjang *history*, ukuran *threshold*, dan penggunaan deteksi bayangan. Setelah itu, *background subtraction* diterapkan pada gambar dengan *learning rate* sebesar 0.001. Hasil *background subtraction* akan disimpan di variabel *foreground*.

```
//set ROI
Rect rectRoi = new Rect(0,0,297,176);
roi = new Mat(frameClone, rectRoi);

//background subtraction
private BackgroundSubtractorMOG2 mog;
private double learningRate = 0.001;
private double imageThreshold = 20;
private int history = 1500;
mog =
org.opencv.video.Video.createBackgroundSubtractorMOG2(history,
imageThreshold, false);
mog.apply(frameClone, foreground, learningRate);
```

Kode 4.2 Implementasi *preprocessing*

4.2.3. Implementasi Processing Video

Tahap *processing* video terdiri dari pencarian bentuk mobil, inisialisasi objek mobil, pembaruan posisi mobil, dan penghitungan kecepatan mobil.

4.2.3.1. Implementasi Pencarian Bentuk Mobil

Pencarian bentuk mobil merupakan langkah awal pada tahap *processing*. Langkah ini merupakan implementasi langkah *Find Contours* pada *flowchart* Gambar 3.1. Setelah pencarian bentuk mobil, sistem akan membuat 3 garis (dua garis awal dan

satu garis akhir), dan membuat kotak pada objek yang terdeteksi. Pembuatan garis dan kotak pada objek ini merupakan implementasi untuk langkah *set start and finish line* dan *draw rectangle on detected contours* yang terdapat pada *flowchart*.

Sistem menggunakan algoritma dari penelitian oleh S.Suzuki [13] untuk pencarian objek mobil. Dapat dilihat pada Kode 4.3, algoritma ini terdapat pada fungsi *findContours* dimana parameternya adalah gambar hitam putih (hasil *background subtraction*), variabel penampung objek-objek yang terdeteksi, penampung vektor, *mode* yang akan digunakan, dan *method* yang akan digunakan. Saat ini *mode* sistem yang digunakan adalah *CHAIN_APPROX_NONE*, dimana sistem akan menyimpan semua titik pada objek. *Method* sistem yang digunakan adalah *CHAIN_APPROX_SIMPLE*, dimana sistem akan melakukan kompresi segmen horizontal, vertikal, dan diagonal menjadi beberapa titik (contoh: bentuk persegi panjang menjadi 4 titik) [15]. Sistem akan membuat garis awal dan akhir pada gambar menggunakan fungsi *line* dengan parameter *input* gambar, titik pertama, titik kedua, warna garis (RGB), dan ketebalan garis.

Sistem akan membandingkan luas objek yang terdeteksi dengan luas yang diinginkan. Jika objek yang terdeteksi memiliki luas lebih besar dari yang diinginkan, maka objek yang terdeteksi tersebut akan disimpan pada daftar *goodContours* (objek diidentifikasi sebagai mobil) dan membuat kotak pada objek-objek tersebut.

4.2.3.2. Implementasi Inisialisasi Objek Mobil

Implementasi ini dilakukan pada saat objek yang terdeteksi berada di antara 2 garis awal. Seperti *flowchart* pada Gambar 3.1, setelah langkah penggambaran kotak pada mobil, sistem akan melakukan pengecekan apakah mobil yang terdeteksi berada di antara 2 garis awal. Langkah ini merupakan langkah dari *conditional state* pada *flowchart*, yaitu *detected contour's position is between 2 start line*. Jika kondisinya *yes*, maka sistem akan melakukan inisialisasi objek mobil. Langkah ini merupakan

langkah *set detected contour as car with initial position and time* dari *flowchart* sistem.

```
//find contours
List<MatOfPoint> contours = new ArrayList<MatOfPoint>();
Imgproc.findContours(binary, contours, new Mat(),
Imgproc.CHAIN_APPROX_NONE, Imgproc.CHAIN_APPROX_SIMPLE);

//draw start and finish line
Imgproc.line(image, lineCount1, lineCount2, new Scalar(0, 0,
255), 1);
Imgproc.line(image, lineCount3, lineCount4, new Scalar(0, 0,
255), 1);
Imgproc.line(image, lineSpeed1, lineSpeed2, new Scalar(255, 0,
0), 1);

//draw rectangle on detected contours
public List<MatOfPoint> goodContours = new
ArrayList<MatOfPoint>();
private int areaThreshold = 400;
for (int i = 0; i < contours.size(); i++) {
    MatOfPoint currentContour = contours.get(i);
    double currentArea = Imgproc.contourArea(currentContour);
    if (currentArea > areaThreshold) {
       goodContours.add(contours.get(i));
       Rect rectangle = Imgproc.boundingRect(currentContour);
       Imgproc.rectangle(image, rectangle.tl(),
rectangle.br(), new Scalar(255, 0, 0), 1);
    }
}
```

Kode 4.3 Implementasi pencarian bentuk mobil, membuat garis awal dan akhir, dan membuat kotak pada objek yang terdeteksi

Sistem akan membuat objek mobil dengan atribut *id* mobil, waktu awal mobil, waktu akhir mobil, posisi awal mobil, penampung untuk pembaruan posisi mobil dan posisi akhir mobil. Dapat dilihat pada Kode 4.4, pada fungsi *isVehicleToAdd()* terdapat pengecekan apakah objek mobil berada di antara 2 garis awal. Proses pengecekan ini dapat dilihat pada fungsi *rectContainLineStart()*. Sistem mengecek apakah posisi mobil yang terdeteksi berada di antara 2 garis awal. Jika suatu mobil

yang terdeteksi di antara 2 garis tersebut, maka sistem melakukan inisialisasi mobil dengan mengisi posisi awal mobil, waktu awal mobil, dan *id* mobil. Mobil yang terdeteksi ini akan disimpan ke suatu *list*.

```
//pada class utama
if (countVehicles.isVehicleToAdd()) {
   counter++;
   Car car = countVehicles.getCar();
   car.setID(counter);
   cars.add(car);
   //...
}

//pada class countVehicles dan fungsi isVehicleToAdd()
for (int i = 0; i < goodContours.size(); i++) {
    this.rectangle =
Imgproc.boundingRect(goodContours.get(i));
      if (checkRectLine.rectContainLineStart(rectangle)) {
        contourVehicle = getGoodContours().get(i);
        countingFlag = true;
        break;
    }
}
if(countingFlag == true) {
   if (crossingLine == false) {
        crossingLine = true;
        car = new Car();
        car.setRect(rectangle);
        car.setStartRect(rectangle);
        car.setStarttime(System.currentTimeMillis());
        return true;
   //...
}

//pada class checkRectLine dan fungsi rectContainLineStart()
if(rect.br().x >= l1.x && rect.br().x <= l3.x) return true;
else return false;
```

Kode 4.4 Implementasi inisialisasi objek mobil

4.2.3.3. Implementasi Pembaruan Posisi Mobil

Langkah selanjutnya adalah pembaruan posisi mobil. Sistem akan melakukan pembaruan posisi mobil selama mobil yang baru terdeteksi berada di antara garis awal dan garis akhir.

Oleh karena itu, sistem akan mengecek apakah mobil yang baru terdeteksi berada di antara garis awal dan garis akhir. Pengecekan ini merupakan langkah dari *conditional statement* pada *flowchart* Gambar 3.1, yaitu *detected contour's position is between start and finish line*. Jika kondisinya *yes*, maka sistem akan melakukan pembaruan posisi mobil dengan posisi mobil yang baru terdeteksi. Pembaruan posisi mobil merupakan langkah dari pernyataan *update position of the car* pada *flowchart.*

```
if(!cars.isEmpty()) {
  for(int x=0;x<cars.size();x++) {
      //get goodContours
      ArrayList<MatOfPoint> listFoundContours =
(ArrayList<MatOfPoint>) countVehicles.getGoodContours();
      ArrayList<Rect> listFoundRect = new ArrayList<>();
      for(int a=0;a<listFoundContours.size();a++) {
          Rect rectangle = new Rect();
          rectangle =
Imgproc.boundingRect(listFoundContours.get(a));
           listFoundRect.add(rectangle);
      }
      //check goodContours with car in list of cars
      if(!listFoundRect.isEmpty()){
         for(int y=0;y<listFoundRect.size();y++) {
            Point centerFoundRect = new
Point((listFoundRect.get(y).x + listFoundRect.get(y).width)/2,
(listFoundRect.get(y).y + listFoundRect.get(y).height)/2);
            Point centerCar = new
Point((cars.get(x).getRect().x +
cars.get(x).getRect().width)/2, (cars.get(x).getRect().y +
cars.get(x).getRect().height)/2);
            if(((centerCar.y-centerFoundRect.y) < 15) &&
((centerFoundRect.x-centerCar.x) > 0) && ((centerFoundRect.x-
centerCar.x) < 20)){
               cars.get(x).setRect(listFoundRect.get(y));
               //...
            }
         }
      }
  }
}
```

Kode 4.5 Implementasi pembaruan posisi mobil

Dapat dilihat pada Kode 4.5, sistem mencari beberapa mobil yang baru terdeteksi pada gambar (fungsi *getGoodContours*, fungsi mengambil *goodContours* yang terdapat pada langkah pencarian bentuk mobil). Kumpulan mobil yang baru terdeteksi ini akan dilakukan pengecekan terhadap mobil yang sudah disimpan sebelumnya. Pengecekan dilakukan dengan ketentuan, apakah selisih titik tengah koordinat *y* mobil yang disimpan dengan titik tengah koordinat *y* mobil yang baru terdeteksi kurang dari 15, apakah selisih titik tengah koordinat *x* mobil yang baru terdeteksi dengan titik tengah koordinat *x* mobil yang disimpan lebih dari 0, dan apakah selisih titik tengah koordinat *x* mobil yang baru terdeteksi dengan titik tengah koordinat *x* mobil yang disimpan kurang dari 20. Jika ketiga hal tersebut terpenuhi, maka sistem akan memperbarui mobil yang terdapat pada *list*. Pengecekan ini dilakukan agar tidak terjadi kesalahan pembaruan setiap mobil. Kesalahan ini sering terjadi jika terdapat 2 mobil yang berada di antara garis awal dan garis akhir, atau posisi 2 mobil yang saling sejajar satu sama lain.

```
//perubahan waktu dari milliseconds ke jam
float seconds = (longtraveltime / 1000.0f);
float minutes = seconds / 60.0f;
float hour = minutes / 60.0f;

//penerapan formula jarak
double distance = 0.001*(32/(end.x-
start.x))*(car.getEndRect().br().x-car.getStartRect().br().x);

//penerapan formula kecepatan
float speed = distance / hour;
```

Kode 4.6 Implementasi perubahan waktu dan formula jarak serta kecepatan

4.2.3.4. Implementasi Penghitungan Kecepatan

Sebelum melakukan penghitungan kecepatan, sistem akan mengecek apakah suatu mobil berada pada garis akhir. Langkah ini merupakan langkah dari *conditional statement* pada *flowchart* Gambar 3.1, yaitu *detected contour's position at finish line*. Jika

kondisinya *yes*, maka sistem akan mengisi posisi akhir, dan waktu akhir mobil yang bersangkutan. Setelah itu, dilakukan penghitungan jarak tempuh dan kecepatan mobil menggunakan (2.6), dan (2.5). Langkah ini merupakan langkah dari pernyataan pada *flowchart* sistem, yaitu *set end time and position of that car*, *get distance of that car*, dan *get speed of that car*.

```
//pada class utama
if(!cars.isEmpty()) {
   for(int x=0;x<cars.size();x++){
       if (countVehicles.isVehicleSpeed(cars.get(x))) {
          cars.get(x).setEndtime(System.currentTimeMillis());
          cars.get(x).setEndRect(cars.get(x).getRect());
          System.out.println("CarID: "+ cars.get(x).getID() +
" finish!");
          long longtraveltime = cars.get(x).getEndtime() -
cars.get(x).getStarttime();
          float hour = convertLongToHour(longtraveltime);
          System.out.println("CarID: "+ cars.get(x).getID() +
" with speed: " + calculatespeed(hour, cars.get(x)) + "
km/h");
           //...
       }
    }
}
```

Kode 4.7 Implementasi pada class utama pengecekan mobil berada pada garis akhir

Dapat dilihat pada Kode 4.7, dan Kode 4.8, sistem mengecek mobil dari daftar mobil. Apakah posisi mobil lebih dari atau sama dengan garis akhir. Setelah itu, sistem akan melakukan pengurangan antara waktu akhir dan waktu awal seperti pada (2.7). Lalu mengubah selisih waktu tersebut dari *milliseconds* ke jam. Langkah ini merupakan implementasi dari pernyataan pada *flowchart*, yaitu *conversion time from millisecond to hour*. Konversi waktu, dan implementasi formula jarak dan kecepatan dapat dilihat pada Kode 4.6. Formula jarak dan kecepatan yang digunakan berdasarkan (2.6) dan (2.5).

```
//pada class countVehicles dan fungsi isVehicleSpeed()
if (checkSpeedLine.rectContainLineFinish(car.getRect())) {
    speedFlag = true;
}
if(speedFlag == true) {
   if (car.isCrossingSpeedLine()== false) {
       car.setCrossingSpeedLine(true);
       return true;
   } else {
       return false;
   }
}
else {
   car.setCrossingSpeedLine(false);
   return false;
}

//pada class checkRectLine dan fungsi rectContainLineFinish()
if(rect.br().x >= l1.x) return true;
else return false;
```

Kode 4.8 Implementasi pada fungsi pengecekan mobil berada pada garis akhir

4.2.4. Implementasi Output

Setelah sistem mengumpulkan informasi mobil dengan hasil kecepatannya, sistem akan mengecek apakah masih terdapat mobil yang terdeteksi pada gambar. Langkah ini merupakan langkah dari *conditional statement* pada *flowchart* Gambar 3.1, yaitu *there is still detected contour*. Jika kondisinya *no*, maka sistem akan mengambil *frame* selanjutnya pada video. Jika tidak ada *frame* yang akan diambil, maka sistem akan membuat berkas Excel yang berisi kumpulan informasi mobil dengan kecepatannya. Langkah ini merupakan implementasi *conditional statement* dan pernyataan dari *flowchart* sistem, yaitu *there is frame available* dan *create Excel file with list of speed of the cars*. Dapat dilihat pada Kode 4.9, sistem melakukan inisialisasi berkas Excel dan menambahkan label pada berkas. Setelah itu, sistem mengisi data dari kumpulan mobil dengan kecepatan dan waktu tempuh. Untuk menambahkan data ke Excel dapat dilihat pada

Kode 4.10. Dapat dilihat juga pada Kode 4.10, sistem melakukan pembuatan berkas Excel setelah video berakhir.

```
private File fileToSaveXLS;
private WritableSheet sheet;
private WritableSheet sheet2;
private WritableWorkbook workbook;

//inisialisasi file Excel
String xlsSavePath = "Results.xls";
fileToSaveXLS = new File(xlsSavePath);
try {
     workbook = Workbook.createWorkbook(file);
} catch (IOException e) {
     e.printStackTrace();
}
sheet = workbook.createSheet("Results", 0);
addLabel(sheet, 0, 0, "CarID.");
addLabel(sheet, 1, 0, "Speed [km/h]");
addLabel(sheet, 2, 0, "StartTime");
//...

//pengisian data Excel dari implementasi penghitungan kec.
if (countVehicles.isVehicleSpeed(cars.get(x))) {
    //...
    addNumberInteger(sheet, 0, cars.get(x).getID(),
cars.get(x).getID());
    addNumberDouble(sheet, 1, cars.get(x).getID(), (double)
calculatespeed(hour, cars.get(x)));
    DateInt date = new DateInt();
    date = convertLongToDate(cars.get(x), date);
    String labelStartDate = date.getDaystart() + "/" +
date.getMonthstart() + "/" + date.getYearstart() + " " +
date.getHourstart() + ":" + date.getMinutestart() + ":" +
date.getSecondstart() + ":" + date.getMillistart();
    addLabel(sheet, 2, cars.get(x).getID(), labelStartDate);
    addLabel(sheet, 3, cars.get(x).getID(), labelFinishDate);
    addNumberDouble(sheet, 4, cars.get(x).getID(),
cars.get(x).getStartRect().br().x);
    addNumberDouble(sheet, 5, cars.get(x).getID(),
cars.get(x).getEndRect().br().x);
    //...
}
```

Kode 4.9 Implementasi inisialisasi berkas Excel dan pengisian data Excel

```
//pembuatan berkas Excel
if (!frame.empty())
{
   //...
}
else{
   try {
     workbook.write();
     workbook.close();
   } catch (IOException | WriteException e) {
     e.printStackTrace();
   }
   //...
}

//fungsi untuk menambahkan label, integer, dan double
private void addLabel(WritableSheet sheet, int column, int
row, String text) {
   label = new Label(column, row, text);
   try {
     sheet.addCell(label);
   } catch (WriteException e) {
     e.printStackTrace();
   }
}
private void addNumberInteger(WritableSheet sheet, int column,
int row, Integer integer) throws WriteException {
     number = new Number(column, row, integer);
     sheet.addCell(number);
}
private void addNumberDouble(WritableSheet sheet, int column,
int row, Double d) throws WriteException {
     number = new Number(column, row, d);
     sheet.addCell(number);
}
```

Kode 4.10 Implementasi pembuatan berkas Excel dan fungsi menambah data

BAB V
PENGUJIAN DAN EVALUASI

Bab ini membahas pengujian dan evaluasi pada aplikasi yang dikembangkan. Pengujian yang dilakukan adalah pengujian seberapa besar *error* yang dihasilkan oleh sistem dan terhadap tujuan dibuatnya aplikasi ini, yaitu mengetahui kecepatan suatu kendaraan pada video.

5.1. Lingkungan Pengujian

Lingkungan pengujian sistem dilakukan pada perangkat sebagai berikut:

Prosesor	: Prosesor Intel® Core™ i7-CPU
RAM	: 12 GB
Jenis Perangkat	: Laptop
Sistem Operasi	: Windows 10

5.2. Skenario Pengujian

Terdapat 4 skenario dengan perubahan nilai FPS dan 4 sub-skenario dengan perubahan garis akhir. Pemilihan skenario dan sub-skenario ini bertujuan untuk melihat dampak pada nilai *error* dari sistem. Penjelasan 4 skenario dan 4 sub-skenario dapat dilihat pada Tabel 5.1. Pada penghitungan *error* sistem, dilakukan uji coba menggunakan laptop sesuai yang dijelaskan pada lingkungan pengujian. Untuk ilustrasi penggunaan sub-skenario perubahan garis akhir dapat dilihat pada Gambar 5.1. Pada gambar tersebut terdapat 2 garis merah yang terletak di sebelah kiri dan 1 garis biru yang terletak di sebelah kanan. Dua garis merah tersebut adalah 2 garis awal, dan garis biru merupakan garis akhir. Pengujian pada kecepatan mobil yang terdeteksi ini dimulai pada saat pengguna menekan tombol "*Start Video*" pada aplikasi.

Tabel 5.1 Skenario Pengujian

Skenario		FPS	Koordinat Garis Akhir	Jarak Sebenarnya (m)
1	a	30	(296,0) dan (296,240)	32
	b		(282,0) dan (282,240)	27
	c		(270,0) dan (270,240)	23
	d		(248,0) dan (248,240)	18
2	a	27	(296,0) dan (296,240)	32
	b		(282,0) dan (282,240)	27
	c		(270,0) dan (270,240)	23
	d		(248,0) dan (248,240)	18
3	a	25	(296,0) dan (296,240)	32
	b		(282,0) dan (282,240)	27
	c		(270,0) dan (270,240)	23
	d		(248,0) dan (248,240)	18
4	a	20	(296,0) dan (296,240)	32
	b		(282,0) dan (282,240)	27
	c		(270,0) dan (270,240)	23
	d		(248,0) dan (248,240)	18

5.3. Akurasi Pengujian Fungsionalitas

Pada subbab ini akan diberikan hasil evaluasi dari pengujian yang telah dilakukan. Evaluasi yang diberikan meliputi evaluasi pengujian *error* yang dihasilkan oleh sistem. Pada penghitungan *error* sistem, penulis akan membandingkan hasil skenario dengan hasil sebenarnya (kecepatan mobil sebenarnya). Untuk kecepatan mobil sebenarnya dapat dilihat pada Tabel 5.2. *Error* sistem dihitung menggunakan (5.1) dimana KS adalah kecepatan sebenarnya dan KU adalah kecepatan uji coba.

$$Error = \left(\frac{KS - KU}{KS} \right) \times 100\% \qquad (5.1)$$

Tabel 5.2 Hasil Kenyataan

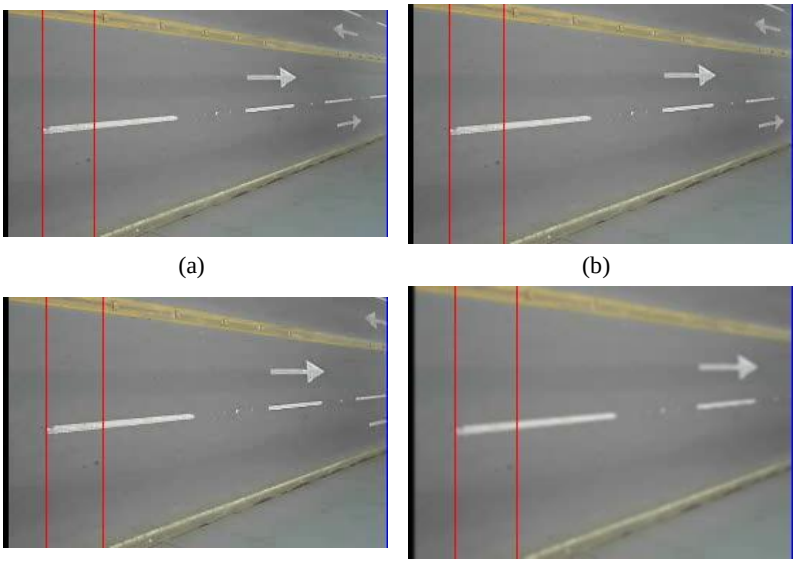
Skenario	Car ID	Start Time (s)	Finish Time (s)	delta time (h)	Real Distance (km)	Kecepatan (km/h)
a	1	1,970	3,570	0,0004444444	0,032	72,00
	2	3,500	5,040	0,0004277778	0,032	74,81
	3	4,000	5,460	0,0004055556	0,032	78,90
	4	6,540	7,870	0,0003694444	0,032	86,62
	5	9,670	11,100	0,0003972222	0,032	80,56
b	1	1,970	3,300	0,0003694444	0,027	73,08
	2	3,500	4,800	0,0003611111	0,027	74,77
	3	4,000	5,200	0,0003333333	0,027	81,00
	4	6,540	7,700	0,0003222222	0,027	83,79
	5	9,670	10,870	0,0003333333	0,027	81,00
c	1	1,970	3,150	0,0003277778	0,023	70,17
	2	3,500	4,633	0,0003147222	0,023	73,08
	3	4,000	5,067	0,0002963889	0,023	77,60
	4	6,540	7,550	0,0002805556	0,023	81,98
	5	9,670	10,740	0,0002972222	0,023	77,38
d	1	1,970	2,870	0,0002500000	0,018	72,00
	2	3,500	4,380	0,0002444444	0,018	73,64
	3	4,000	4,850	0,0002361111	0,018	76,24
	4	6,540	7,340	0,0002222222	0,018	81,00
	5	9,670	10,500	0,0002305556	0,018	78,07

5.3.1. Skenario 1

A. Skenario 1a

Pada skenario ini, penulis akan membandingkan hasil dari sistem pada skenario 1a dengan skenario A. Dapat dilihat pada Tabel 5.3 bahwa setiap percobaan memiliki hasil yang berbeda.

Kumpulan hasil kecepatan setiap mobil akan dirata-rata untuk mendapatkan *error* pada sistem.



(a) (b)

(c) (d)

Gambar 5.1 skenario letak garis akhir (a) sub-skenario a, (b) sub-skenario b, (c) sub-skenario c, (d) sub-skenario d.

Tabel 5.3 *Error* sistem pada skenario 1a

Estimasi Kecepatan							
CarID	Percobaan					Rata-rata	Error (%)
	1	2	3	4	5		
1	79,36	78,69	78,36	80,27	79,30	79,19	9,99
2	77,38	76,28	75,74	77,33	76,77	76,70	2,53
3	84,33	83,10	82,96	84,40	84,05	83,77	6,17
4	86,47	86,40	84,98	86,40	86,33	86,12	0,58
5	81,77	82,03	80,86	82,56	82,36	81,91	1,68
Rata-rata							4,19

B. Skenario 1b

Pada skenario ini, penulis akan membandingkan hasil dari sistem pada skenario 1b dengan skenario B. Dapat dilihat pada Tabel 5.4 bahwa setiap percobaan memiliki hasil yang berbeda. Kumpulan hasil kecepatan setiap mobil akan dirata-rata untuk mendapatkan *error* pada sistem.

Tabel 5.4 *Error* sistem pada skenario 1b

Estimasi Kecepatan							
CarID	Percobaan (km/h)					Rata-rata	Error (%)
	1	2	3	4	5		
1	77,46	77,33	76,77	76,83	77,72	77,22	5,67
2	78,62	78,62	77,21	77,34	78,42	78,04	4,38
3	83,38	83,30	81,63	82,02	83,13	82,69	2,09
4	84,59	85,17	84,01	84,09	85,08	84,59	0,95
5	82,34	82,26	80,33	80,55	82,18	81,53	0,65
Rata-rata							2,75

C. Skenario 1c

Tabel 5.5 *Error* sistem pada skenario 1c

Estimasi Kecepatan							
CarID	Percobaan (km/h)					Rata-rata	Error (%)
	1	2	3	4	5		
1	78,16	76,59	78,01	78,47	78,24	77,90	11,01
2	74,43	73,36	74,94	75,16	74,87	74,55	2,02
3	79,53	78,32	79,97	80,59	80,32	79,75	2,77
4	87,59	86,55	88,22	87,59	87,69	87,53	6,77
5	84,34	83,66	85,04	85,54	84,24	84,56	9,28
Rata-rata							6,37

Pada skenario ini, penulis akan membandingkan hasil dari sistem pada skenario 1c dengan skenario C. Dapat dilihat pada Tabel 5.5 bahwa setiap percobaan memiliki hasil yang berbeda. Kumpulan hasil kecepatan setiap mobil akan dirata-rata untuk mendapatkan *error* pada sistem.

D. Skenario 1d

Tabel 5.6 *Error* sistem pada skenario 1d

Estimasi Kecepatan							
CarID	Percobaan (km/h)					Rata-rata	Error (%)
	1	2	3	4	5		
1	73,78	73,61	73,44	74,40	74,05	73,86	2,58
2	70,44	70,69	69,45	70,94	69,94	70,29	4,55
3	76,76	77,19	76,34	77,61	76,45	76,87	0,83
4	86,41	86,28	85,50	87,08	86,02	86,26	6,49
5	76,76	76,66	75,93	77,72	77,08	76,83	1,59
Rata-rata							3,21

Pada skenario ini, penulis akan membandingkan hasil dari sistem pada skenario 1d dengan skenario D. Dapat dilihat pada Tabel 5.6 bahwa setiap percobaan memiliki hasil yang berbeda. Kumpulan hasil kecepatan setiap mobil akan dirata-rata untuk mendapatkan *error* pada sistem.

5.3.2. Skenario 2

A. Skenario 2a

Pada skenario ini, penulis akan membandingkan hasil dari sistem pada skenario 2a dengan skenario A. Dapat dilihat pada Tabel 5.7 bahwa setiap percobaan memiliki hasil yang berbeda. Kumpulan hasil kecepatan setiap mobil akan dirata-rata untuk mendapatkan *error* pada sistem.

Tabel 5.7 *Error* sistem pada skenario 2a

Estimasi Kecepatan							
CarID	Percobaan (km/h)					Rata-rata	Error (%)
	1	2	3	4	5		
1	71,63	72,65	71,49	71,63	71,99	71,88	0,17
2	68,86	70,34	69,89	69,26	69,22	69,51	7,08
3	74,41	75,80	75,52	74,52	74,57	74,96	4,99
4	80,99	82,67	81,63	81,50	80,80	81,52	5,89
5	73,91	75,28	73,96	74,23	73,17	74,11	8,01
Rata-rata							5,23

B. Skenario 2b

Tabel 5.8 *Error* sistem pada skenario 2b

Estimasi Kecepatan							
CarID	Percobaan (km/h)					Rata-rata	Error (%)
	1	2	3	4	5		
1	70,23	71,52	70,03	71,73	69,51	70,60	3,39
2	70,81	72,05	70,98	72,17	70,59	71,32	4,61
3	73,85	75,23	74,18	75,43	73,92	74,52	8,00
4	76,46	77,90	76,26	77,76	76,19	76,92	8,20
5	73,80	75,04	73,61	75,17	73,17	74,16	8,45
Rata-rata							6,53

Pada skenario ini, penulis akan membandingkan hasil dari sistem pada skenario 2b dengan skenario B. Dapat dilihat pada Tabel 5.8 bahwa setiap percobaan memiliki hasil yang berbeda. Kumpulan hasil kecepatan setiap mobil akan dirata-rata untuk mendapatkan *error* pada sistem.

C. Skenario 2c

Pada skenario ini, penulis akan membandingkan hasil dari sistem pada skenario 2c dengan skenario C. Dapat dilihat pada

Tabel 5.9 bahwa setiap percobaan memiliki hasil yang berbeda. Kumpulan hasil kecepatan setiap mobil akan dirata-rata untuk mendapatkan *error* pada sistem.

Tabel 5.9 *Error* sistem pada skenario 2c

Estimasi Kecepatan							
CarID	Percobaan (km/h)					Rata-rata	Error (%)
	1	2	3	4	5		
1	71,21	71,21	70,33	71,53	70,46	70,95	1,10
2	68,75	68,08	67,55	68,75	67,55	68,14	6,76
3	72,34	72,19	71,26	72,85	71,26	71,98	7,24
4	75,06	74,23	73,05	75,06	73,56	74,19	9,50
5	72,25	71,96	71,53	72,32	70,97	71,81	7,20
Rata-rata							6,36

D. Skenario 2d

Tabel 5.10 *Error* sistem pada skenario 2d

Estimasi Kecepatan							
CarID	Percobaan (km/h)					Rata-rata	Error (%)
	1	2	3	4	5		
1	66,76	66,62	66,69	66,98	68,07	67,02	6,91
2	63,90	63,63	63,63	63,63	64,73	63,90	13,22
3	68,77	68,68	68,68	68,51	69,81	68,89	9,64
4	77,99	77,99	78,20	77,56	78,63	78,07	3,61
5	69,57	69,05	68,96	69,22	69,74	69,31	11,22
Rata-rata							8,92

Pada skenario ini, penulis akan membandingkan hasil dari sistem pada skenario 2d dengan skenario D. Dapat dilihat pada Tabel 5.10 bahwa setiap percobaan memiliki hasil yang berbeda. Kumpulan hasil kecepatan setiap mobil akan dirata-rata untuk mendapatkan *error* pada sistem.

5.3.3. Skenario 3

A. Skenario 3a

Tabel 5.11 *Error* sistem pada skenario 3a

Estimasi Kecepatan							
CarID	Percobaan (km/h)					Rata-rata	Error (%)
	1	2	3	4	5		
1	66,43	66,71	63,31	67,03	66,55	66,00	8,33
2	65,00	64,65	63,77	65,12	64,77	64,66	13,56
3	70,13	69,75	68,89	70,08	69,79	69,73	11,62
4	75,77	75,83	74,95	75,55	75,66	75,55	12,78
5	69,05	68,91	67,94	69,38	69,28	68,91	14,46
Rata-rata							12,15

Pada skenario ini, penulis akan membandingkan hasil dari sistem pada skenario 3a dengan skenario A. Dapat dilihat pada Tabel 5.11 bahwa setiap percobaan memiliki hasil yang berbeda. Kumpulan hasil kecepatan setiap mobil akan dirata-rata untuk mendapatkan *error* pada sistem.

B. Skenario 3b

Tabel 5.12 *Error* sistem pada skenario 3b

Estimasi Kecepatan							
CarID	Percobaan (km/h)					Rata-rata	Error (%)
	1	2	3	4	5		
1	64,37	63,94	65,63	63,98	63,98	64,38	11,90
2	65,12	65,07	65,83	65,02	64,98	65,20	12,80
3	67,94	67,83	68,77	68,05	67,94	68,11	15,92
4	70,16	69,76	70,79	69,82	69,76	70,06	16,39
5	67,65	67,60	69,30	67,87	67,55	67,99	16,06
Rata-rata							14,61

Pada skenario ini, penulis akan membandingkan hasil dari sistem pada skenario 3b dengan skenario B. Dapat dilihat pada Tabel 5.12 bahwa setiap percobaan memiliki hasil yang berbeda. Kumpulan hasil kecepatan setiap mobil akan dirata-rata untuk mendapatkan *error* pada sistem.

C. Skenario 3c

Pada skenario ini, penulis akan membandingkan hasil dari sistem pada skenario 3c dengan skenario C. Dapat dilihat pada Tabel 5.13 bahwa setiap percobaan memiliki hasil yang berbeda. Kumpulan hasil kecepatan setiap mobil akan dirata-rata untuk mendapatkan *error* pada sistem.

Tabel 5.13 *Error* sistem pada skenario 3c

Estimasi Kecepatan							
CarID	Percobaan (km/h)					Rata-rata	Error (%)
	1	2	3	4	5		
1	64,60	64,55	64,44	64,55	65,29	64,69	7,82
2	61,65	61,60	62,09	61,84	62,50	61,94	15,25
3	65,30	65,24	65,53	65,24	66,02	65,46	15,64
4	67,29	67,36	67,17	67,79	68,23	67,57	17,58
5	65,19	65,37	65,14	65,14	65,85	65,34	15,56
Rata-rata							14,37

D. Skenario 3d

Pada skenario ini, penulis akan membandingkan hasil dari sistem pada skenario 3d dengan skenario D. Dapat dilihat pada Tabel 5.14 bahwa setiap percobaan memiliki hasil yang berbeda. Kumpulan hasil kecepatan setiap mobil akan dirata-rata untuk mendapatkan *error* pada sistem.

Tabel 5.14 *Error* sistem pada skenario 3d

Estimasi Kecepatan							
CarID	Percobaan (km/h)					Rata-rata	Error (%)
	1	2	3	4	5		
1	61,38	61,08	61,08	61,80	61,74	61,42	14,70
2	59,10	58,29	58,63	59,39	59,27	58,94	19,97
3	63,04	62,83	63,04	64,14	64,06	63,42	16,81
4	71,06	71,50	71,15	72,14	71,78	71,52	11,70
5	63,24	63,24	63,09	64,48	64,19	63,65	18,47
Rata-rata							16,33

5.3.4. Skenario 4

A. Skenario 4a

Pada skenario ini, penulis akan membandingkan hasil dari sistem pada skenario 4a dengan skenario A. Dapat dilihat pada Tabel 5.15 bahwa setiap percobaan memiliki hasil yang berbeda. Kumpulan hasil kecepatan setiap mobil akan dirata-rata untuk mendapatkan *error* pada sistem.

Tabel 5.15 *Error* sistem pada skenario 4a

Estimasi Kecepatan							
CarID	Percobaan (km/h)					Rata-rata	Error (%)
	1	2	3	4	5		
1	54,80	54,40	53,95	54,61	54,00	54,35	24,51
2	52,94	52,94	53,02	52,97	52,07	52,79	29,44
3	57,16	57,06	57,22	57,09	56,18	56,94	27,83
4	62,02	61,87	62,05	62,13	61,36	61,89	28,56
5	56,95	56,67	56,83	56,98	55,77	56,64	29,69
Rata-rata							28,01

B. Skenario 4b

Pada skenario ini, penulis akan membandingkan hasil dari sistem pada skenario 4b dengan skenario B. Dapat dilihat pada

Tabel 5.16 bahwa setiap percobaan memiliki hasil yang berbeda. Kumpulan hasil kecepatan setiap mobil akan dirata-rata untuk mendapatkan *error* pada sistem.

Tabel 5.16 *Error* sistem pada skenario 4b

Estimasi Kecepatan							
CarID	Percobaan (km/h)					Rata-rata	Error (%)
	1	2	3	4	5		
1	52,58	53,20	52,46	52,78	52,40	52,69	27,91
2	53,26	53,79	53,51	53,60	53,32	53,50	28,45
3	55,72	56,35	55,94	55,83	55,72	55,91	30,98
4	57,47	57,82	57,59	57,36	57,17	57,48	31,40
5	55,82	56,41	55,79	55,86	55,68	55,91	30,97
Rata-rata							29,94

C. Skenario 4c

Tabel 5.17 *Error* sistem pada skenario 4c

Estimasi Kecepatan							
CarID	Percobaan (km/h)					Rata-rata	Error (%)
	1	2	3	4	5		
1	53,32	52,76	53,79	53,47	53,57	53,38	23,92
2	51,39	50,98	51,22	51,43	51,32	51,27	29,84
3	54,32	54,08	54,32	54,32	54,28	54,27	30,07
4	55,84	55,17	55,97	55,80	55,67	55,69	32,07
5	54,21	53,21	54,26	54,21	53,89	53,96	30,27
Rata-rata							29,24

Pada skenario ini, penulis akan membandingkan hasil dari sistem pada skenario 4c dengan skenario C. Dapat dilihat pada Tabel 5.17 bahwa setiap percobaan memiliki hasil yang berbeda. Kumpulan hasil kecepatan setiap mobil akan dirata-rata untuk mendapatkan *error* pada sistem.

D. Skenario 4d

Pada skenario ini, penulis akan membandingkan hasil dari sistem pada skenario 4d dengan skenario D. Dapat dilihat pada Tabel 5.18 bahwa setiap percobaan memiliki hasil yang berbeda. Kumpulan hasil kecepatan setiap mobil akan dirata-rata untuk mendapatkan *error* pada sistem.

Tabel 5.18 *Error* sistem pada skenario 4d

Estimasi Kecepatan							
CarID	Percobaan (km/h)					Rata-rata	Error (%)
	1	2	3	4	5		
1	50,79	49,94	50,54	50,83	50,10	50,44	29,94
2	48,50	47,87	48,34	48,50	47,95	48,23	34,50
3	52,26	51,43	52,16	52,16	51,62	51,93	31,89
4	58,96	58,71	59,14	59,08	58,23	58,82	27,38
5	52,64	51,80	52,49	52,44	51,61	52,20	33,14
Rata-rata							31,37

5.4. Evaluasi Pengujian

Tabel 5.19 Rangkuman Hasil Pengujian *Error* Sistem

Skenario Start/Finish Line	Skenario FPS			
	1	2	3	4
a	4,19	5,23	12,15	28,01
b	2,75	6,53	14,61	29,94
c	6,37	6,36	14,37	29,24
d	3,21	8,92	16,33	31,37
Rata-rata	4,13	6,76	14,37	29,64

Dapat dilihat pada Tabel 5.19, bahwa skenario FPS 1 (30 FPS) memiliki *error* paling kecil. Hal ini dapat disimpulkan bahwa pengurangan FPS meningkatkan *error* pada sistem, sehingga hasil yang terbaik terdapat pada video dengan 30 FPS. Dapat dilihat pada tabel tersebut, bahwa peletakan garis awal dan akhir yang memiliki *error* paling kecil adalah skenario b pada video dengan 30 FPS. Oleh karena itu, hasil yang terbaik untuk keseluruhan terdapat pada skenario dengan video 30 FPS dan peletakan garis akhir ((282,0) dan (282,240)) dengan jarak sebenarnya 27 meter.

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 aplikasi yang dilakukan, dapat diambil kesimpulan sebagai berikut:

1. *Error* terkecil yang dihasilkan oleh sistem sebesar 2,75%.
2. Pengurangan FPS pada video dapat meningkatkan *error* pada sistem. Video dengan 30 FPS memiliki hasil yang terbaik.
3. Peletakan dan garis akhir berpengaruh pada *error* yang dihasilkan oleh sistem. Peletakan garis akhir ((282,0) dan (282,240)) dengan jarak sebenarnya 27 meter memiliki hasil yang paling baik.

6.2. Saran

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

1. Penggunaan *enhancement* pada gambar yang mungkin berpotensi mempengaruhi hasil deteksi kecepatan.
2. Peningkatan tampilan antarmuka pada aplikasi.

[Halaman ini sengaja dikosongkan]

DAFTAR PUSTAKA

[1] A. G. Rad, A. Dehghani, and M. R. Karim, "Vehicle speed detection in video image sequences using CVS method," *Int. J. Phys. Sci.*, vol. 5, no. 17, pp. 2555–2563, 2010.

[2] H. S. Sundoro and A. Harjoko, "VEHICLE COUNTING AND VEHICLE SPEED MEASUREMENT BASED ON VIDEO PROCESSING," *J. Theor. Appl. Inf. Technol.*, vol. 84, no. 2, p. 233, 2016.

[3] "OpenCV," *Wikipedia*. 29-Apr-2017.

[4] "OpenCV library." [Online]. Available: http://opencv.org/. [Accessed: 07-May-2017].

[5] "OpenCL," *Wikipedia*. 25-Apr-2017.

[6] "Frame rate," *Wikipedia*. 29-Apr-2017.

[7] "Background subtraction," *Wikipedia*. 08-Feb-2017.

[8] "OpenCV: Background Subtraction." [Online]. Available: http://docs.opencv.org/trunk/db/d5c/tutorial_py_bg_subtraction.html. [Accessed: 07-May-2017].

[9] Z. Zivkovic, "Improved adaptive Gaussian mixture model for background subtraction," in *Pattern Recognition, 2004. ICPR 2004. Proceedings of the 17th International Conference on*, 2004, vol. 2, pp. 28–31.

[10] Z. Zivkovic and F. van der Heijden, "Efficient adaptive density estimation per image pixel for the task of background subtraction," *Pattern Recognit. Lett.*, vol. 27, no. 7, pp. 773–780, May 2006.

[11] "e (mathematical constant)," *Wikipedia*. 06-May-2017.

[12] "Region of interest," *Wikipedia*. 12-Jan-2017.

[13] S. Suzuki and Abe, Keiichi, "Topological structural analysis of digitized binary images by border following," *Comput. Vis. Graph. Image Process.*, vol. 30, no. 1, pp. 32–46, 1985.

[14] C. Chia Yik, "Vehicle Tracking and Speed Estimation System," University Malaysia Pahang, Malaysia, Jun. 2012.

[15] "Structural Analysis and Shape Descriptors — OpenCV 2.4.13.2 documentation." [Online]. Available: http://docs.opencv.org/2.4/modules/imgproc/doc/structural_

analysis_and_shape_descriptors.html. [Accessed: 07-May-2017].

BIODATA PENULIS

Andrew, lahir pada tanggal 4 Februari 1995 di Surabaya. Penulis menempuh pendidikan perguruan tinggi di Institut Teknologi Sepuluh Nopember Surabaya di Jurusan Teknik Informatika Fakultas Teknologi Informasi pada tahun 2013. Penulis memiliki pengalaman menjadi asisten dosen pada mata kuliah Matematika Diskrit, Manajemen Basis Data dan Analisis Perancangan Sistem Informasi. Dalam melakukan pengerjaan Tugas Akhir, penulis memiliki ketertarikan pada bidang Dasar dan Terapan Komputasi (DTK). Untuk menghubungi penulis dapat melalui *email*: andrewwidjaja@outlook.com.