



TUGAS AKHIR -SM141501

**PENGEMBANGAN PERANGKAT LUNAK DETEKSI
KECEPATAN KENDARAAN BERGERAK BERBASIS
PENGOLAHAN CITRA DIGITAL**

**FARAH FAJRIYAH
NRP 1211 100 015**

**Dosen Pembimbing
Dr. Budi Setiyono, S.Si, MT**

**JURUSAN MATEMATIKA
Fakultas Matematika dan Ilmu Pengetahuan Alam
Institut Teknologi Sepuluh Nopember
Surabaya 2016**



FINAL PROJECT -SM141501

SOFTWARE DEVELOPMENT FOR SPEED DETECTION OF MOVING VEHICLE BASED ON DIGITAL IMAGE PROCESSING

FARAH FAJRIYAH
NRP 1211 100 015

Supervisor
Dr. Budi Setiyono, S.Si, MT

DEPARTMENT OF MATHEMATICS
Faculty of Mathematics and Natural Science
Sepuluh Nopember Institute of Technology
Surabaya 2016

LEMBAR PENGESAHAN

PENGEMBANGAN PERANGKAT LUNAK DETEKSI KECEPATAN KENDARAAN BERGERAK BERBASIS PENGOLAHAN CITRA DIGITAL

SOFTWARE DEVELOPMENT FOR SPEED DETECTION OF MOVING VEHICLE BASED ON DIGITAL IMAGE PROCESSING

TUGAS AKHIR


Diajukan Untuk Memenuhi Salah Satu Syarat
Untuk Memperoleh Gelar Sarjana Sains
Pada Bidang Studi Ilmu Komputer
Program Studi S-1 Jurusan Matematika
Fakultas Matematika dan Ilmu Pengetahuan Alam
Institut Teknologi Sepuluh Nopember Surabaya

Oleh :

FARAH FAJRIYAH
NRP. 1211 100 015

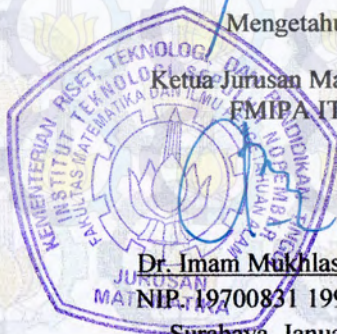
Menyetujui,

Dosen Pembimbing,


Dr. Budi Setiyono, S.Si, MT
NIP. 19720207 199702 1 001

Mengetahui,

Ketua Jurusan Matematika
FMIPA ITS


Dr. Imam Mukhlis, S.Si, MT
NIP. 19700831 199403 1 003

Surabaya, Januari 2016

KATA PENGANTAR

Segala Puji bagi Allah SWT Tuhan semesta alam yang telah memberikan karunia, rahmat dan anugerah-Nya sehingga penulis dapat menyelesaikan Tugas Akhir yang berjudul: **“Pengembangan Perangkat Lunak Deteksi Kecepatan Kendaraan Bergerak Berbasis Pengolahan Citra Digital”** yang merupakan salah satu persyaratan akademis dalam menyelesaikan Program Studi S-1 pada Jurusan Matematika Fakultas Matematika dan Ilmu Pengetahuan Alam Institut Teknologi Sepuluh Nopember Surabaya.

Tugas Akhir ini dapat diselesaikan dengan berkat kerjasama, bantuan, dan dukungan dari banyak pihak. Sehubungan dengan hal itu, penulis mengucapkan terima kasih kepada:

1. Dr. Budi Setiyono, S.Si, MT selaku dosen pembimbing yang senantiasa membimbing dengan sabar dan memberikan kritik dan saran dalam penyusunan Tugas Akhir ini.
2. Dr. Imam Mukhlas, S.Si, MT selaku Ketua Jurusan Matematika.
3. Dra. Farida Agustini Widjajati, MS. selaku Dosen Wali.
4. Dr. Imam Mukhlash, S.Si, MT, Dr. Chairul Imron, MI.Komp., Drs. Soetrisno, MI.Komp., M. Syifa'ul Mufid, S.Si, M.Si selaku dosen penguji Tugas Akhir ini.
5. Dr. Chairul Imron, MI.Komp. selaku Koordinator Tugas Akhir.
6. Seluruh jajaran dosen dan staf jurusan Matematika ITS.
7. Teman-teman mahasiswa jurusan Matematika ITS

Penulis menyadari bahwa Tugas Akhir ini masih jauh dari kesempurnaan. Oleh karena itu, penulis mengharapkan saran dan kritik dari pembaca. Akhir kata, semoga Tugas Akhir ini bermanfaat bagi semua pihak yang berkepentingan.

Surabaya, Januari 2016
Penulis

special thanks to

Selama proses pembuatan Tugas Akhir ini, banyak pihak yang telah memberikan bantuan dan dukungan untuk penulis. Penulis mengucapkan terima kasih dan apresiasi secara khusus kepada:

1. Ayah Moh. Kholil dan Ibu Achadah tersayang yang senantiasa dengan ikhlas memberikan semangat, perhatian, kasih sayang, doa, motivasi dan nasihat yang sangat berarti bagi penulis.
2. Kakak Wawan Heri Zunaidi yang menjadi salah satu motivasi bagi penulis untuk segera menyelesaikan Tugas Akhir ini.
3. Teman-teman pejuang 112 serta teman-teman pejuang 113 yang tidak bisa disebutkan satu per satu yang telah bersama-sama penulis berjuang dalam suka dan duka.
4. Mas Danang, Habib dan teman-teman yang sudah membantu dalam mamahami dan mengerjakan Tugas Akhir.
5. Sahabat F6 yaitu Diani, Bilqis, Mas Danis, Mas Makki, Rizqi dan Sahabat Asrama yaitu Mbak Hesti, Yessy, Muna, Tuffahatul serta adik-adik kamarku yaitu Ika dan Luthfi yang selalu menjadi tempat *curhat* dan pendengar yang baik bagi penulis serta tidak bosan dalam memberi semangat dan saran untuk menyelesaikan Tugas Akhir ini.
6. Tuffahatul, Muna, Lusi, Faing, Dina, Bundo, Fikri, Wiwit, Rimoed, Angga, Desy dan teman-teman angkatan 2011 yang tidak bisa disebutkan satu per satu, terimakasih atas segala bentuk semangat dan dukungannya kepada penulis.
7. Keluarga besar Cinta Rebana ITS yang tidak bisa disebutkan satu persatu, terimakasih atas segala bentuk semangat, do'a dan dukungannya kepada penulis.

Tentu saja masih banyak pihak lain yang turut andil dalam penyelesaian tugas akhir ini yang tidak bisa penulis sebutkan satu persatu. Semoga Allah membalas dengan balasan yang lebih baik bagi semua pihak yang telah membantu penulis. *Aamiin yaa rabbal 'aalamiin.*

DAFTAR ISI

	Halaman
HALAMAN JUDUL	i
LEMBAR PENGESAHAN	v
ABSTRAK	vii
ABSTRACT	ix
KATA PENGANTAR	xi
DAFTAR ISI	xiii
DAFTAR GAMBAR	xvii
DAFTAR TABEL	xix
DAFTAR LAMPIRAN	xxi
BAB I. PENDAHULUAN	
1.1 Latar Belakang Masalah	1
1.2 Rumusan Masalah.....	3
1.3 Batasan Masalah	3
1.4 Tujuan Penelitian	3
1.5 Manfaat Penelitian	3
1.6 Sistematika Penulisan	3
BAB II. DASAR TEORI	
2.1 Definisi Citra Digital	6
2.2 Video Digital	7
2.3 Algoritma <i>Background Subtraction</i>	8
2.4 <i>Tracking</i>	8
2.5 Metode <i>Gaussian Mixture Model</i>	8
2.5.1 Tahap Pencocokan <i>Input</i> Terhadap Distribusi	10
2.5.2 Tahap Pemilihan Distribusi yang Mencerminkan <i>Background</i>	11
2.6 Metode Analisis <i>Blob</i>	13
2.7 Formula untuk Perhitungan Kecepatan Kendaraan dengan Posisi Kamera di Atas	14
2.7.1 Pengujian Kamera.....	14

2.7.2	Perhitungan Sudut dan Bidang Tegak Lurus Pandang.....	14
2.8	<i>Speed Detection</i>	16
BAB III. METODOLOGI		
3.1	Studi Literatur dan Perekaman Video.....	17
3.2	Analisis dan Desain Sistem.....	17
3.3	Implementasi Sistem.....	18
3.4	Uji Coba dan Pembahasan	18
3.5	Penarikan Kesimpulan	19
3.6	Penulisan laporan Tugas Akhir.....	19
BAB IV. PERANCANGAN DAN IMPLEMENTASI SISTEM		
4.1	Analisis Sistem.....	21
4.1.1	Analisis Sistem Perangkat Lunak	20
4.1.2	Analisis Kebutuhan Sistem	25
4.2	Perancangan Sistem	28
4.2.1	Perancangan Data Sistem.....	28
4.2.1.1	Data Masukan	29
4.2.1.2	Data Proses	30
4.2.1.3	Data Keluaran	31
4.2.2	Perancangan Proses.....	31
4.2.2.1	Perancangan Proses Algoritma	31
4.2.2.2	Perancangan Proses GMM	33
4.2.2.3	Perancangan Proses Deteksi <i>Blob</i>	34
4.2.2.4	Perancangan Proses Deteksi Kecepatan Kendaraan	36
4.2.3	Perancangan Antar Muka Sistem.....	40
4.2.3.1	Perancangan Halaman Utama	40
4.2.3.2	Perancangan Halaman Detail.....	41
4.3	Implementasi Sistem.....	42
4.3.1	Implementasi Input Video	42
4.3.2	Implementasi Pemilihan Area ROI.....	43
4.3.3	Implementasi Proses GMM	44
4.3.4	Implementasi <i>Filtering</i> Citra	45
4.3.5	Implementasi Deteksi <i>Blob</i>	46

4.3.6 Implementasi Pendeteksian Kecepatan Kendaraan	46
---	----

BAB V. UJI COBA DAN PEMBAHASAN

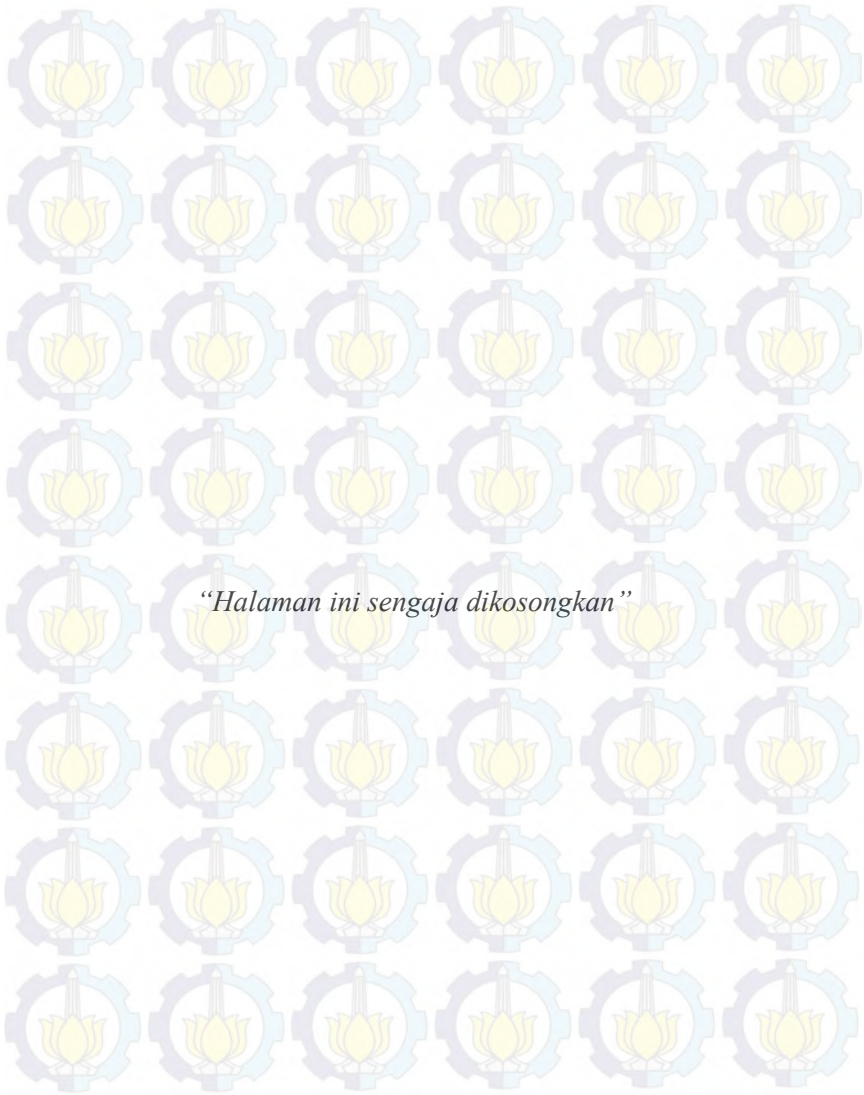
5.1 Data Uji Coba	49
5.2 Uji Coba Pendeteksian Kecepatan Kendaraan Bergerak	50
5.2.1 Rekaman Video Movie1.avi	51
5.2.2 Rekaman Video Movie2.avi	52
5.2.3 Rekaman Video Movie3.avi	52
5.2.4 Rekaman Video Movie4.avi	53
5.2.5 Rekaman Video Movie5.avi	53
5.2.6 Rekaman Video Movie6.avi	54
5.2.7 Rekaman Video Movie0.avi	55
5.2.8 Rekaman Video Movie01.avi	56
5.2.9 Rekaman Video Movie12.avi	57
5.2.10 Rekaman Video DeltaMobil.avi	58
5.2.11 Rekaman Video DeltaMobil2.avi	58
5.2.12 Rekaman Video jalantol99.avi	59
5.2.13 Rekaman video jalantol100.avi	60
5.2.14 Rekaman video jalantol105a.avi	61
5.2.15 Rekaman video jalantol105b.avi	61
5.2.16 Rekaman video jalantol106a.avi.....	62
5.2.17 Rekaman video jalantol109b.avi	62
5.2.18 Pembahasan Uji Coba Pendeteksian Kecepatan Kendaraan Bergerak.....	63

BAB VI.PENUTUP

6.1 Kesimpulan	71
6.2 Saran	72

DAFTAR PUSTAKA	73
-----------------------------	----

LAMPIRAN	75
-----------------------	----



“Halaman ini sengaja dikosongkan”

DAFTAR GAMBAR

	Halaman
Gambar 2.1 Diagram Alir Proses GMM	12
Gambar 2.2 Analisis <i>Blob</i>	13
Gambar 2.3 <i>Setting</i> Kamera dengan Posisi di Atas	14
Gambar 2.4 Kemiringan Kamera Hasil Rekaman dari Atas	15
Gambar 2.5 Tampilan <i>grid</i> Hasil Rekaman dari Atas	16
Gambar 3.1 Diagram Metodologi Penelitian	20
Gambar 4.1 <i>Use Case Diagram</i> Perangkat Lunak	22
Gambar 4.2 <i>Activity Diagram</i> Perangkat Lunak	23
Gambar 4.3 Proses <i>Scanning</i> Video	24
Gambar 4.4 Citra dengan ROI	25
Gambar 4.5 Citra Hasil <i>Background Subtraction</i>	25
Gambar 4.6 Citra Hasil Deteksi <i>Blob</i>	26
Gambar 4.7 Citra Hasil Deteksi	27
Gambar 4.8 <i>Business Rule</i> Sistem Perangkat Lunak	28
Gambar 4.9 <i>Layout</i> untuk Pengambilan Video	29
Gambar 4.10 Diagram Alir Sistem Perangkat Lunak	32
Gambar 4.11 Diagram Alir GMM	33
Gambar 4.12 Diagram Alir Deteksi <i>Blob</i>	35
Gambar 4.13 Diagram Alir Deteksi Kecepatan	39
Gambar 4.14 Antar Muka Halaman Awal	41
Gambar 4.15 Antar Muka Halaman Detail	42
Gambar 4.16 Antar Muka <i>Input</i> Video	43
Gambar 4.17 Antar Muka Area dengan ROI	44
Gambar 4.18 Citra Hasil Morfologi	45
Gambar 4.19 Antar Muka Aplikasi Hasil Deteksi	48



“Halaman ini sengaja dikosongkan”

DAFTAR TABEL

	Halaman
Tabel 4.1	Tabel Data Proses 30
Tabel 5.1	Tabel Data Video yang Digunakan 49
Tabel 5.2	Tabel Pendeteksian Kecepatan Kendaraan pada Video Movie1.avi 51
Tabel 5.3	Tabel Pendeteksian Kecepatan Kendaraan pada Video Movie2.avi 52
Tabel 5.4	Tabel Pendeteksian Kecepatan Kendaraan pada Video Movie3.avi 52
Tabel 5.5	Tabel Pendeteksian Kecepatan Kendaraan pada Video Movie4.avi 53
Tabel 5.6	Tabel Pendeteksian Kecepatan Kendaraan pada Video Movie5.avi 53
Tabel 5.7	Tabel Pendeteksian Kecepatan Kendaraan pada Video Movie6.avi 54
Tabel 5.8	Tabel Pendeteksian Kecepatan Kendaraan pada Video Movie0.avi 55
Tabel 5.9	Tabel Pendeteksian Kecepatan Kendaraan pada Video Movie01.avi 56
Tabel 5.10	Tabel Pendeteksian Kecepatan Kendaraan pada Video Movie12.avi 57
Tabel 5.11	Tabel Pendeteksian Kecepatan Kendaraan pada Video DeltaMobil.avi 58
Tabel 5.12	Tabel Pendeteksian Kecepatan Kendaraan pada Video DeltaMobil2.avi 58
Tabel 5.13	Tabel Pendeteksian Kecepatan Kendaraan pada Video jalantol99.avi 59
Tabel 5.14	Tabel Pendeteksian Kecepatan Kendaraan pada Video jalantol100.avi 60
Tabel 5.15	Tabel Pendeteksian Kecepatan Kendaraan pada Video jalantol105a.avi 61
Tabel 5.16	Tabel Pendeteksian Kecepatan Kendaraan pada Video jalantol105b.avi 61

Tabel 5.17	Tabel Pendeteksian Kecepatan Kendaraan pada Video jalantol106a.avi	62
Tabel 5.18	Tabel Pendeteksian Kecepatan Kendaraan pada Video jalantol109b.avi	62
Tabel 5.19	Tabel Prosentase Keberhasilan Video Movie1.avi	64
Tabel 5.20	Tabel Prosentase Keberhasilan Video Movie2.avi	64
Tabel 5.21	Tabel Prosentase Keberhasilan Video Movie3.avi	65
Tabel 5.22	Tabel Prosentase Keberhasilan Video Movie4.avi	65
Tabel 5.23	Tabel Prosentase Keberhasilan Video Movie5.avi	66
Tabel 5.24	Tabel Prosentase Keberhasilan Video Movie6.avi	66
Tabel 5.25	Tabel Prosentase Keberhasilan Video Movie0.avi	67
Tabel 5.26	Tabel Prosentase Keberhasilan Video Movie01.avi	67
Tabel 5.27	Tabel Prosentase Keberhasilan Video Movie12.avi	68



BAB I

PENDAHULUAN

1.1 Latar Belakang Masalah

Volume kepadatan arus lalu lintas setiap tahunnya mengalami peningkatan. Selain faktor kemudahan memiliki kendaraan bermotor, pertumbuhan penduduk yang cukup tinggi membuat kebutuhan akan transportasi juga semakin meningkat. Berdasarkan data dari Badan Pusat Statistik jumlah kendaraan bermotor pada tahun 2013 mencapai 104.118.969 unit[1]. Namun, pertumbuhan jumlah kendaraan tersebut tidak diimbangi dengan fasilitas jalan yang memadai, sehingga timbul permasalahan-permasalahan lalu lintas.

Salah satu diantaranya yaitu tidak seimbangnya jumlah kendaraan dengan ruas jalan. Hal ini mengakibatkan kondisi lalu lintas sehari-hari mengalami kepadatan. Hal ini diperparah dengan cara berkendara yang tidak sesuai dengan aturan, salah satunya adalah kebut-kebutan yang dilakukan oleh para pengguna kendaraan, sehingga rawan terjadi kecelakaan. Bahkan kecelakaan lalu lintas menjadi salah satu penyebab kematian terbesar di Indonesia maupun secara global. Maka dari itu perlu dilakukan upaya untuk menertibkan perilaku berbahaya ini. Penentuan batas kecepatan maksimum yang ditunjukkan dengan rambu lalu lintas adalah salah satu aturan efektif untuk menanggulangi kebut-kebutan. Namun tanpa ada petugas lalu lintas yang berjaga, pelanggaran kerap terjadi. Maka pengamatan kecepatan sangat penting untuk membantu pengawasan lalu lintas.

Dalam mengamati kecepatan kendaraan, kebanyakan masih menggunakan metode LIDAR (*Laser Infrared Detection and Ranging*) atau RADAR (*Radio Detection and Ranging*). Metode ini digunakan untuk mengurangi pelanggaran lalu lintas batas kecepatan kendaraan. Tetapi metode ini membutuhkan biaya yang besar untuk membeli alatnya serta membutuhkan keahlian khusus untuk mengoperasikan alatnya. Sebagai alternatif lain, saat ini

pengawasan lalu lintas telah memanfaatkan kamera CCTV yaitu *road traffic monitoring center (RTMC)*. Sejalan dengan hal tersebut, saat ini pemerintah sedang membangun infrastruktur jalan, terbukti dengan pemasangan kamera CCTV pada perempatan jalan yang sudah banyak dilakukan. Hal ini dilakukan karena dianggap mampu melingkupi wilayah yang luas, sehingga diharapkan dapat meningkatkan keamanan. Namun pemasangan CCTV tersebut hanya dimanfaatkan sebagai pengawasan kepadatan lalu lintas saja, padahal video hasil dari rekaman kamera CCTV tersebut dapat juga digunakan untuk mendeteksi kecepatan kendaraan.

Beberapa penelitian yang terkait dengan Tugas Akhir ini diantaranya adalah penelitian yang dilakukan oleh Huei-Yung Lin, dkk dengan judul "*Vehicle Speed Detection from a Single Motion Blurred Image*"[2], kemudian penelitian yang dilakukan oleh Chomtip Pornpanomchai, dkk yang berjudul "*Vehicle Speed Detection System*"[3] dan penelitian yang dilakukan oleh Arash dkk, dengan judul penelitiannya "*Vehicle Speed Detection in Video Image Sequence using CVS Method*"[4]. Dari hasil penelitian tersebut, tingkat akurasi pendeteksian kecepatan kendaraan dipengaruhi oleh intensitas cahaya yang diterima oleh video digital. Sehingga menyebabkan ketidakstabilan dalam pendeteksian kecepatan kendaraan.

Oleh karena itu, berdasarkan paparan latar belakang di atas, maka dalam Tugas Akhir ini akan dilakukan pendeteksian kecepatan kendaraan yang berbasis pengolahan citra digital dengan memanfaatkan video hasil rekaman kamera CCTV. Dalam penelitian ini penulis berencana membuat sistem perangkat lunak berbasis algoritma *Background Subtraction* untuk mendeteksi kecepatan kendaraan menggunakan metode *Gaussian Mixture Model* yang mampu beradaptasi dengan perubahan intensitas cahaya matahari.

1.2 Rumusan Masalah

Rumusan masalah dari Tugas Akhir ini adalah bagaimana membuat perangkat lunak untuk mendeteksi kecepatan kendaraan bergerak yang memanfaatkan informasi dari video digital?

1.3 Batasan Masalah

Batasan-batasan yang ada dalam pembuatan Tugas Akhir ini adalah :

1. Data yang digunakan adalah rekaman video kendaraan yang bergerak di jalan.
2. Sudut pandang yang diambil ketika merekam adalah dari atas dengan jangkauan pandangan seluruh jalan arus satu arah.
3. Menggunakan kamera digital.
4. Deteksi kecepatan kendaraan dilakukan pada objek yang masuk pada daerah ROI.
5. Deteksi kecepatan kendaraan dilakukan untuk *single obyek*.

1.4 Tujuan Penelitian

Tujuan dari Tugas Akhir ini adalah membuat dan menganalisa kinerja perangkat lunak untuk mendeteksi kecepatan kendaraan bergerak yang memanfaatkan informasi dari video digital.

1.5 Manfaat Penelitian

Manfaat yang diperoleh dari Tugas Akhir ini adalah sistem yang telah dibangun dapat memberikan informasi tentang kecepatan kendaraan yang melaju di jalan sehingga nantinya dapat dijadikan bahan pertimbangan bagi instansi pemerintah terkait untuk merumuskan kebijakan rekayasa lalu lintas seperti perencanaan pembangunan jaringan jalan yang tepat dan sesuai dengan beban yang melintasinya.

1.6 Sistematika Penulisan

Sistematika Penulisan Tugas Akhir ini dibagi menjadi lima bab. Isi dari masing- masing bab tersebut adalah sebagai berikut :

1. Bab I Pendahuluan

Pada bab ini dibahas tentang latar belakang, rumusan masalah, batasan masalah, tujuan dan manfaat dari penelitian yang dilakukan, serta sistematika penulisan Tugas Akhir.

2. Bab II Dasar Teori

Pada bab ini diuraikan beberapa teori pendukung yang berasal dari jurnal dan buku yang berkaitan atau yang melandasi pembahasan pada penelitian ini untuk membantu menyelesaikan permasalahan Tugas Akhir.

3. Bab III Metodologi

Pada bab ini berisi tentang langkah-langkah yang digunakan untuk menyelesaikan Tugas Akhir.

4. Bab IV Perancangan dan Implementasi Sistem

Bab ini menjelaskan tentang beberapa ketentuan dalam pengambilan data rekaman di jalan raya. Data rekaman tersebut akan digunakan sebagai masukan program yang dibangun. Pada bab ini juga dijelaskan mengenai analisis sistem yang berisi aturan main berjalannya program dari awal hingga akhir. Dan bagaimana proses pengolahan informasi video digital sehingga dapat dimanfaatkan untuk mendeteksi kecepatan kendaraan yang bergerak.

Dari hasil analisis, kemudian diimplementasikan menggunakan bahasa pemrograman yang telah ditentukan untuk mendeteksi kecepatan kendaraan bergerak di jalan.

5. Bab V Uji Coba dan Pembahasan

Bab ini menjelaskan mengenai hasil uji coba program dari beberapa rekaman video arus lalu lintas. Dan pembahasan dari setiap rekaman yang ada untuk kemudian dicatat sebagai bahan pertimbangan merumuskan beberapa kesimpulan dan saran.

6. Bab VI Penutup

Bab ini merupakan penutup, berisi tentang kesimpulan yang dapat diambil berdasarkan hasil uji coba dan saran yang selanjutnya dilakukan bila Tugas Akhir ini dilanjutkan.

BAB II

DASAR TEORI

Bab ini menjelaskan tentang kajian teori dari referensi penunjang serta penjelasan permasalahan yang dibahas dalam tugas akhir ini, meliputi Penelitian Sebelumnya dan beberapa dasar-dasar teori yang akan dijelaskan pada sub bab selanjutnya.

Penelitian Sebelumnya

Terdapat beberapa penelitian dengan tema yang terkait dengan Tugas Akhir ini diantaranya yaitu penelitian yang dilakukan oleh Huei-Yung Lin, dkk (2006). Dalam penelitiannya yang berjudul "*Vehicle Speed Detection from a Single Motion Blurred Image*" telah berhasil mendeteksi kecepatan kendaraan dengan akurasi keberhasilan maksimal 95%[2]. Dalam penelitian tersebut, posisi kamera yang digunakan untuk menangkap gambarnya adalah posisi di samping, dan untuk mendeteksi kecepatan kendaraannya dengan menganalisa hasil *motion blur*. Pada tahun 2009 Chomtip Pornpanomchai, dkk juga melakukan pendeteksian kecepatan kendaraan. Dalam penelitiannya yang berjudul "*Vehicle Speed Detection System*" perangkat lunak yang dibangun mampu menghitung kecepatan kendaraan dengan tingkat keberhasilan maksimal 95%[3]. Penelitian berikutnya menggunakan metode CVS (*Combinaton of Saturation and Value*) juga berhasil diterapkan untuk mendeteksi kecepatan kendaraan dengan akurasi keberhasilan maksimal 96% dalam penelitian Arash Gholami Rad, dkk yang berjudul "*Vehicle Speed Detection in Video Image Sequence using CVS Method*"[4].

R. Arif Firdaus Lazuardi[9] telah melakukan penelitian untuk menghitung kendaraan menggunakan algoritma *Background Subtraction* dengan metode *Gaussian Mixture Model*. Dalam penelitiannya, perangkat lunak yang dibangun mampu menghitung kendaraan dengan tingkat keberhasilan rata-rata pada waktu pagi hari adalah 94,51% dan siang hari 95,28%. Kemudian penelitian tersebut dilanjutkan oleh Zamroji

Hariyanto[11], dimana penelitian tersebut untuk mengklasifikasi jenis kendaraan. Dalam penelitiannya, perangkat lunak yang dibangun dapat mengklasifikasikan kendaraan dalam 3 jenis kendaraan yaitu motor, mobil, dan truk/bus dengan tingkat akurasi tertinggi sebesar 94,27%. Namun, perangkat lunak yang dibangun belum dapat mendeteksi kecepatan kendaraan.

Tugas akhir ini akan mengembangkan penelitian dari R. Arif dan Zamroji Hariyanto, yaitu mendeteksi kecepatan kendaraan dalam perangkat lunak yang dibangun. Pada subbab selanjutnya akan dijelaskan dasar teori yang digunakan dalam penelitian ini.

2.1 Definisi Citra Digital

Definisi citra menurut Kamus Besar Bahasa Indonesia memiliki makna rupa, gambar, atau gambaran. Sedangkan menurut kamus Webster, citra adalah suatu representasi, kemiripan, atau imitasi dari suatu objek atau benda. Citra terbagi menjadi dua yaitu citra diam dan citra bergerak. Citra diam adalah citra tunggal yang tidak bergerak. Sedangkan, citra bergerak adalah rangkaian citra diam yang ditampilkan secara beruntun sehingga memberi kesan pada mata kita sebagai gambar yang bergerak.

Pada awalnya citra yang dikenal manusia berbentuk citra kontinu. Suatu representasi objek yang dihasilkan dari sistem optik yang menerima sinyal analog dan dinyatakan dalam bidang dua dimensi. Nilai cahaya yang ditransmisikan pada citra kontinu memiliki rentang nilai yang tak terbatas. Contoh dari citra kontinu adalah mata manusia dan kamera analog.

Sebuah citra analog tidak dapat direpresentasikan secara langsung oleh komputer. Oleh sebab itu dilakukan sebuah proses untuk merubah nilai-nilai yang ada pada citra analog agar komputer dapat membaca dan menerjemahkan informasi yang terdapat pada citra analog. Hasil dari pemrosesan tersebut dinamakan sebagai citra digital[9].

Dalam buku *Digital Image Processing* yang ditulis oleh Rafael C. Gonzalez dan Richard E. Woods[5], menjelaskan bahwa citra digital merupakan fungsi dua dimensi yang dapat dinyatakan dengan fungsi $f(x,y)$, dimana x dan y merupakan titik koordinat spasial. Dan amplitudo dari fungsi f pada sembarang koordinat (x,y) merupakan nilai intensitas cahaya, yang merupakan representasi dari warna cahaya yang ada pada citra analog. Citra digital adalah suatu citra dimana (x,y) dan nilai intensitas dari f terbatas (*discrete quantities*), dan telah dilakukan proses digitalisasi spasial dan digitalisasi kuantitas.

2.2 Video Digital

Video adalah teknologi untuk menangkap, merekam, memproses, menyimpan, dan merekonstruksi suatu urutan dari beberapa citra. Alan C. Bovik dalam bukunya *Handbook of Image and Video Processing* menjelaskan bahwa video digital merupakan hasil *sampling* dan kuantisasi dari video analog[6]. Secara mendasar, tidak ada perbedaan proses *sampling* dan kuantisasi antara citra digital dan video digital.

Bagaimanapun juga, video analog yang kita lihat sehari-sehari seperti tampilan pada TV analog, sebenarnya bukan sesuatu yang benar-benar kontinu, melainkan terdiri dari beberapa *frame* yang ditampilkan dengan kecepatan tertentu. Setiap *frame* merupakan citra analog dan kecepatan untuk menampilkan citra-citra yang ada disebut sebagai *framerate* dengan satuan *fps* (*frame per second*). Jika *framerate* cukup tinggi, maka akan terlihat sebagai rangkaian yang kontinu sehingga tercipta ilusi gerak yang halus.

Video analog dapat dinyatakan dengan fungsi $I(x,y,t)$, dimana (x,y) adalah nilai kontinu dari fungsi I dan t menyatakan waktu. Sebenarnya tampilan video analog di TV maupun monitor merupakan representasi dari fungsi sinyal elektrik satu dimensi $V(t)$. Dimana sinyal elektrik satu dimensi tersebut terdiri dari beberapa citra analog $I(x,y,t)$ dengan jumlah citra (x,y) tertentu

dan waktu (t) tertentu. Proses pemisahan video ke beberapa unit *frame* citra disebut sebagai *scanning*.

2.3 Algoritma *Background Subtraction*

Background subtraction (BS) digunakan untuk mendapatkan objek yang bergerak pada serangkaian image. Objek yang bergerak dapat diidentifikasi dengan melakukan pengurangan antara frame pada waktu t dengan *background model* (latar belakang model). Kemudian nilai citra hasil pengurangan tersebut dibandingkan dengan nilai ambang batas (*threshold*) sesuai dengan metode yang dipakai untuk membangun *foreground image*[7].

2.4 *Tracking*

Proses mencari obyek bergerak dalam urutan frame yang dikenal sebagai pelacakan. Pelacakan ini dapat dilakukan dengan menggunakan ekstraksi ciri benda dan mendeteksi obyek/benda bergerak diurutan *frame*. Dengan menggunakan nilai posisi obyek di setiap *frame*, bisa digunakan untuk menghitung posisi dan kecepatan obyek bergerak[8].

2.5 Metode *Gaussian Mixture Model*

Gaussian Mixture Model (GMM) merupakan metode yang tepat untuk berbagai kondisi yang terdapat pada citra, seperti *background* citra yang selalu statis, multimodal, maupun yang mengandung *noise* (gangguan atau objek yang tidak diinginkan terdapat pada citra).

Pada proses GMM dibutuhkan algoritma *clustering* untuk mengelompokkan piksel-piksel mana saja yang termasuk *foreground* atau *background*. Pada umumnya metode *clustering* yang digunakan adalah *Expectation Maximization* dan K-means. Chris Stauffer dalam penelitiannya menggunakan algoritma *clustering* K-means dalam mencari dan meng-update model *background* karena metode tersebut memiliki kecepatan proses yang lebih cepat dan hasil yang cukup baik[10].

GMM merupakan tipe *density model* yang terdiri dari komponen fungsi-fungsi Gaussian. Komponen fungsi tersebut terdiri dari *weight* yang berbeda untuk menghasilkan *multi model density*. Model-model GMM terbentuk dari data warna piksel berdasarkan waktu. Hasil model tersebut akan menjadi 2 bagian, model yang mencerminkan *background* dan model *non-background*. Jumlah model GMM yang digunakan mempengaruhi jumlah model *background*. Semakin besar jumlah model GMM yang dipakai semakin banyak model *background* yang dimiliki suatu piksel.

GMM memproses tiap piksel pada citra, baik citra tersebut berupa skalar (citra *grayscale*) maupun vektor (citra berwarna/RGB). Untuk sebarang waktu t , citra tersebut dimodelkan sebagai berikut[9]:

$$\{X_1, \dots, X_t\} = \{I(x_0, y_0, i) : 1 \leq i \leq t\} \quad (2.1)$$

dengan X_t adalah citra skalar maupun vektor, dan I adalah serangkaian citra.

Untuk setiap piksel, $\{X_1, \dots, X_t\}$, dimodelkan dengan *mixture K* distribusi Gaussian. Adapun untuk mengambil nilai probabilitas di setiap pikselnya didapatkan melalui persamaan ini:

$$P(X_t) = \sum_{i=1}^K \omega_{i,t} * \eta(X_t | \mu_{i,t}, \Sigma_{i,t}) \quad (2.2)$$

dengan K adalah jumlah distribusi, $\omega_{i,t}$ adalah estimasi *weight* ke- i *mixture* Gaussian pada waktu t , $\mu_{i,t}$ adalah nilai mean ke- i *mixture* Gaussian pada waktu t , $\Sigma_{i,t}$ adalah *covariance* matriks ke- i *mixture* Gaussian pada waktu t , dan η adalah *Probability Density Function* Gaussian,

$$\eta(X_t | \mu_{i,t}, \Sigma_{i,t}) = \frac{1}{(2\pi)^{\frac{n}{2}} |\Sigma_{i,t}|^{\frac{1}{2}}} e^{-\frac{1}{2}(X_t - \mu_{i,t})^T \Sigma_{i,t}^{-1} (X_t - \mu_{i,t})} \quad (2.3)$$

dengan $|\Sigma_{i,t}|$ adalah determinan dari *covariance*, pangkat T adalah transpose matriks, pangkat -1 adalah invers matriks, e adalah eksponensial, π adalah phi, dan n adalah ukuran citra skalar X , maupun citra vektor X (RGB). K ditentukan oleh

kemampuan memori dan kekuatan *hardware* yang dipakai, antara 3 hingga 5.

Dan *covariance* matriks didapatkan dari;

$$\Sigma_{i,t} = \sigma_i^2 I \quad (2.4)$$

dengan σ_k^2 adalah varian dari Gaussian ke- i dan I adalah matriks identitas.

Secara singkat, terdapat dua proses pada GMM; yaitu tahap pencocokan *input* terhadap distribusi dan tahap pemilihan distribusi yang mencerminkan *background*.

2.5.1 Tahap Pencocokan *Input* Terhadap Distribusi.

Pada tahap ini akan dilakukan *update* parameter GMM yang akan digunakan untuk memproses *input* selanjutnya.

Berikut model matematis GMM dari tahap pencocokan *input* hingga *update* parameter[9]:

- a. Suatu piksel masuk dalam distribusi jika nilai piksel masuk dalam jarak 2.5 standar deviasi dari sebuah distribusi

$$\mu_i - 2.5 * \sigma_i < X_t < \mu_i + 2.5 * \sigma_i \quad (2.5)$$

μ_i adalah vektor nilai *mean* citra RGB dari Gaussian ke- i , σ_i adalah standar deviasi dari Gaussian ke- i , dan X_t adalah vektor dari citra RGB

- b. Dari persamaan (2.1), komponen GMM yang akan di *update* nilainya adalah $\omega_{i,t}$ (*weight*), $\mu_{i,t}$ (*mean*), dan $\sigma^2_{i,t}$ (*varian*). *Update weight* dilakukan setiap saat:

$$\omega_{i,t} = (1 - \alpha)\omega_{i,t-1} + \alpha(M_{i,t}) \quad (2.6)$$

α adalah *learning rate* dan $M_{i,t}$ bernilai 1 untuk model yang cocok dan 0 untuk lainnya.

Update mean dilakukan jika dan hanya jika ada model yang cocok:

$$\mu_{i,t} = (1 - \rho)\mu_{i,t-1} + \rho X_t \quad (2.7)$$

Update variance dilakukan jika dan hanya jika ada model yang cocok:

$$\sigma^2_{i,t} = (1 - \rho)\sigma^2_{i,t-1} + \rho(X_t - \mu_{i,t})^T (X_t - \mu_{i,t}) \quad (2.8)$$

dengan,

$$\rho = \alpha \eta (X_t | \mu_{i,t}, \Sigma_{i,t}) \quad (2.9)$$

2.5.2 Tahap Pemilihan Distribusi yang Mencerminkan *Background*.

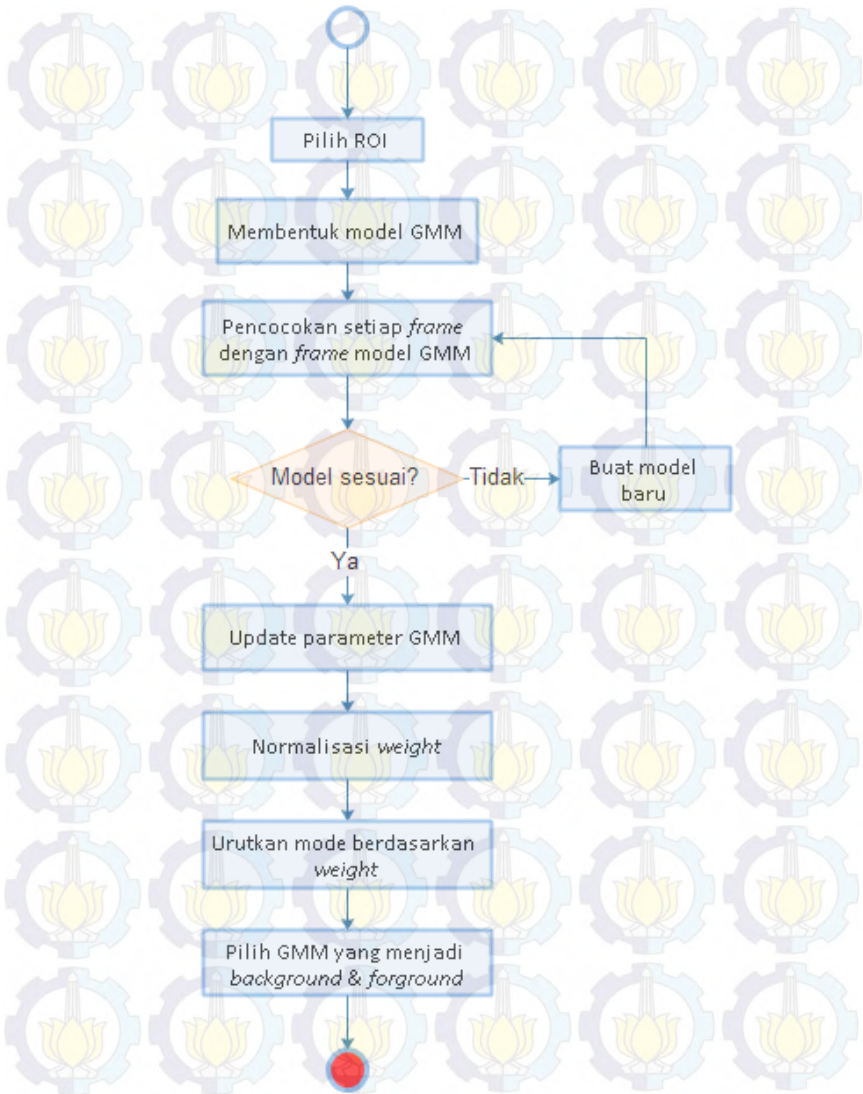
Pada tahap ini akan dipilih model yang menyerupai *background*. Model akan diurutkan berdasarkan ω/σ^2 sehingga distribusi yang menyerupai *background* akan berada di urutan atas sedangkan yang tidak menyerupai akan berada di urutan bawah yang nantinya digantikan oleh distribusi lain yang cocok. Model matematis untuk memilih B distribusi pertama yang dijadikan *background* adalah;

$$B = \arg \min_b \left(\sum_{k=1}^b \omega_k > \tau \right) \quad (2.10)$$

dengan nilai default τ adalah 0.4[9].

R. Arif[9] telah melakukan Uji coba terhadap parameter-parameter Gaussian menggunakan video rekaman yang berdurasi 30 detik. Sehingga diperoleh kesimpulan pada penelitiannya yaitu dengan menggunakan jumlah model antara 3 s.d 7 dan *learning rate* 0.001, 0.005, atau 0.01 dapat diperoleh model menyerupai *background* sebenarnya[9].

Berikut disajikan diagram alir proses GMM[9]:



Gambar 2.1. Diagram alir proses GMM.

2.6 Metode Analisis *Blob*

Analisis *blob* ini menggunakan metode *connected component*, dimana di setiap kumpulan piksel yang tingkat keabuannya bernilai satu, dikategorikan sebagai satu objek. Setiap objek yang terdeteksi akan diberi label untuk mempermudah pengolahan. Pada Gambar 2.2 menjelaskan tentang pemilihan area *blob* yang terdeteksi. Dari analisis *blob* tersebut akan diperoleh informasi tentang *centroid*, luas area, tinggi, dan lebar sebuah objek dari bentuk *rectangle*.

0	0	0	0	0	0	0	0	0	0
0	0	1	1	1	0	0	0	0	0
0	1	0	0	1	0	0	0	0	0
0	1	1	1	1	1	0	0	0	0
0	1	1	0	1	1	0	0	0	0
0	0	1	1	1	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	1	1	1	1
0	0	0	0	0	0	1	0	1	0
0	0	0	0	0	0	1	1	1	0

a

0	0	0	0	0	0	0	0	0	0
0	0	1	1	1	0	0	0	0	0
0	1	0	0	1	0	0	0	0	0
0	0	1	1	1	1	0	0	0	0
0	1	1	0	1	1	0	0	0	0
0	0	1	1	1	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	1	1	1	1
0	0	0	0	0	0	1	0	1	0
0	0	0	0	0	0	1	1	1	0

b

Gambar 2.2. Bagian a adalah representasi citra ukuran 10x10 piksel dan citra dengan nilai piksel 1 merupakan representasi dari objek (*foreground*). Sedangkan bagian b merupakan *blob* yang terdeteksi.

(sumber: “Penghitungan Kendaraan Bergerak Berbasis Algoritma Background Subtraction Menggunakan Metode Gaussian Mixture Model” [9])

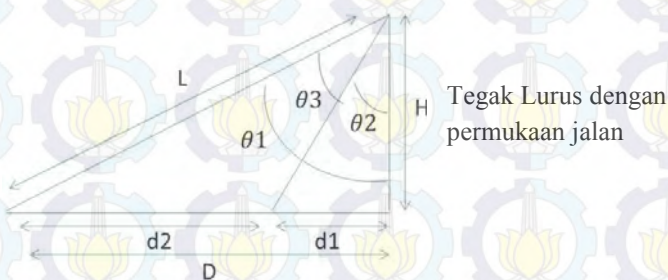
2.7 Formula untuk Perhitungan Kecepatan Kendaraan dengan Posisi Kamera di Atas

Untuk pendeteksian kecepatan kendaraan dengan posisi kamera di atas, sebelumnya akan dilakukan beberapa langkah saat melakukan pengambilan video. Beberapa langkah diantaranya yaitu :

2.7.1 Pengujian Kamera

Pengujian kamera adalah salah satu aspek yang penting dari penelitian. Pada bagian ini, dilakukan beberapa hal sebelum melakukan penelitian, diantaranya yaitu perhitungan fungsi pola antara kendaraan koordinat pada video dan kondisi nyata, menentukan posisi kamera video yang akan dipasang untuk pengambilan video lalu lintas jalan, serta menentukan karakteristik yang terlibat di dalamnya. Kamera diatur pada ketinggian H dengan posisi tegak lurus di atas permukaan jalan dengan sumbu kemiringan yaitu θ_1 [4].

2.7.2 Perhitungan Sudut dan Bidang Tegak Lurus Pandang



Gambar 2.3. Setting Kamera dengan posisi di atas.

Berdasarkan Gambar 2.3, jika diasumsikan tidak ada kendaraan yang lewat, maka :

$$\theta_1 = \arctan\left(\frac{D}{H}\right) \quad (2.11)$$

dan sudut untuk daerah yang tidak terlihat oleh kamera adalah :

$$\theta_2 = \arctan\left(\frac{d_1}{H}\right) \quad (2.12)$$

dengan :

θ_1 adalah sudut pemasangan kamera;

θ_2 adalah sudut daerah yang tidak terlihat oleh kamera;

θ_3 adalah sudut pandang yang dicakup oleh kamera;

H adalah tinggi kamera dari permukaan jalan;

D adalah jarak horizontal antara kamera dan kendaraan;

L adalah jarak nyata antara kamera dan kendaraan;

d_1 adalah daerah yang tidak terlihat oleh kamera;

d_2 adalah daerah yang terlihat oleh kamera;

Selain itu juga, dapat diketahui bahwa :

$$\theta_3 = \theta_1 - \theta_2 \rightarrow \theta_2 = \theta_3 - \theta_1 \quad (2.13)$$

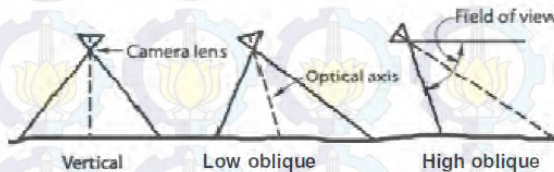
Daerah d_1 bisa diperoleh dengan :

$$d_1 = H \tan(\theta_1 - \theta_3) = H \tan \theta_2 \quad (2.14)$$

Dan untuk daerah d_2 bisa diperoleh dengan :

$$d_2 = D - d_1 \quad (2.15)$$

Kecenderungan dari kamera dengan posisi seperti Gambar 2.3 akan menghasilkan gambar dari tiga jenis area seperti yang ditunjukkan pada Gambar 2.4. Sedangkan Gambar 2.5 merupakan *grid* bagian garis yang terlihat pada berbagai jenis sudut kamera[4].



Gambar 2.4. Kemiringan Kamera hasil rekaman dari atas[4].



Gambar 2.5. Tampilan *grid* hasil rekaman dari atas[4].

2.8 Speed Detection

Kecepatan kendaraan diperoleh dari *frame* hasil deteksi *foreground*, yaitu dengan menentukan posisi kendaraan pada setiap *frame*. Jadi perlu ditentukan *boundingbox* dan *centroid* dari hasil yang telah dilakukan sebelumnya. *Centroid* disini sangat penting untuk mengetahui jarak kendaraan yang bergerak dalam *frame* yang berurutan.

Untuk mengetahui jarak yang ditempuh dalam *pixel*, dapat dihitung dengan mencari koordinat seperti berikut:

Titik koordinat objek saat berada pada *frame* $i = (x_i, y_i)$,

Titik koordinat objek saat berada pada *frame* $i + 1 = (x_{i+1}, y_{i+1})$ [4].

Jarak perpindahan dari kendaraan yaitu:

$$d_i = \sqrt{(x_i - x_{i+1})^2 + (y_i - y_{i+1})^2} \quad (2.16)$$

dan jika *framerate* dari video adalah 25 *frame* per detik, maka waktu antara dua *frame* berturut-turut yaitu 0,04s[4].

Dan untuk menghitung kecepatan dapat menggunakan persamaan berikut :

$$V = \frac{\Delta x}{\Delta t} \quad (2.17)$$

dimana Δx adalah jarak perpindahan dan Δt adalah waktu[4].

$$\Delta x = \sum_{i=1}^n d_i \quad (2.18)$$

dimana n adalah banyaknya *frame*.

BAB III METODOLOGI

Tahapan pengerjaan dalam menyelesaikan Tugas Akhir ini adalah sebagai berikut:

3.1 Studi Literatur dan Perekaman Video

Pada tahap pertama ini akan dilakukan pengkajian tentang algoritma pendeteksian kecepatan kendaraan. Pada tahap ini juga akan dilakukan perekaman video kendaraan yang bergerak di jalan. Studi ini dilakukan dengan membaca buku, jurnal, ataupun artikel yang terkait serta melakukan pengambilan video di beberapa tempat, diantaranya yaitu di Halaman Sekolah, di Lapangan, di Jalan Raya dan di Jalan Tol.

Video diambil menggunakan kamera digital, yang dimana dalam pengambilan video tersebut dilakukan beberapa pengukuran parameter diantaranya yaitu besar sudut pemasangan kamera, tinggi pemasangan kamera dan jarak horizontal kamera ke daerah titik awal yang tercakup oleh kamera. Rekaman video diambil berdasarkan ketentuan berikut:

- a. Sudut pandang yang diambil ketika merekam adalah dari atas dengan jangkauan pandangan seluruh jalan arus satu arah.

3.2 Analisis dan Desain Sistem

Pada tahap kedua ini akan dilakukan analisis video dan penentuan parameter-parameter apa yang akan dibutuhkan dalam pembuatan program. Kemudian dibuat desain sistem dari program sesuai dengan hasil analisis.

Berikut beberapa tahapan untuk mengolah rekaman video *off-line* arus kendaraan:

- a. Sistem ini memiliki *inputan* berupa video digital. Pemilihan area ROI (*Region of Interest*) pada rekaman video yang digunakan sebagai area pendeteksian.

- b. *Background Subtraction* menggunakan metode GMM untuk segmentasi setiap *frame* menjadi citra *background* dan *foreground*.
- c. Penghilangan bayangan dan *filtering* agar citra bebas dari *noise* sehingga dapat diolah dengan mudah.
- d. Analisis *blob* pada citra *foreground* untuk mendeteksi objek pada citra. Selanjutnya proses *tracking* untuk menentukan apakah objek pada *frame* ke-*i* dengan *frame* ke- *i-1* merupakan objek yang sama.
- e. Objek-objek itu akan dicari *centroid*-nya yang nantinya akan digunakan untuk menghitung jarak daerah ROI, kemudian dilanjutkan dengan mencari waktu, yang selanjutnya jarak dan waktu yang didapat tersebut digunakan untuk mendeteksi kecepatan kendaraan.

Untuk mengimplementasikan beberapa kebutuhan yang telah dicatat pada tahap analisis. Perangkat lunak pemrograman yang dipakai untuk membuat program pendeteksi kecepatan kendaraan bergerak adalah MATLAB.

3.3 Implementasi Sistem

Pada tahap ketiga akan dilanjutkan dengan implementasi sistem dalam bentuk perangkat lunak sesuai dengan hasil analisis dan desain sistem. Akan dibuat desain *interface* yang menarik dan *user friendly* untuk memudahkan dan membuat nyaman pengguna.

3.4 Uji Coba dan Pembahasan

Pada tahap keempat dilakukan pengujian terhadap perangkat lunak yang telah selesai dibuat menggunakan *input* video rekaman kendaraan bergerak di jalan. Kemudian dilakukan analisis pembahasan sehingga dapat dicatat beberapa hal yang dijadikan pertimbangan dalam menarik kesimpulan. Pada tahap ini juga dilakukan evaluasi terhadap program yang telah dibangun. Hasil evaluasi dicatat untuk membenahi hal-hal yang masih kurang. Kemudian dilanjutkan dengan uji coba deteksi

kecepatan kendaraan, data hasil deteksi kecepatan dari program akan dibandingkan dengan kecepatan data sebenarnya. Kemudian dihitung tingkat akurasi dari program menggunakan persamaan (3.1). Selain itu akan dilakukan evaluasi sehingga dapat membenahi hal-hal yang masih kurang dan mengatasi error yang ditemukan.

$$PK = \frac{KS - abs(KS - KP)}{KS} \times 100 \quad (3.1)$$

dengan PK adalah prosentase keberhasilan, KP adalah kecepatan kendaraan yang terdeteksi program, dan KS adalah kecepatan kendaraan sebenarnya.

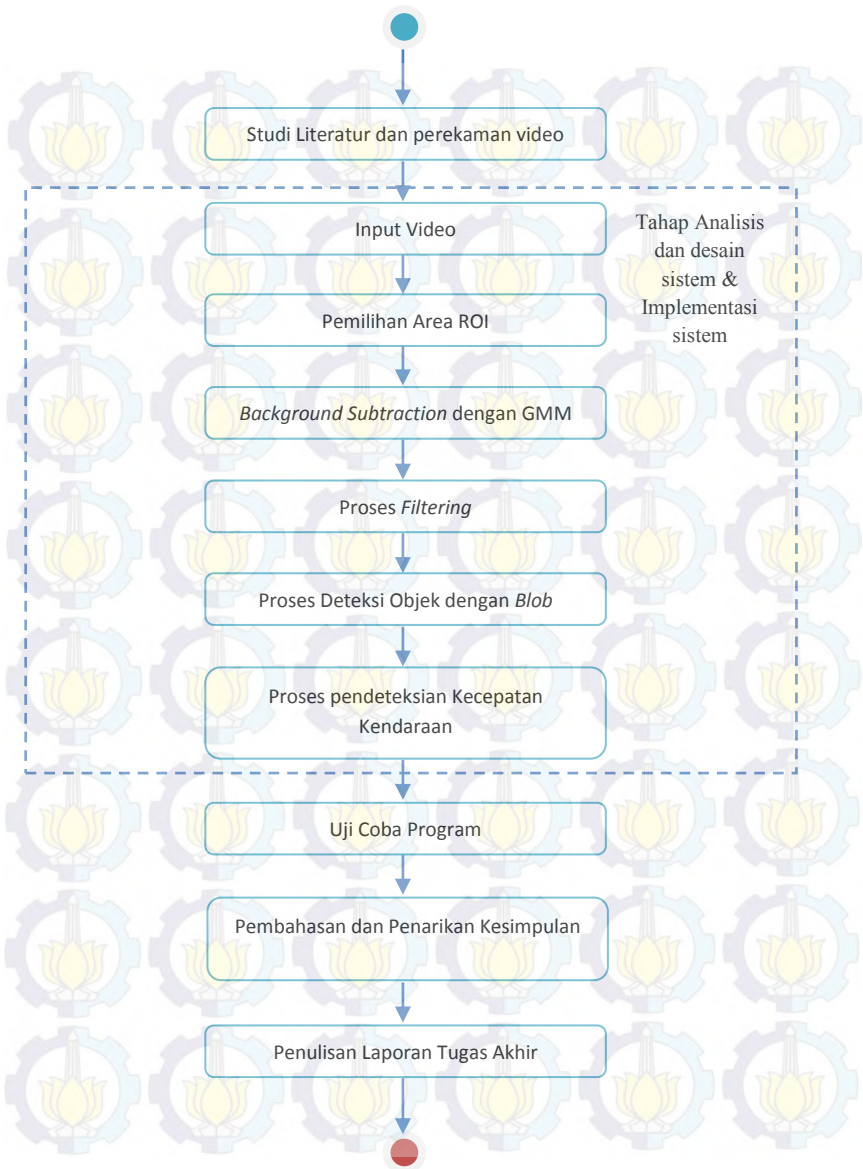
3.5 Penarikan Kesimpulan

Tahap kelima merupakan tahap terakhir dari Tugas Akhir yang diusulkan, yang berisi penarikan kesimpulan dari hasil pengerjaan dan memberikan saran untuk pengembangan berikutnya.

3.6 Penulisan Laporan Tugas Akhir

Pada tahap keenam ini penulis menuliskan semua hasil yang telah didapatkan selama pengerjaan Tugas Akhir.

Adapun diagram dari tahap penelitian ini dapat dilihat pada Gambar 3.1.



Gambar 3.1. Diagram Metodologi Penelitian

BAB IV

PERANCANGAN DAN IMPLEMENTASI SISTEM

Pada bab ini akan dibahas mengenai perancangan dan implementasi sistem dimulai dari pembahasan proses pengambilan data masukan, pengolahan data masukan dengan algoritma *Background Subtraction* dengan metode *Gaussian Mixture Model* serta metode analisis *blob*, serta penjelasan mengenai cara untuk mendapatkan data keluaran yang sesuai dengan tujuan dari penelitian Tugas Akhir ini.

Selain hal yang telah disebutkan di atas, perancangan sistem juga meliputi perancangan antar muka (*user interface*) untuk memudahkan penilitan dalam memproses data dan melakukan analisis terhadap keluaran dari sistem yang telah dibangun.

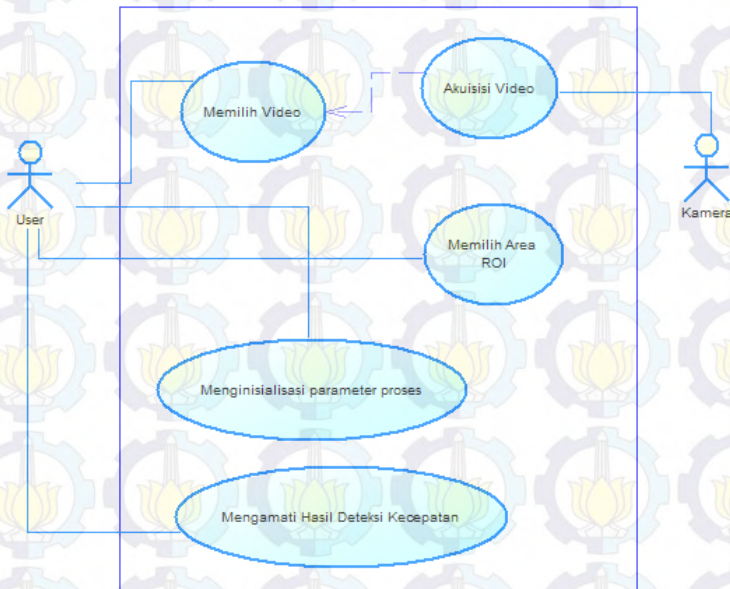
Hasil dari analisis dan perancangan sistem dilanjutkan dengan implementasi sistem. Sehingga perangkat lunak yang dibutuhkan untuk memproses informasi dari rekaman video digital dapat dilakukan.

4.1 Analisis Sistem

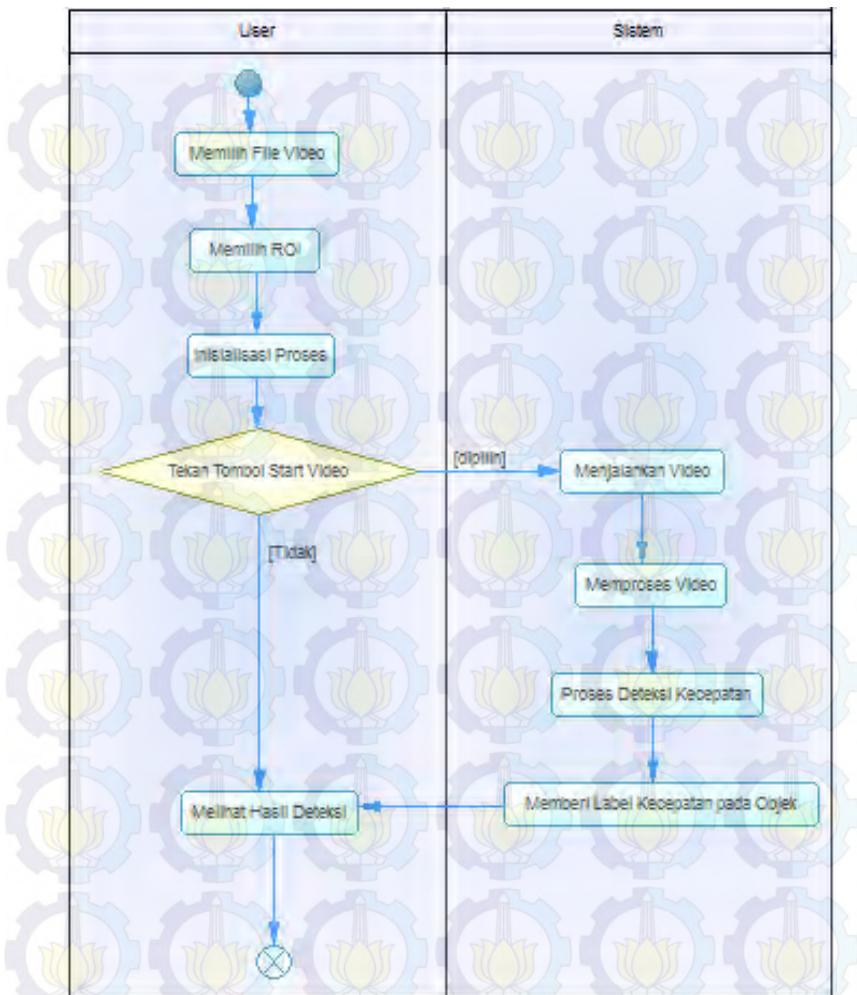
Untuk membangun perangkat lunak yang dapat mendeteksi kecepatan kendaraan bergerak berbasis video digital. Maka diperlukan sistem yang dapat mengolah data masukan berupa rekaman kendaraan bergerak, pada kasus Tugas Akhir ini menggunakan data masukan berupa video digital arus kendaraan secara *offline*. Dengan mengkombinasikan algoritma *Background Subtraction* dengan metode *Gaussisan Mixture Model*[9] untuk mendapatkan objek kendaraan dalam video dan metode analisis *blob* untuk mendapatakan *centroid* yang nantinya akan digunakan dalam mendeteksi kecepatan kendaraan maka akan tercipta sistem yang sesuai dengan tujuan dari Penelitian Tugas Akhir ini.

4.1.1 Analisis Sistem Perangkat Lunak

Perangkat lunak yang akan dibangun dapat digunakan oleh dinas perhubungan untuk mengamati kondisi arus lalu lintas di suatu jalan sehingga dapat menganalisa kebutuhan infrastruktur jalan tersebut. Selain itu masyarakat juga dapat memperoleh informasi kondisi lalu lintas pada suatu jalan. *Use Case Diagram* dari perangkat lunak ini disajikan pada Gambar 4.1 dan *Activity Diagram* pada Gambar 4.2.



Gambar 4.1. *Use Case Diagram* Perangkat Lunak Deteksi Kecepatan Kendaraan

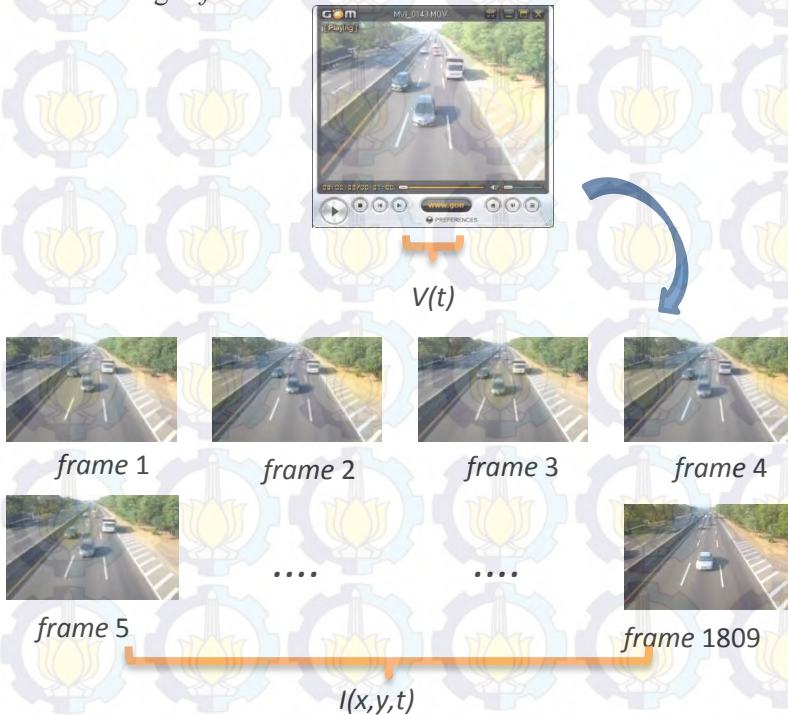


Gambar 4.2. Activity Diagram Perangkat Lunak Deteksi Kecepatan Kendaraan.

Sistem perangkat lunak yang dibangun ini memiliki beberapa tahapan sebagai berikut :

a. Akuisisi video

Data masukan berupa rekaman video digital *offline* arus kendaraan yang kemudian di-*scanning*. *Scanning* adalah proses pemecahan video menjadi beberapa rangkaian citra yang sering disebut dengan *frame*.



Gambar 4.3. Proses *Scanning* Video.

Pengambilan video kendaraan bergerak di Jalan. Video diambil menggunakan kamera digital Nikon 16.1 Mega Piksel. Video yang diambil berdasarkan ketentuan berikut:

- Sudut pandang yang diambil ketika merekam adalah dari atas dengan jangkauan pandangan seluruh jalan satu arah.

b. Penentuan ROI (*Region of Interest*)

ROI (*Region of Interest*) akan digunakan sebagai area pendeteksian kecepatan kendaraan bergerak.



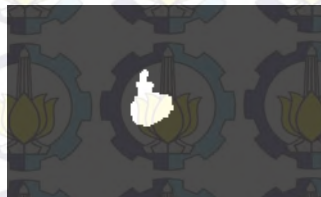
Gambar 4.4. Citra dengan ROI (daerah didalam kotak merah).

c. Segmentasi citra

Kemudian sistem membagi *frame* menjadi citra *background* dan citra *foreground* dengan menggunakan algoritma *Background Subtraction* dengan metode *Gaussian Mixture Model*[9]. Citra *background* adalah citra yang menjadi latar belakang dari suatu objek yang bergerak. Sedangkan *foreground* adalah gambaran objek bergerak yang terdeteksi.



(a)



(b)

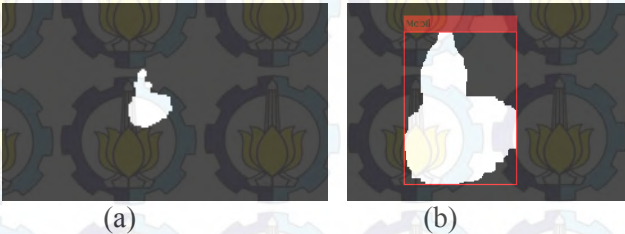
Gambar 4.5. Citra hasil *Background Subtraction* dengan metode *Gaussian Mixture Model*. (a) citra asli, (b) citra *foreground* Proses filtering

Citra *foreground* yang diperoleh di-*filter* untuk menghilangkan beberapa *noise* yang ada pada citra. Sehingga citra akan lebih mudah untuk diolah dan objek kendaraan akan tampak lebih jelas. *Noise* ini merupakan citra yang tertangkap akibat pengaruh cahaya, hembusan angin, bayangan yang muncul

dari objek, serta gangguan-gangguan lain yang tertangkap oleh kamera.

d. Deteksi objek kendaraan

Pendeteksian objek pada Citra hasil *filtering* dengan analisis *blob* untuk memberi label pada setiap objek yang terdeteksi. Analisis *blob* ini menggunakan metode *connected component*, dimana di setiap kumpulan objek yang tingkat keabuan pikselnya bernilai satu, dikategorikan sebagai satu objek. Dari analisis *blob* tersebut akan diperoleh informasi tentang *centroid*, luas area, tinggi, dan lebar sebuah objek dari bentuk *rectangle*.



Gambar 4.6. Citra Hasil Deteksi *Blob*, bagian (a) *foreground* dan (b) merupakan *blob* yang terdeteksi.

e. *Tracking* objek kendaraan

Langkah berikutnya adalah *tracking* objek kendaraan dengan *tracking blob*. *Tracking blob* bertujuan untuk menentukan bahwa objek yang terdeteksi pada *frame* ke- i dengan *frame* ke- $i-1$ adalah objek yang sama atau berbeda. Hal tersebut dapat diketahui dengan mengukur jarak dari *centroid* masing-masing *blob* yang terdeteksi. Jika jaraknya kurang dari *threshold* yang telah ditentukan maka objek yang terdeteksi pada *frame-frame* tersebut adalah objek yang sama.

f. Proses deteksi kecepatan kendaraan

Objek hasil *tracking* tersebut akan dideteksi kecepatannya. Setiap objek hasil *blob tracking* akan dihitung nilai *centroidnya* pada *frame* ke- i dan *frame* ke- $i-1$.

Pada Gambar 4.7 ditunjukkan hasil deteksi kecepatan kendaraan.



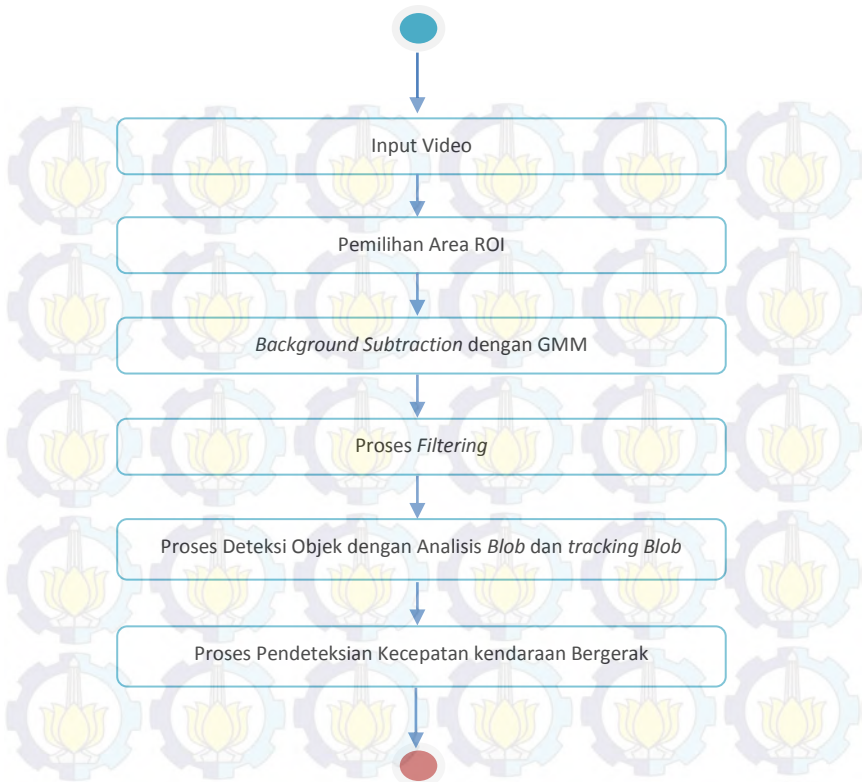
Gambar 4.7. Citra Hasil Deteksi, bagian (a) citra objek hasil *blob tracking* dan (b) hasil deteksi kecepatannya

4.1.2 Analisis Kebutuhan Sistem

Perangkat lunak ini dibangun menggunakan bahasa pemrograman MATLAB R2013a.

Selain itu juga diperlukan kamera digital untuk merekam arus kendaraan di jalan. Program ini dikembangkan menggunakan laptop dengan spesifikasi Intel Core i3-2310M CPU @2.10 GHz 4,0GB RAM, Intel HD Graphics 3000.

Gambar 4.8 menjelaskan tentang gambaran pokok *business rule* dari perangkat lunak yang dibangun.



Gambar 4.8. *Busniess Rule* Sistem Perangkat Lunak.

4.2 Perancangan Sistem

Setelah analisis sistem, kemudian dilanjutkan dengan perancangan sistem. Perancangan sistem tersebut meliputi perancangan data sistem, perancangan proses, dan perancangan antar muka sistem.

4.2.1 Perancangan Data Sistem

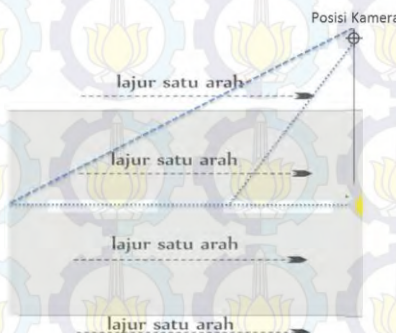
Terdapat tiga macam data yang digunakan oleh sistem ini antara lain data masukan, data proses, dan data keluaran. Pada Tugas Akhir ini data masukan berupa video rekaman *offline* di

beberapa jalan diambil secara mandiri oleh peneliti. Data proses merupakan data yang berisi parameter-parameter yang akan digunakan oleh algoritma *Background Subtraction* dan metode *Gaussian Mixture Model* serta metode analisis *blob* dan *speed detection*. Sedangkan data keluaran adalah informasi mengenai kecepatan kendaraan bergerak yang terdeteksi oleh sistem.

4.2.1.1 Data Masukan

Data masukan sistem ini berupa video *offline* rekaman kendaraan di jalan yang diambil menggunakan kamera digital. Video diambil dengan posisi lensa kamera menjangkau ke seluruh area jalan satu lajur. Dalam perekaman video tersebut dilakukan beberapa pengukuran parameter diantaranya yaitu : besar sudut pemasangan kamera, tinggi pemasangan kamera dan jarak horizontal kamera ke daerah titik awal yang tercakup oleh kamera. *Layout* untuk pengambilan video rekaman dapat dilihat pada Gambar 4.9. File video yang diambil memiliki spesifikasi sesuai dengan batasan masalah yang terdapat dalam penelitian Tugas Akhir ini yaitu:

- a. Format ekstensi rekaman video *offline* dapat berupa: *.avi.



Gambar 4.9. *Layout* Untuk Pengambilan Video.

4.2.1.2 Data Proses

Data proses merupakan data yang digunakan dalam proses pengolahan data masukan. Data proses ini diperoleh dari hasil pengolahan data masukan sesuai dengan tahapan algoritma dan metode yang telah disusun. Tabel 4.1 menjelaskan tahapan dari data proses.

Tabel 4.1. Tabel data proses

No	Tahapan	Input	Output
1.	<i>Input</i> Awal	Video	<i>Frame</i> Citra
2.	Pilih Area ROI	<i>Frame</i> Citra	Citra ROI
3.	Hitung Parameter yang dibutuhkan	Parameter	Parameter
4.	Proses GMM	Citra ROI	Citra <i>Foreground</i>
5.	Filtering	Citra <i>Foreground</i>	Citra <i>Foreground</i> yang telah difilter
6.	Deteksi <i>Blob</i>	Citra <i>Foreground</i> yang telah difilter	Area <i>blob</i> (<i>centroid</i> , <i>weight</i> , <i>height</i>)
7.	Hitung koordinat kendaraan pada saat melewati daerah ROI	Objek hasil deteksi <i>blob</i>	Koordinat x dan y
8.	Hitung jarak area yang dilewati kendaraan	Objek hasil deteksi <i>blob</i>	Jarak
9.	Perhitungan waktu yang dibutuhkan	Objek hasil deteksi <i>blob</i>	Waktu
10.	Hitung Kecepatan Kendaraan	Jarak dan Waktu	Kecepatan kendaraan

4.2.1.3 Data Keluaran

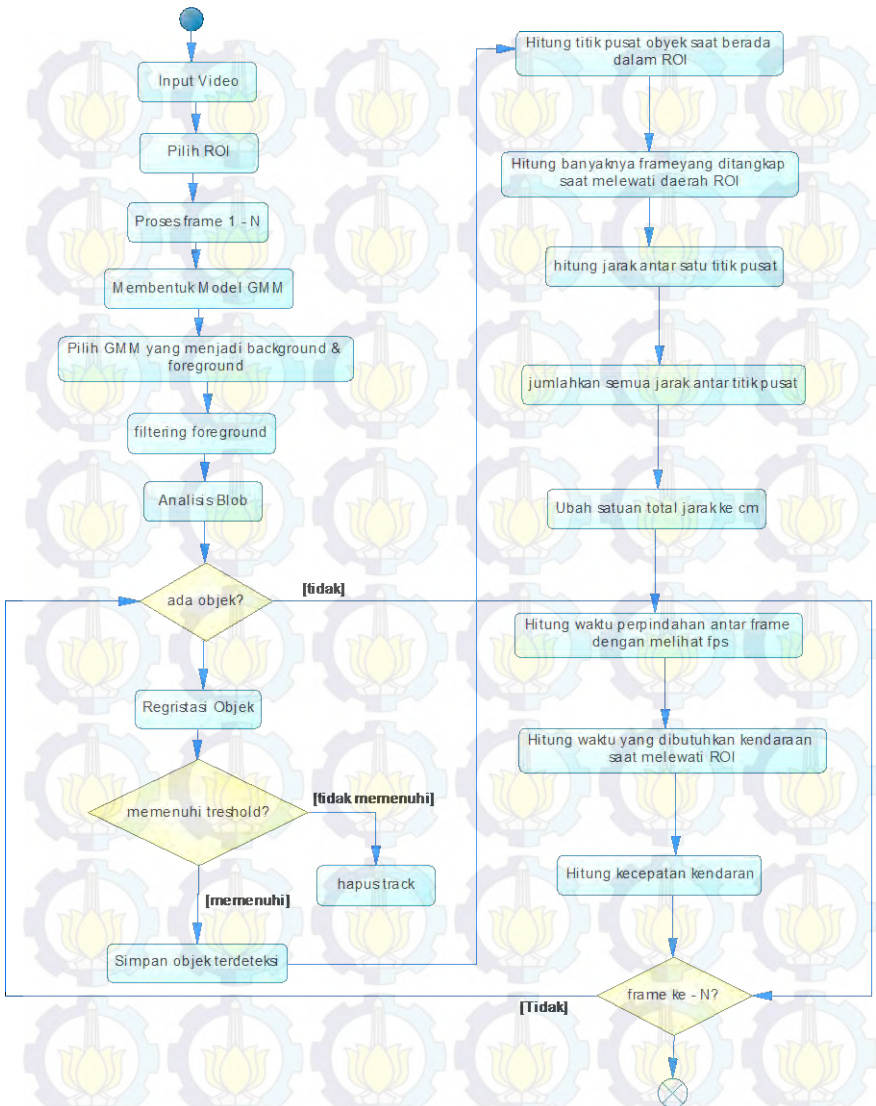
Data keluaran berupa hasil deteksi kecepatan kendaraan bergerak. Program ini juga menghasilkan keluaran berupa citra ROI, dan citra *foreground*.

4.2.2 Perancangan Proses

4.2.2.1 Perancangan Proses Algoritma

Proses algoritma ini dimulai dengan pemilihan ROI (*Region of Interest*) yang digunakan sebagai area pendeteksian kecepatan kendaraan bergerak. Selanjutnya citra ROI tersebut diproses menggunakan algoritma *Background Subtraction* dengan metode *Gaussian Mixture Model* untuk mengelompokkan bagian piksel yang tergolong citra *foreground* atau *background*[9]. Proses berikutnya adalah *filtering* citra *foreground* untuk menghilangkan *noise*. Citra *foreground* yang telah difilter akan dianalisis *blob* untuk mendeteksi objek yang ada pada citra tersebut. Pada tahap akhir objek-objek yang terdeteksi dideteksi kekecepatannya.

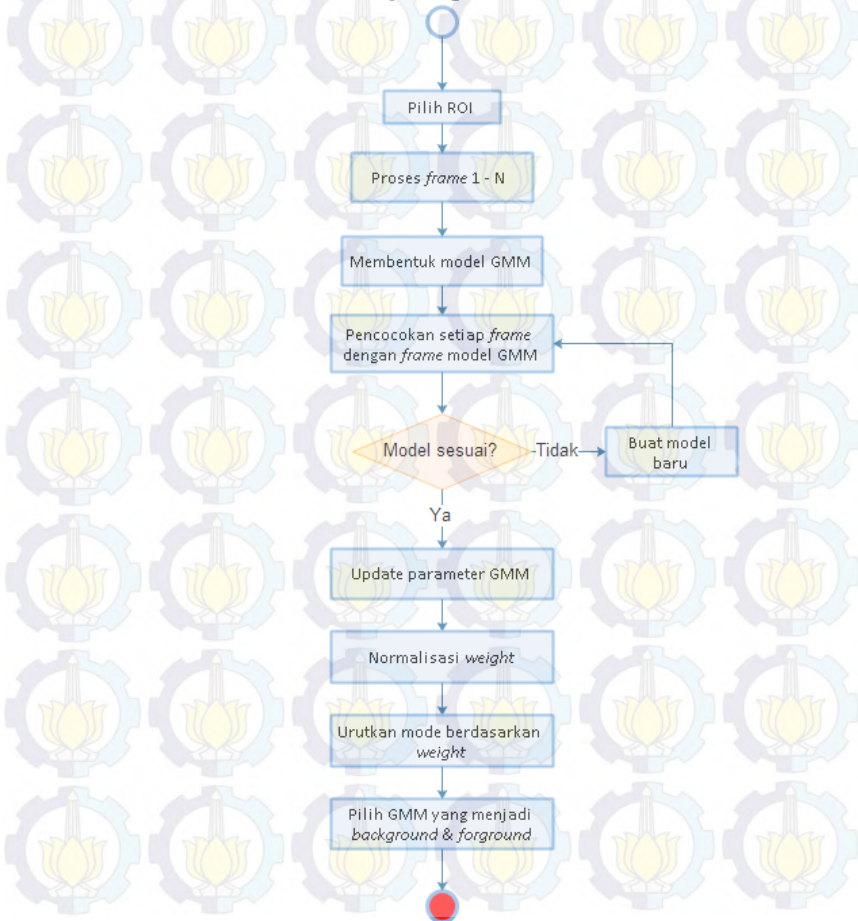
Gambar 4.10 menjelaskan diagram alir sistem perangkat lunak.



Gambar 4.10. Diagram Alir Sistem Perangkat Lunak.

4.2.2.2 Perancangan Proses *Gaussian Mixture Model*

Berikut adalah alur proses untuk mengolah citra per *frame* menggunakan metode GMM. Berdasarkan hasil uji coba *input* parameter GMM pada penelitian R. Arif [9], nilai komponen Gaussian yang baik digunakan adalah 3,4 dan 5, sedangkan nilai *learning rate* adalah 0.001,0.005, dan 0.01. Berikut adalah blok diagram proses GMM.



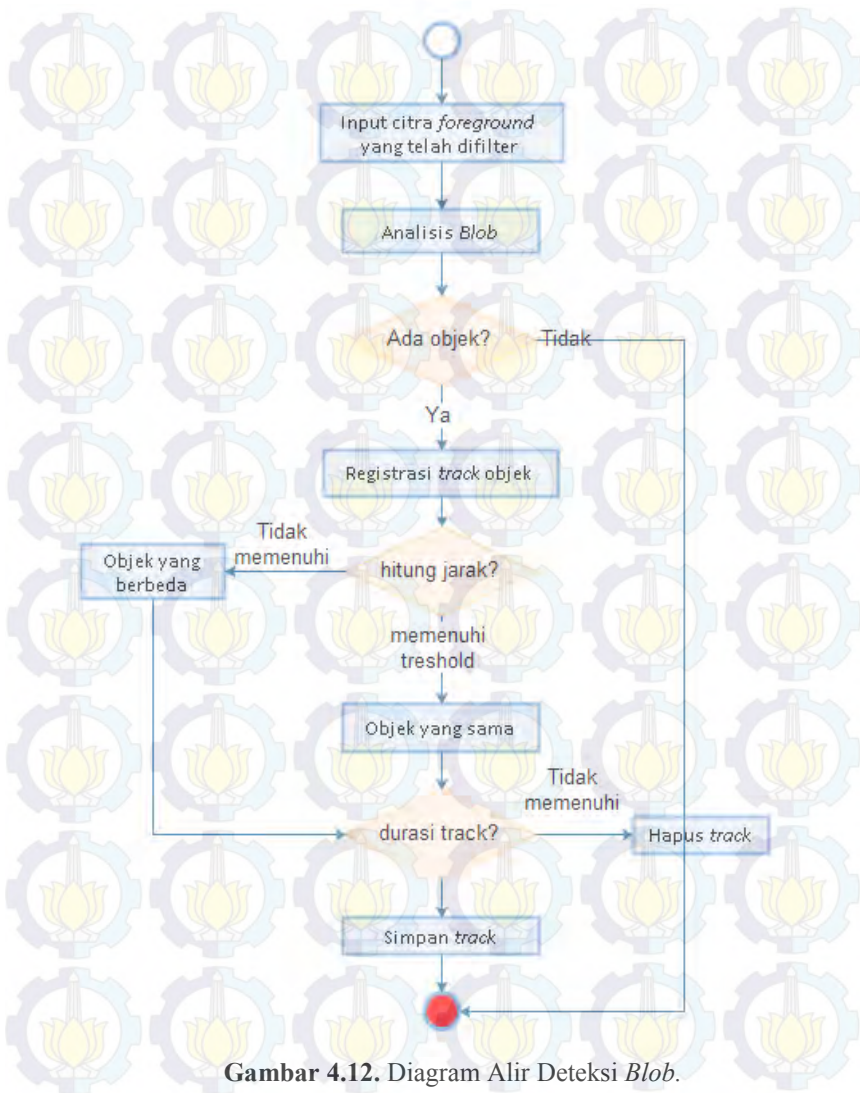
Gambar 4.11. Diagram Alir GMM.

4.2.2.3 Perancangan Proses Deteksi *Blob*

Deteksi *blob* digunakan untuk menentukan apakah objek yang terdeteksi pada *frame* ke- $i-1$ sama atau merupakan objek yang berbeda dari objek yang terdeteksi pada *frame* ke- i . Analisis *blob* menghasilkan nilai *centroid*, *weight*, dan *height* di setiap objek yang terdeteksi. Dari informasi nilai tersebut, program dapat menghitung jarak antara objek pada *frame* ke- $i-1$ dan objek pada *frame* ke- i . Apabila jarak yang diketahui dibawah *threshold* yang telah ditentukan, maka objek tersebut termasuk objek yang sama. Jika tidak maka objek tersebut dikategorikan sebagai objek yang lain. Dengan analisis *blob* inilah program dapat menghitung jumlah objek atau kendaraan yang bergerak[9].

Hasil dari deteksi *blob* ini adalah kumpulan *blob* yang setiap *blob*-nya memiliki informasi *centroid* (titik pusat koordinat *blob*), *weight* (lebar *blob*), dan *height* (panjang *blob*). Informasi tersebut nantinya digunakan untuk proses pendeteksian kecepatan.

Berikut ini adalah blok diagram proses deteksi *blob*:



Gambar 4.12. Diagram Alir Deteksi *Blob*.

4.2.2.4 Perancangan Proses Deteksi Kecepatan Kendaraan

i. Mencari Titik Pusat dari Kendaraan

Proses mencari jarak perpindahan antar pixel dilakukan dengan mencari koordinat titik pusat dan menghitung jarak antar titik pusat. Untuk mendapatkan koordinat titik pusat maka terlebih dahulu mengolah objek-objek hasil dari analisis *blob*.

Secara sederhana, untuk menentukan titik pusat, langkah-langkah yang dilakukan adalah :

1. Menghitung koordinat x , dengan :

$$\text{Posisi } x = \text{bbox}(1) + \left(\frac{\text{bbox}(3)}{2} \right) \quad (4.1)$$

2. Menghitung koordinat y , dengan :

$$\text{Posisi } y = \text{bbox}(2) + \left(\frac{\text{bbox}(4)}{2} \right) \quad (4.2)$$

dimana :

$\text{bbox}(1)$ = nilai matriks *boundingbox* kolom ke 1

$\text{bbox}(2)$ = nilai matriks *boundingbox* kolom ke 2

$\text{bbox}(3)$ = nilai matriks *boundingbox* kolom ke 3

$\text{bbox}(4)$ = nilai matriks *boundingbox* kolom ke 4

ii. Pengukuran Jarak

Untuk mengukur kecepatan sebuah obyek, harus diketahui informasi atas variabel yang berpengaruh, diantaranya variabel jarak dan variabel waktu. Variabel jarak didapat dari jarak diantara dua titik pusat obyek dari *frame* satu dengan lainnya yang bisa dihitung dengan menggunakan persamaan (2.16)[4].

iii. Transformasi jarak ke jarak sebenarnya

Pengukuran jarak yang didapat dari tahap sebelumnya yaitu dalam satuan *pixel*, oleh karena itu jarak tersebut harus diubah ke jarak sebenarnya. Jarak kamera serta sudut pemasangan kamera untuk perekaman obyek sangat

berpengaruh pada transformasi jarak. Untuk menghitung hubungan antar pixel dengan jarak sebenarnya adalah :

$$z = \frac{1}{x/y} \quad (4.3)$$

dimana :

x = jarak jangkauan sebenarnya

y = jarak jangkauan pada citra

z = hasil hubungan antar pixel dengan jarak sebenarnya

iv. Menghitung Waktu

Untuk mendapatkan variabel waktu maka akan dicari waktu yang dibutuhkan obyek dari mulai *start* ROI hingga *finish* ROI. *Frame rate per second* adalah banyaknya *frame* yang ditangkap dalam satu detik. Untuk menghitung waktu bisa dihitung dengan menggunakan rumus :

$$\Delta t = f \times w \quad (4.4)$$

dimana :

f = jumlah *frame* (*frame*)

w = waktu yang diperlukan satu *frame* $\left(\frac{\text{detik}}{\text{frame}}\right)$

Δt = waktu (*detik*)

v. Deteksi Kecepatan

Deteksi kecepatan kendaraan bergerak merupakan langkah akhir dari semua proses, dengan tujuan mendapatkan informasi berapa kecepatan kendaraan tersebut dalam satuan *cm/s*. Setelah didapatkan jarak (Δs) dan waktu (Δt) maka bisa dideteksi kecepatannya menggunakan persamaan (2.17)[4].

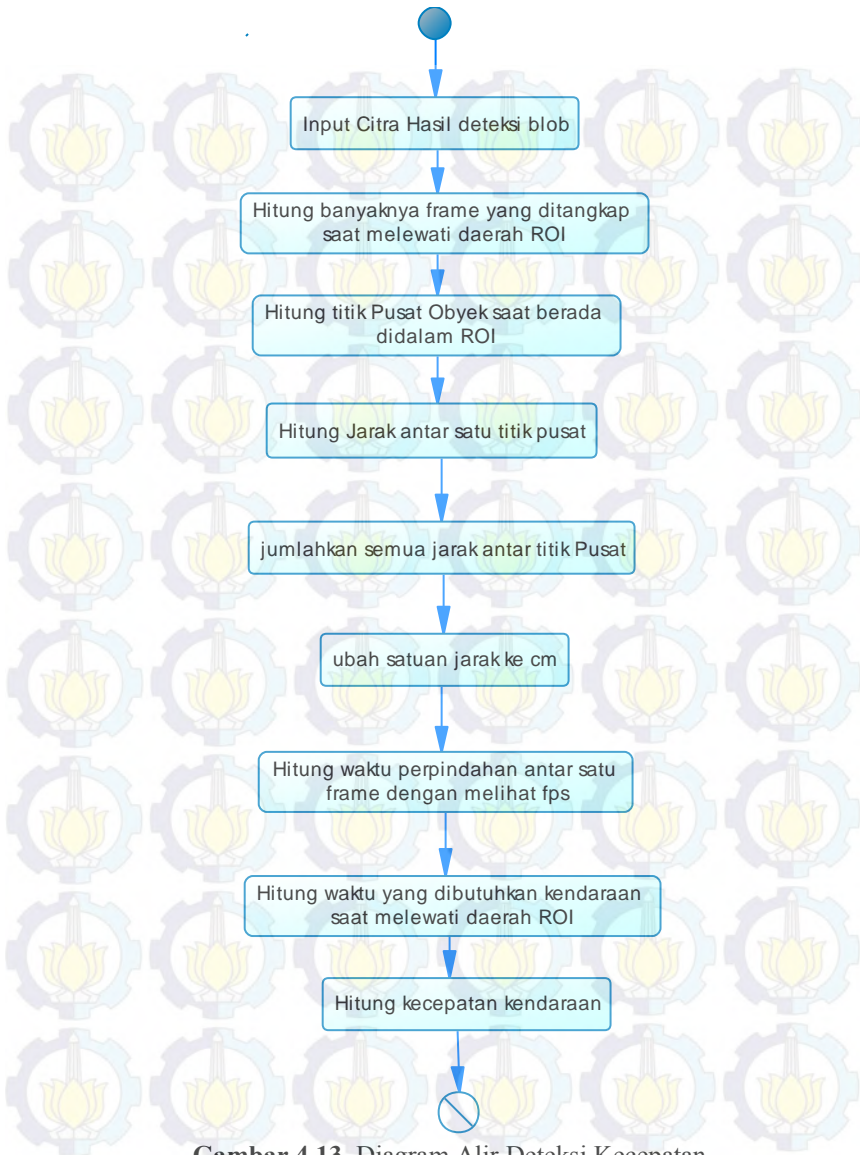
Proses deteksi kecepatan kendaraan dengan cara mengolah objek-objek hasil dari analisis *blob*. Untuk mengukur kecepatan sebuah obyek, harus diketahui informasi atas variabel yang berpengaruh, diantaranya variabel jarak dan variabel waktu. Proses mencari jarak perpindahan antar

pixel dilakukan dengan mencari koordinat titik pusat dan menghitung jarak antar titik pusat. Secara sederhana, untuk menentukan titik pusat bisa menggunakan persamaan (4.1) dan (4.2) yang telah dijelaskan diatas. Setelah dilakukan perhitungan titik pusat (x,y) pada *frame* ke- i dan *frame* ke- $i-1$, maka nilai yang didapat tersebut akan digunakan untuk menghitung jarak yang ditempuh kendaraan saat melewati daerah ROI. Variabel jarak didapat dari jarak diantara dua titik pusat obyek dari *frame* satu dengan lainnya yang bisa dihitung dengan menggunakan persamaan (2.12) yang juga telah dijelaskan pada BAB II[4].

Input yang dibutuhkan untuk proses deteksi kecepatan adalah:

- a. Jarak horizontal antara kamera dan kendaraan saat pertama kali masuk
- b. Tinggi pemasangan kamera yang digunakan saat merekam
- c. Sudut pandang yang dicakup oleh kamera
- d. Sudut pandang daerah yang tidak tertangkap oleh kamera
- e. *Framerate* dari video

Diagram alir pendeteksian kecepatan kendaraan dapat dilihat pada Gambar 4.13.



Gambar 4.13. Diagram Alir Deteksi Kecepatan.

4.2.3 Perancangan Antar Muka Sistem

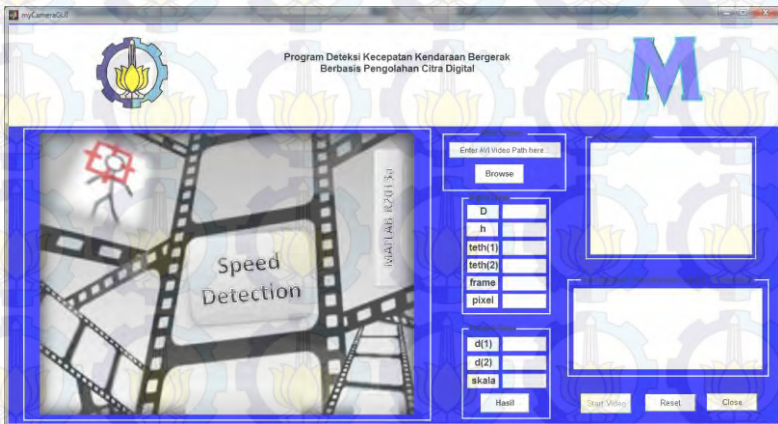
Desain antar muka sistem dibutuhkan agar pengguna dengan mudah mengoperasikan perangkat lunak yang dibangun. Desain ini dibuat semenarik mungkin dan *user friendly*. Berikut adalah desain antar muka halaman utama dan halaman detail proses:

4.2.3.1 Perancangan Halaman Utama

Halaman utama ini memiliki desain antar muka yang ditunjukkan pada Gambar 4.14. Pada halaman ini dilakukan proses pemilihan video, pemilihan ROI, dan mengisi parameter proses. Halaman ini menampilkan video hasil deteksi kecepatan kendaraan. Antar muka halaman utama terdiri dari :

1. *Axis1*, berfungsi untuk menampilkan frame pertama dari video input.
2. *Axis2*, berfungsi untuk menampilkan objek yang akan dilacak pergerakannya. Objek ini diperoleh setelah pengguna memilih ROI objek pada frame pertama yang ditampilkan pada *axis1*
3. *Push button Browse*, berfungsi untuk memilih video input yang akan digunakan dalam proses pelacakan.
4. *Push button Start Video*, berfungsi untuk menjalankan video dan juga untuk melakukan proses pendeteksian kecepatan
5. *Push button Reset* digunakan untuk menghapus semua historis dari proses sebelumnya.
6. *Push button Hasil* digunakan untuk menghitung beberapa parameter output yang akan digunakan untuk proses pendeteksian kecepatan.
7. *Edit text D*, berfungsi agar pengguna dapat menentukan ukuran jarak horizontal antara kamera dan kendaraan saat pertama kali tertangkap oleh kamera. Ukuran blok ini digunakan untuk melakukan perhitungan jarak.
8. *Edit text h*, berfungsi agar pengguna dapat menentukan ukuran tinggi pemasangan kamera.

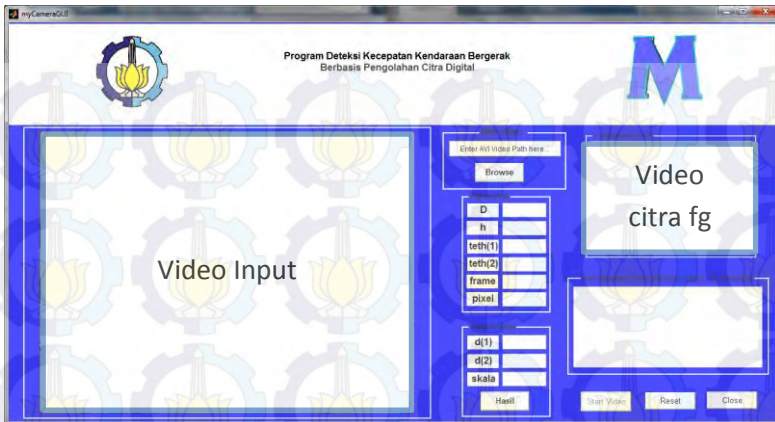
9. *Edit text teth(1)*, berfungsi agar pengguna dapat menentukan ukuran sudut pandang yang dicakup oleh kamera.
10. *Edit text teth(2)*, berfungsi agar pengguna dapat menentukan ukuran sudut daerah yang tidak tercakup oleh kamera.
11. *Edit text Frame*, berfungsi agar pengguna dapat menentukan *framerate* dari video.
12. *Edit text Pixel*, berfungsi agar pengguna dapat menentukan *hight pixel* dari video.



Gambar 4.14. Antar muka halaman awal.

4.2.3.2 Perancangan Halaman Detail

Pada halaman detail ini ditampilkan video *input*, video citra *foreground*. Desain antar muka dari halaman detail ditunjukkan pada Gambar 4.15.



Gambar 4.15. Antar muka halaman detail.

4.3 Implementasi Sistem

4.3.1 Implementasi *Input* Video

Pada proses ini akan ditentukan video yang nantinya digunakan sebagai data masukan untuk diproses. Penjabaran tentang proses pemilihan video adalah sebagai berikut :

- Fungsi : menginputkan video bertipe *.avi pada form simulasi
 Input : video bertipe *.avi
 Deskripsi : mengambil video bertipe*.avi yang telah disimpan pada komputer

Gambar 4.16 adalah tampilan antar muka pengambilan *input* video. Kode program untuk menerima *input* video adalah sebagai berikut :

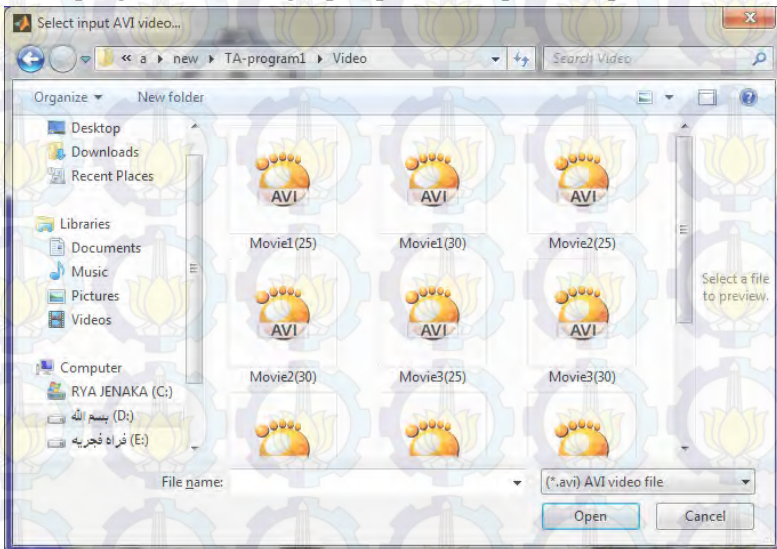
```
[FileName, PathName] =
uigetfile({'*.avi', '(*.avi) AVI video
file'}, ...
'Select input AVI video...');
if FileName~=0
```

```

set(handles.Vid_path, 'String', strcat(PathName, FileName));
    Vid_path_Callback(handles.Vid_path, [], handles);
End

```

Kode program lebih lengkap dapat dilihat pada lampiran.



Gambar 4.16. Antar muka *input* video

4.3.2 Implementasi Pemilihan area ROI

Region of Interest (ROI) adalah daerah bagian dari citra atau *frame* yang akan diproses pada sistem ini. Pemilihan ROI ini bertujuan untuk memudahkan dan membantu program dalam menspesifikasikan kebutuhan program. Ukuran ROI pada sistem ini dianjurkan memuat setengah dari ukuran *frame* inputan. Sehingga dapat menangkap objek kendaraan secara utuh dan hasil yang diinginkan lebih akurat dan lebih efisien. Hal ini dikarenakan program hanya mengolah informasi dari bagian

piksel yang dibutuhkan saja yaitu ROI. Kode program untuk pemilihan ROI adalah sebagai berikut :

```
h = imrect;
setColor(h, 'red');
position = wait(h);
if ~isempty(position)
    set(handles.startVideo, 'Enable', 'on');
end
```

Kode program selengkapnya disajikan pada Lampiran. Gambar 4.17 merupakan tampilan program setelah pemilihan ROI dilakukan.



Gambar 4.17. Kotak warna merah pada *frame* tersebut adalah area ROI yang dipilih.

4.3.3 Implementasi Proses GMM

Proses GMM digunakan untuk mencari citra *foreground* dan *background*. Citra *foreground* merupakan citra yang merepresentasikan keadaan objek yang bergerak. Parameter untuk menggunakan proses GMM ini adalah: Banyaknya komponen gaussian digunakan adalah 3 sesuai hasil penelitian[9]. Kode program GMM dalam Tugas Akhir ini dapat dilihat pada Lampiran.

```
obj.detector =
vision.ForegroundDetector('NumGaussians', 3,
...
'NumTrainingFrames', 40,
'MinimumBackgroundRatio', 0.7);
...
```

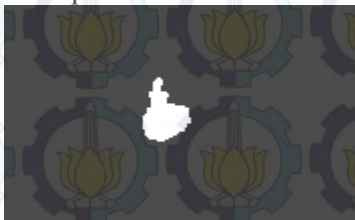
4.3.4 Implementasi *filtering* citra

Proses *filtering* citra dilakukan untuk menghilangkan *noise* atau objek gangguan yang tidak dibutuhkan oleh program. Sehingga program akan dengan mudah mengolah informasi yang ada pada citra.

Citra yang telah difilter kemudian dilakukan proses morfologi pada citra. Proses morfologi ini bertujuan untuk menggabungkan daerah yang memiliki intensitas yang sama. Berikut kode dari proses ini:

```
mask = step(obj.detector, frame);
mask = imfilter(mask, si);
mask = imerode(mask, se);
mask = imdilate(mask, se);
mask = imopen(mask, so);
mask = imclose(mask, sc);
mask = imfill(mask, 'holes');
...
```

Kode lebih lengkap pada Lampiran. Citra hasil proses morfologi dapat dilihat pada Gambar 4.18.



Gambar 4.18. Citra *foreground* yang telah dilakukan proses *filtering* dan morfologi

4.3.5 Implementasi Deteksi *blob*

Deteksi *blob* digunakan untuk menemukan kumpulan *blob* pada suatu citra. Dimana *blob-blob* tersebut akan disimpan sebagai objek kendaraan yang kemudian diolah lebih lanjut untuk mendeteksi kecepatan kendaraan bergerak. Kode program deteksi *blob* dalam Tugas Akhir dapat dilihat pada Lampiran.

```
obj.blobAnalyser =
vision.BlobAnalysis('BoundingBoxOutputPort',
true, ...
'AreaOutputPort', true,
'CentroidOutputPort', true, ...
'MinimumBlobArea', 400);
...
```

4.3.6 Implementasi Pendeteksian Kecepatan Kendaraan

Implementasi ini merupakan tahap terakhir dan juga merupakan fungsi utama program.

Objek-objek hasil dari analisis blob akan dideteksi kecepatannya. Alur program sesuai dengan rancangan pada sub bab 4.2.4.3. Kode program untuk menghitung titik pusat objek dan jumlah *frame* yang ditangkap saat kendaraan melewati daerah ROI adalah sebagai berikut :

```
...
x0 = bboxes(3)/2;
y0 = bboxes(4)/2;

% mencari centroid
x(ids, hitung) = bbox(1) + x0;
y(ids, hitung) = bbox(2) + y0;
citra(ids, hitung) = indx;
hitung = hitung + 1;
indx = indx+1;
...
```

Kode selengkapnya pada Lampiran.

Berikut ini kode program pendeteksian kecepatan kendaraan selama melewati daerah ROI :

```

if hilang(ids) == 0
    a(ids) = length(citra(ids,:));
    if isempty(find(x(ids,')==0))
        index=length(x(ids,:));
        for d = 1:index-1
            deltax(ids,d) = x(ids,d) - x(ids,d+1);
            deltay(ids,d) = y(ids,d) - y(ids,d+1);
            pixeldistance(ids,d) =
            hypot(deltax(ids,d),deltay(ids,d));
        end
    else
        indexs = find(x(ids,')==0);
        index = indexs(1)-1;
        for d = 1:index-1
            deltax(ids,d) = x(ids,d) - x(ids,d+1);
            deltay(ids,d) = y(ids,d) - y(ids,d+1);
            pixeldistance(ids,d) =
            hypot(deltax(ids,d),deltay(ids,d));
        end
    end
totaljarak(ids) = sum(pixeldistance(ids,:));
jarakkalibrasi(ids) = totaljarak(ids);
jaraksbn(ids) = jarakkalibrasi(ids) * skala;
waktu(ids) = index * fram;
v(ids) = jaraksbn(ids) / waktu(ids);
kecepatan = v(ids) * 3600 / 100000;

set(handles.kecepatan, 'String', kecepatan);
...

```

Kode selengkapnya terdapat pada Lampiran. Selanjutnya, setelah pendeteksian selesai dilakukan, maka nilainya akan ditampilkan di dalam halaman. Gambar 4.19 adalah antar muka dari hasil deteksi kecepatan kendaraan.



Gambar 4.19. Antar muka aplikasi saat pendeteksian kecepatan kendaraan.

BAB V

UJI COBA DAN PEMBAHASAN

Bab ini menjelaskan mengenai pengujian program dan pembahasan dari hasil uji coba. Pengujian yang dilakukan disini adalah pengujian program dengan *input* video rekaman kendaraan bergerak di beberapa tempat, diantaranya yaitu di Halaman Sekolah, di Lapangan, di Jalan Raya dan di Jalan Tol. Pengambilan video juga dilakukan pada waktu yang berbeda, yaitu pada waktu pagi, siang dan sore hari yang merepresentasikan kondisi yang berbeda-beda. Objek yang direkam dalam video tersebut adalah Truck, Mobil, Motor, Sepeda dan Orang Jalan Kaki.

5.1 Data Uji Coba

Uji coba pada program dalam Tugas Akhir ini dilakukan terhadap video *.avi. Video uji coba sudah tersimpan dalam penyimpanan komputer dan diperoleh dari hasil rekaman. Video yang digunakan sebanyak 26 video. Video memiliki *framerate* yang berbeda-beda. Daftar input uji coba tersebut antara lain disajikan dalam Tabel 5.1.

Tabel 5.1 Tabel Data Video yang Digunakan.

No	Nama	Framerate	Pixel
1.	Movie1.avi	25	1280x720
2.	Movie2.avi	25	1280x720
3.	Movie3.avi	25	1280x720
4.	Movie4.avi	25	1280x720
5.	Movie5.avi	25	1280x720
6.	Movie6.avi	25	1280x720
7.	Movie1.avi	30	1280x720
8.	Movie2.avi	30	1280x720
9.	Movie3.avi	30	1280x720
10.	Movie4.avi	30	1280x720
11.	Movie5.avi	30	1280x720

12.	Movie6.avi	30	1280x720
13.	Movie0.avi	25	640x480
14.	Movie01.avi	25	640x480
15.	Movie12.avi	25	640x480
16.	Movie0.avi	30	640x480
17.	Movie01.avi	30	640x480
18.	Movie12.avi	30	640x480
19.	DeltaMobil.avi	25	720x576
20.	DeltaMobil2.avi	25	720x576
21.	Jalantol99.avi	25	720x576
22.	Jalantol100.avi	25	720x576
23.	Jalantol105a.avi	25	720x576
24.	Jalantol105b.avi	25	720x576
25.	Jalantol106a.avi	25	720x576
26.	Jalantol109b.avi	25	720x576

5.2 Uji Coba Pendeteksian Kecepatan Kendaraan Bergerak

Pengujian program, yang pertama dilakukan dengan *input* video rekaman kendaraan bergerak di Halaman Sekolah. Pengambilan video dilakukan pada waktu sore hari yang merepresentasikan kondisi memiliki bayangan. Durasi rekaman yang digunakan adalah 15 detik dengan objek yang direkam dalam video tersebut adalah motor, sepeda dan orang jalan kaki.

Pengujian dilakukan dengan memasukkan beberapa nilai parameter, diantaranya yaitu :

1. $D = 2400$ cm
2. $H = 500$ cm
3. $\theta_1 = 80^\circ$
4. $\theta_2 = 10^\circ$
5. *Frame* = sesuai banyaknya *frame* dari hasil rekaman video yaitu 25 atau 30.
6. *Pixel* = 720

Setelah sebuah parameter itu dimasukkan selanjutnya akan dilakukan proses perhitungan untuk mendapatkan hasil

skala perbandingan jarak nyata dengan jarak pada video. Kemudian dilakukan perhitungan kecepatannya yaitu dengan menjalankan hasil rekaman video pada program. Saat kendaraan berjalan dan kemudian memasuki daerah ROI maka pada saat kendaraan berada pada daerah ROI tersebut akan dilakukan proses perhitungan untuk mendapatkan kecepatan dari kendaraan tersebut. Proses perhitungan disini didapat dari proses deteksi objek kendaraan yang kemudian diproses juga dengan parameter yang sudah didapatkan pada proses sebelumnya. Setelah kendaraan keluar dari daerah ROI maka akan diperoleh kecepatan kendaraan saat berada pada daerah ROI tersebut.

Dengan menggunakan video rekaman kendaraan, menghasilkan perhitungan kecepatan kendaraan sebagai berikut :

5.2.1 Rekaman Video Movie1.avi

Tabel 5.2 Tabel Pendeteksian Kecepatan Kendaraan pada Video Movie1.avi

Kendaraan	<i>Framerate</i>	Sebenarnya (km/jam)	Program (km/jam)
Motor	25	18	19,52
Motor	30	18	19,18

Berdasarkan Tabel 5.2 terlihat bahwa hasil deteksi kecepatan kendaraan dengan parameter *framerate* yang berbeda juga menghasilkan hasil deteksi yang berbeda, meskipun nilai yang terpaut tidak terlalu jauh. Hasil deteksi kecepatan dengan *framerate* 25 lebih cepat dibandingkan dengan *framerate* 30, tapi hasil deteksi kecepatan yang lebih mendekati dengan kecepatan sebenarnya yaitu hasil deteksi kecepatan dengan parameter *framerate* 30.

5.2.2 Rekaman Video Movie2.avi

Tabel 5.3 Tabel Pendeteksian Kecepatan Kendaraan pada Video Movie2.avi

Kendaraan	<i>Framerate</i>	Sebenarnya (km/jam)	Program (km/jam)
Motor	25	20	21,49
Motor	30	20	21,17

Berdasarkan Tabel 5.3 terlihat bahwa hasil deteksi kecepatan kendaraan dengan parameter *framerate* yang berbeda juga menghasilkan hasil deteksi kecepatan yang berbeda, meskipun nilai yang terpaut tidak terlalu jauh. Hasil deteksi kecepatan dengan *framerate* 25 lebih cepat dibandingkan dengan *framerate* 30, tapi hasil deteksi kecepatan yang lebih mendekati dengan kecepatan sebenarnya yaitu hasil deteksi kecepatan dengan parameter *framerate* 30.

5.2.3 Rekaman Video Movie3.avi

Tabel 5.4 Tabel Pendeteksian Kecepatan Kendaraan pada Video Movie3.avi

Kendaraan	<i>Framerate</i>	Sebenarnya (km/jam)	Program (km/jam)
Motor	25	24	24,70
Motor	30	24	25,23

Berdasarkan Tabel 5.4 terlihat bahwa hasil deteksi kecepatan kendaraan dengan parameter *framerate* yang berbeda juga menghasilkan hasil deteksi kecepatan yang berbeda, meskipun nilai yang terpaut tidak terlalu jauh. Hasil deteksi kecepatan dengan *framerate* 30 lebih cepat dibandingkan dengan *framerate* 25, tapi hasil deteksi kecepatan yang lebih mendekati dengan kecepatan sebenarnya yaitu hasil deteksi kecepatan dengan parameter *framerate* 25.

5.2.4 Rekaman Video Movie4.avi

Tabel 5.5 Tabel Pendeteksian Kecepatan Kendaraan pada Video Movie4.avi

Kendaraan	<i>Framerate</i>	Sebenarnya (km/jam)	Program (km/jam)
Jalan	25	7	7,798
Sepeda	25	11	10,38
Jalan	30	7	8,581
Sepeda	30	11	11,23

Berdasarkan Tabel 5.5 terlihat bahwa hasil deteksi kecepatan kendaraan dengan parameter *framerate* yang berbeda juga menghasilkan hasil deteksi kecepatan yang berbeda, meskipun nilai yang terpaut tidak terlalu jauh. Untuk objek orang jalan kaki hasil deteksi kecepatan dengan *framerate* 30 lebih cepat dibandingkan dengan *framerate* 25, tapi hasil deteksi kecepatan yang lebih mendekati dengan kecepatan sebenarnya yaitu hasil deteksi kecepatan dengan parameter *framerate* 25. Sedangkan untuk objek sepeda hasil deteksi kecepatan dengan *framerate* 30 lebih cepat dibandingkan dengan *framerate* 25, tapi hasil deteksi kecepatan yang lebih mendekati dengan kecepatan sebenarnya yaitu hasil deteksi kecepatan dengan parameter *framerate* 30.

5.2.5 Rekaman Video Movie5.avi

Tabel 5.6 Tabel Pendeteksian Kecepatan Kendaraan pada Video Movie5.avi

Kendaraan	<i>Framerate</i>	Sebenarnya (km/jam)	Program (km/jam)
Jalan	25	7	7,561
Sepeda	25	10	9,629
Jalan	30	7	7,953
Sepeda	30	10	10,38

Berdasarkan Tabel 5.6 terlihat bahwa hasil deteksi kecepatan kendaraan dengan parameter *framerate* yang berbeda juga menghasilkan hasil deteksi kecepatan yang berbeda, meskipun nilai yang terpaut tidak terlalu jauh. Untuk objek orang jalan kaki hasil deteksi kecepatan dengan *framerate* 30 lebih cepat dibandingkan dengan *framerate* 25, tapi hasil deteksi kecepatan yang lebih mendekati dengan kecepatan sebenarnya yaitu hasil deteksi kecepatan dengan parameter *framerate* 25. Sedangkan untuk objek sepeda hasil deteksi kecepatan dengan *framerate* 30 lebih cepat dibandingkan dengan *framerate* 25, tapi hasil deteksi kecepatan dengan *framerate* yang berbeda tersebut hasilnya mempunyai selisih yang sama dengan kecepatan sebenarnya.

5.2.6 Rekaman Video Movie6.avi

Tabel 5.7 Tabel Pendeteksian Kecepatan Kendaraan pada Movie6.avi

Kendaraan	<i>Framerate</i>	Sebenarnya (km/jam)	Program (km/jam)
Jalan	25	8	8,198
Sepeda	25	11	10,29
Jalan	30	8	9,561
Sepeda	30	11	10,53

Berdasarkan Tabel 5.7 terlihat bahwa hasil deteksi kecepatan kendaraan dengan parameter *framerate* yang berbeda juga menghasilkan hasil deteksi kecepatan yang berbeda, meskipun nilai yang terpaut tidak terlalu jauh. Untuk objek orang jalan kaki hasil deteksi kecepatan dengan *framerate* 30 lebih cepat dibandingkan dengan *framerate* 25, tapi hasil deteksi kecepatan yang lebih mendekati dengan kecepatan sebenarnya yaitu hasil deteksi kecepatan dengan parameter

framerate 25. Sedangkan untuk objek sepeda hasil deteksi kecepatan dengan *framerate* 30 lebih cepat dibandingkan dengan *framerate* 25, tapi hasil deteksi kecepatan yang lebih mendekati dengan kecepatan sebenarnya yaitu hasil deteksi kecepatan dengan parameter *framerate* 30.

Uji coba percobaan yang kedua dilakukan dengan *input* video rekaman kendaraan bergerak di Lapangan. Pengambilan video dilakukan pada waktu pagi hari yang merepresentasikan kondisi memiliki bayangan. Durasi rekaman yang digunakan adalah 7 detik dengan objek yang direkam dalam video tersebut adalah motor.

Pengujian dilakukan dengan memasukkan beberapa nilai parameter, diantaranya yaitu :

1. $D = 2600 \text{ cm}$
2. $H = 300 \text{ cm}$
3. $\theta_1 = 85^\circ$
4. $\theta_2 = 10^\circ$
5. *Frame* = sesuai banyaknya *frame* dari hasil rekaman video yaitu 25 atau 30
6. *Pixel* = 480

5.2.7 Rekaman Video Movie0.avi

Tabel 5.8 Tabel Pendeteksian Kecepatan Kendaraan pada Movie0.avi

Kendaraan	<i>Framerate</i>	Sebenarnya (km/jam)	Program (km/jam)
Motor	25	30	35,94
Motor	30	30	28,10

Berdasarkan Tabel 5.8 terlihat bahwa hasil deteksi kecepatan kendaraan dengan parameter *framerate* yang berbeda juga menghasilkan hasil deteksi kecepatan yang berbeda, dan

nilai yang terpaut sudah cukup jauh. Hasil deteksi kecepatan dengan *framerate* 25 lebih cepat dibandingkan dengan *framerate* 30, tapi hasil deteksi kecepatan yang lebih mendekati dengan kecepatan sebenarnya yaitu hasil deteksi kecepatan dengan parameter *framerate* 30.

Pengujian selanjutnya juga dilakukan dengan *input* video rekaman kendaraan bergerak di Lapangan tapi dengan memasukkan beberapa nilai parameter yang berbeda dengan sebelumnya, diantaranya yaitu :

1. $D = 2400 \text{ cm}$
2. $H = 350 \text{ cm}$
3. $\theta_1 = 80^\circ$
4. $\theta_2 = 10^\circ$
5. *Frame* = sesuai banyaknya *frame* dari hasil rekaman video yaitu 25 atau 30

5.2.8 Rekaman Video Movie01.avi

Tabel 5.9 Tabel Pendeteksian Kecepatan Kendaraan pada Movie01.avi

Kendaraan	<i>Framerate</i>	Sebenarnya (km/jam)	Program (km/jam)
Motor	25	30	36,40
Motor	30	30	33,63

Berdasarkan Tabel 5.9 terlihat bahwa hasil deteksi kecepatan kendaraan dengan parameter *framerate* yang berbeda juga menghasilkan hasil deteksi kecepatan yang berbeda, dan nilai yang terpaut sudah cukup jauh. Dengan kecepatan sebenarnya yang berbeda, hasil deteksi programnya juga berbeda. Dari Tabel terlihat bahwa hasil deteksi kecepatan untuk objek Motor dengan kecepatan sebenarnya 30 km/jam yang lebih mendekati dengan kecepatan sebenarnya yaitu hasil deteksi kecepatan dengan parameter *framerate* 30.

5.2.9 Rekaman Video Movie12.avi

Tabel 5.10 Tabel Pendeteksian Kecepatan Kendaraan pada Movie12.avi

Kendaraan	<i>Framerate</i>	Sebenarnya (km/jam)	Program (km/jam)
Motor	25	35	36,36
Motor	30	35	38,67

Berdasarkan Tabel 5.10 terlihat bahwa hasil deteksi kecepatan kendaraan dengan parameter *framerate* yang berbeda juga menghasilkan hasil deteksi kecepatan yang berbeda, dan nilai yang terpaut sudah cukup jauh. Dengan kecepatan sebenarnya yang berbeda, hasil deteksi programnya juga berbeda. Dari Tabel terlihat untuk objek Motor dengan kecepatan sebenarnya 35 km/jam terlihat bahwa hasil deteksi kecepatan yang lebih mendekati dengan kecepatan sebenarnya yaitu hasil deteksi kecepatan dengan parameter *framerate* 25.

Dan berdasarkan percobaan diatas terlihat bahwa hasil deteksi dengan pemasangan kamera yang tinggi akan menghasilkan tingkat akurasi yang lebih baik.

Selanjutnya uji coba percobaan dilakukan dengan *input* video rekaman kendaraan bergerak di Jalan Raya, yaitu Jalan Raya depan Delta. Pengambilan video dilakukan pada waktu pagi hari yang merepresentasikan kondisi memiliki bayangan. Durasi rekaman yang digunakan adalah 5 detik dengan objek yang direkam dalam video tersebut adalah Mobil. Percobaan dilakukan untuk menghitung kecepatan tanpa mengetahui kecepatan sebenarnya.

Pengujian dilakukan dengan memasukkan beberapa nilai parameter, diantaranya yaitu :

1. $D = 2100 \text{ cm}$
2. $H = 500 \text{ cm}$
3. $\theta_1 = 75^\circ$
4. $\theta_2 = 10^\circ$
5. $Frame = 25$
6. $Pixel = 576$

5.2.10 Rekaman Video DeltaMobil.avi

Tabel 5.11 Tabel Pendeteksian Kecepatan Kendaraan pada DeltaMobil.avi

Kendaraan	<i>Framerate</i>	Sebenarnya (km/jam)	Program (km/jam)
Mobil1	25	-	42,78

Percobaan menggunakan video rekaman arus kendaraan *real* di Jalan Raya yang tanpa mengetahui kecepatan sebenarnya ini dilakukan dengan menggunakan dasar dari percobaan sebelumnya yang telah dilakukan untuk percobaan pada rekaman video yang terdapat data kecepatan sebenarnya. Dan berdasarkan Tabel 5.11, hasil deteksinya sudah cukup sesuai.

5.2.11 Rekaman Video DeltaMobil2.avi

Tabel 5.12 Tabel Pendeteksian Kecepatan Kendaraan pada DeltaMobil2.avi

Kendaraan	<i>Framerate</i>	Sebenarnya (km/jam)	Program (km/jam)
Mobil1	25	-	27,53
Mobil2	25	-	30,32

Percobaan ini juga menggunakan video rekaman arus kendaraan *real* di Jalan Raya yang tanpa mengetahui kecepatan sebenarnya. Dan berdasarkan Tabel 5.12, hasil deteksinya sudah cukup sesuai.

Pengujian selanjutnya dilakukan dengan *input* video rekaman kendaraan bergerak di Jalan Tol, yaitu Jalan Tol Manyar. Pengambilan video dilakukan pada waktu siang hari yang merepresentasikan kondisi sedikit memiliki bayangan. Durasi rekaman yang digunakan adalah 28 detik dengan objek yang direkam dalam video tersebut adalah Mobil dan Truck. Percobaan dilakukan untuk menghitung kecepatan tanpa mengetahui kecepatan sebenarnya.

Pengujian dilakukan dengan memasukkan beberapa nilai parameter, diantaranya yaitu :

1. $D = 4400 \text{ cm}$
2. $H = 500 \text{ cm}$
3. $\theta_1 = 84^\circ$
4. $\theta_2 = 10^\circ$
5. $\text{Frame} = 25$
6. $\text{Pixel} = 576$

5.2.12 Rekaman Video jalantol99.avi

Tabel 5.13 Tabel Pendeteksian Kecepatan Kendaraan pada jalantol99.avi

Kendaraan	Framerate	Sebenarnya (km/jam)	Program (km/jam)
Objek1	25	-	81,02
Objek2	25	-	83,68
Objek3	25	-	73,26
Objek4	25	-	57,80

Percobaan dengan menggunakan video rekaman arus kendaraan *real* di Jalan Tol yang tanpa mengetahui kecepatan. Dan berdasarkan Tabel 5.13, hasil deteksinya sudah cukup sesuai.

Pengujian selanjutnya dengan nilai parameter yang berbeda, yaitu :

1. $D = 5400 \text{ cm}$
2. $H = 500 \text{ cm}$
3. $\theta_1 = 85^\circ$
4. $\theta_2 = 10^\circ$
5. $Frame = 25$
6. $Pixel = 576$

5.2.13 Rekaman Video jalantol100.avi

Tabel 5.14 Tabel Pendeteksian Kecepatan Kendaraan pada jalantol100.avi

Kendaraan	Framerate	Sebenarnya (km/jam)	Program (km/jam)
Objek1	25	-	95,84

Percobaan ini juga menggunakan video rekaman arus kendaraan *real* di Jalan Tol yang juga tidak diketahui kecepatan sebenarnya. Dengan masukan parameter yang berbeda dengan percobaan sebelumnya, berdasarkan Tabel 5.14 terlihat bahwa kecepatan yang dihasilkan jika dilihat dengan kondisi saat melintasi di Jalan Tol tersebut sangat memungkinkan pengendara mengendarai Mobil dengan kecepatan tersebut.

Pengujian dengan nilai parameter yang berbeda, yaitu :

1. $D = 5000 \text{ cm}$
2. $H = 500 \text{ cm}$
3. $\theta_1 = 84^\circ$
4. $\theta_2 = 10^\circ$
5. $Frame = 25$
6. $Pixel = 576$

5.2.14 Rekaman Video jalantol105a.avi

Tabel 5.15 Tabel Pendeteksian Kecepatan Kendaraan pada jalantol105a.avi

Kendaraan	Framerate	Sebenarnya (km/jam)	Program (km/jam)
Objek1	25	-	92,31
Objek2	25	-	96,27

Percobaan di atas juga menggunakan video rekaman arus kendaraan *real* di Jalan Tol yang juga tidak diketahui kecepatan sebenarnya. Dengan masukan parameter yang berbeda dengan percobaan sebelumnya, berdasarkan Tabel 5.15 terlihat bahwa kecepatan yang dihasilkan jika dilihat dengan kondisi saat melintasi di Jalan Tol tersebut sangat memungkinkan pengendara mengendarai Mobil dengan kecepatan tersebut.

5.2.15 Rekaman Video jalantol105b.avi

Tabel 5.16 Tabel Pendeteksian Kecepatan Kendaraan pada jalantol105b.avi

Kendaraan	Framerate	Sebenarnya (km/jam)	Program (km/jam)
Objek1	25	-	86,10
Objek2	25	-	48,99

Percobaan ini juga menggunakan video rekaman arus kendaraan *real* di Jalan Tol yang juga tidak diketahui kecepatan sebenarnya. Dengan masukan parameter yang sama, berdasarkan Tabel 5.16 terlihat bahwa kecepatan yang dihasilkan jika dilihat dengan kondisi saat melintasi di Jalan Tol tersebut sangat memungkinkan pengendara mengendarai Mobil dengan kecepatan tersebut.

5.2.16 Rekaman Video jalantol106a.avi

Tabel 5.17 Tabel Pendeteksian Kecepatan Kendaraan pada jalantol106a.avi

Kendaraan	<i>Framerate</i>	Sebenarnya (km/jam)	Program (km/jam)
Objek1	25	-	91,42
Objek2	25	-	51,64
Objek3	25	-	53,33

Percobaan ini juga menggunakan video rekaman arus kendaraan *real* di Jalan Tol yang juga tidak diketahui kecepatan sebenarnya. Dengan masukan parameter yang masih sama, berdasarkan Tabel 5.17 terlihat bahwa kecepatan yang dihasilkan jika dilihat dengan kondisi saat melintasi di Jalan Tol tersebut sangat memungkinkan pengendara mengendarai Mobil dengan kecepatan tersebut.

Pengujian dengan nilai parameter yang berbeda, yaitu :

1. $D = 3700 \text{ cm}$
2. $H = 500 \text{ cm}$
3. $\theta_1 = 80^\circ$
4. $\theta_2 = 10^\circ$
5. $Frame = 25$
6. $Pixel = 576$

5.2.17 Rekaman Video jalantol109b.avi

Tabel 5.18 Tabel Pendeteksian Kecepatan Kendaraan pada jalantol109b.avi

Kendaraan	<i>Framerate</i>	Sebenarnya (km/jam)	Program (km/jam)
Objek1	25	-	57,80
Objek2	25	-	78,73

Objek3	25	-	79,99
--------	----	---	-------

Percobaan dengan menggunakan video rekaman arus kendaraan *real* di Jalan Tol yang juga tidak diketahui kecepatan sebenarnya. Dengan masukan parameter yang berbeda dengan percobaan sebelumnya, berdasarkan Tabel 5.18 terlihat bahwa dihasilkan kecepatan yang jika dilihat dengan kondisi saat melintasi di Jalan Tol tersebut sangat memungkinkan pengendara mengendarai Mobil dengan kecepatan tersebut.

5.2.18 Pembahasan Uji Coba Pendeteksian Kecepatan Kendaraan

Berdasarkan Tabel 5.2 sampai dengan Tabel 5.18 kemampuan program dalam mendeteksi kecepatan juga dipengaruhi oleh *framerate* dan *pixel*. Jika ada bayangan dari benda lain selain kendaraan tersebut, maka hal tersebut bisa mengganggu dalam pendeteksian kecepatannya. Program juga dapat mendeteksi kecepatan kendaraan bergerak dengan baik pada video Movie1.avi, Movie2.avi, Movie3.avi, Movie4.avi, Movie5.avi, Movie6.avi, Movie0.avi, Movie01.avi, Movie12.avi. Hal itu terlihat dari Tabel, dimana penghitungan program hampir mendekati dengan kecepatan kendaraan sebenarnya.

Dari data yang tersaji pada Tabel 5.2 sampai dengan Tabel 5.10 dapat dihitung tingkat akurasi keberhasilan program dalam mendeteksi kecepatan kendaraan bergerak. Penghitungan tingkat akurasi keberhasilan ini didasarkan pada persamaan berikut:

$$PK = \frac{KS - \text{abs}(KS - KP)}{KS} \times 100$$

dengan,

PK = Prosentase Keberhasilan,

KP = Kecepatan kendaraan yang terdeteksi program

KS = Kecepatan kendaraan sebenarnya

Maka, tingkat akurasi keberhasilan program dalam mendeteksi kendaraan bergerak dapat dilihat pada tabel berikut:

a. Rekaman Video Movie1.avi

Tabel 5.19 Tabel Prosentase Keberhasilan Video Movie1.avi

Kendaraan	<i>Framerate</i>	Sebenarnya (km/jam)	Program (km/jam)	PK
Motor	25	18	19,52	91,56%
Motor	30	18	19,18	93,44%

Berdasarkan Tabel 5.19 terlihat bahwa selain hasil deteksi kecepatan yang berbeda, tingkat akurasi keberhasilan program dalam mendeteksi kecepatan kendaraan dengan parameter *framerate* yang berbeda juga menghasilkan tingkat akurasi yang berbeda, meskipun nilai yang terpaut tidak terlalu jauh. Tingkat akurasi keberhasilan yang paling tinggi terdapat pada uji coba dengan parameter *framerate* 30.

b. Rekaman Video Movie2.avi

Tabel 5.20 Tabel Prosentase Keberhasilan Video Movie2.avi

Kendaraan	<i>Framerate</i>	Sebenarnya (km/jam)	Program (km/jam)	PK
Motor	25	20	21,49	86,89%
Motor	30	20	21,17	94,15%

Berdasarkan Tabel 5.20 terlihat bahwa selain hasil deteksi kecepatan yang berbeda, tingkat akurasi keberhasilan program dalam mendeteksi kecepatan kendaraan dengan parameter *framerate* yang berbeda juga menghasilkan tingkat akurasi yang berbeda. Tingkat akurasi keberhasilan yang

paling tinggi juga terdapat pada uji coba dengan parameter *framerate* 30.

c. Rekaman Video Movie3.avi

Tabel 5.21 Tabel Prosentase Keberhasilan Video Movie3.avi

Kendaraan	<i>Framerate</i>	Sebenarnya (km/jam)	Program (km/jam)	PK
Motor	25	24	24,70	97,08%
Motor	30	24	25,23	94,87%

Berdasarkan Tabel 5.21 terlihat bahwa selain hasil deteksi kecepatan yang berbeda, tingkat akurasi keberhasilan program dalam mendeteksi kecepatan kendaraan dengan parameter *framerate* yang berbeda juga menghasilkan tingkat akurasi yang berbeda, meskipun nilai yang terpaut tidak terlalu jauh. Tingkat akurasi keberhasilan yang paling tinggi terdapat pada uji coba dengan parameter *framerate* 25.

d. Rekaman Video Movie4.avi

Tabel 5.22 Tabel Prosentase Keberhasilan Video Movie4.avi

Kendaraan	<i>Framerate</i>	Sebenarnya (km/jam)	Program (km/jam)	PK
Jalan	25	7	7,798	88,6%
Sepeda	25	11	10,38	94,45%
Jalan	30	7	8,581	77,41%
Sepeda	30	11	11,23	97,90%

Berdasarkan Tabel 5.22 terlihat bahwa selain hasil deteksi kecepatan yang berbeda, tingkat akurasi keberhasilan program dalam mendeteksi kecepatan kendaraan dengan parameter *framerate* yang berbeda juga menghasilkan tingkat akurasi yang berbeda. Untuk objek berupa Sepeda, tingkat akurasi keberhasilan yang paling tinggi terdapat pada uji coba dengan parameter *framerate* 30, sedangkan untuk objek

Orang Jalan Kaki, tingkat akurasi keberhasilan yang paling tinggi juga terdapat pada uji coba dengan parameter *framerate* 25.

e. Rekaman Video Movie5.avi

Tabel 5.23 Tabel Prosentase Keberhasilan Video Movie5.avi

Kendaraan	<i>Framerate</i>	Sebenarnya (km/jam)	Program (km/jam)	PK
Jalan	25	7	7,561	91,98%
Sepeda	25	10	9,629	96,3%
Jalan	30	7	7,953	86,38%
Sepeda	30	10	10,38	96,2%

Berdasarkan Tabel 5.23 terlihat bahwa selain hasil deteksi kecepatan yang berbeda, tingkat akurasi keberhasilan program dalam mendeteksi kecepatan kendaraan dengan parameter *framerate* yang berbeda juga menghasilkan tingkat akurasi yang berbeda. Untuk objek berupa Sepeda, tingkat akurasi keberhasilannya hampir sama baik menggunakan *framerate* 25 ataupun 30, sedangkan untuk objek Orang Jalan Kaki, tingkat akurasi keberhasilan yang paling tinggi juga terdapat pada uji coba dengan parameter *framerate* 25.

f. Rekaman Video Movie6.avi

Tabel 5.24 Tabel Prosentase Keberhasilan Video Movie6.avi

Kendaraan	<i>Framerate</i>	Sebenarnya (km/jam)	Program (km/jam)	PK
Jalan	25	8	8,198	97,52%
Sepeda	25	11	10,29	93,54%
Jalan	30	8	9,561	80,48%
Sepeda	30	11	10,53	95,72%

Berdasarkan Tabel 5.24 terlihat bahwa selain hasil deteksi kecepatan yang berbeda, tingkat akurasi keberhasilan program dalam mendeteksi kecepatan kendaraan dengan parameter *framerate* yang berbeda juga menghasilkan tingkat akurasi yang berbeda. Untuk objek berupa Sepeda, tingkat akurasi keberhasilan yang paling tinggi terdapat pada uji coba dengan parameter *framerate* 30, sedangkan untuk objek Orang Jalan Kaki, tingkat akurasi keberhasilan yang paling tinggi juga terdapat pada uji coba dengan parameter *framerate* 25.

g. Rekaman Video Movie0.avi

Tabel 5.25 Tabel Prosentase Keberhasilan Video Movie0.avi

Kendaraan	<i>Framerate</i>	Sebenarnya (km/jam)	Program (km/jam)	PK
Motor	25	30	35,94	80,2%
Motor	30	30	28,1	93,67%

Berdasarkan Tabel 5.25 terlihat bahwa selain hasil deteksi kecepatan yang berbeda, tingkat akurasi keberhasilan program dalam mendeteksi kecepatan kendaraan dengan parameter *framerate* yang berbeda juga menghasilkan tingkat akurasi yang berbeda. Tingkat akurasi keberhasilan yang paling tinggi terdapat pada uji coba dengan parameter *framerate* 30.

h. Rekaman Video Movie01.avi

Tabel 5.26 Tabel Prosentase Keberhasilan Video Movie01.avi

Kendaraan	<i>Framerate</i>	Sebenarnya (km/jam)	Program (km/jam)	PK
Motor	25	30	36,4	78,67%
Motor	30	30	33,63	87,9%

Berdasarkan Tabel 5.26 terlihat bahwa selain hasil deteksi kecepatan yang berbeda, tingkat akurasi keberhasilan program dalam mendeteksi kecepatan kendaraan dengan parameter *framerate* yang berbeda juga menghasilkan tingkat akurasi yang berbeda. Tingkat akurasi keberhasilan yang paling tinggi terdapat pada uji coba dengan parameter *framerate* 30.

i. Rekaman Video Movie12.avi

Tabel 5.27 Tabel Prosentase Keberhasilan Video Movie12.avi

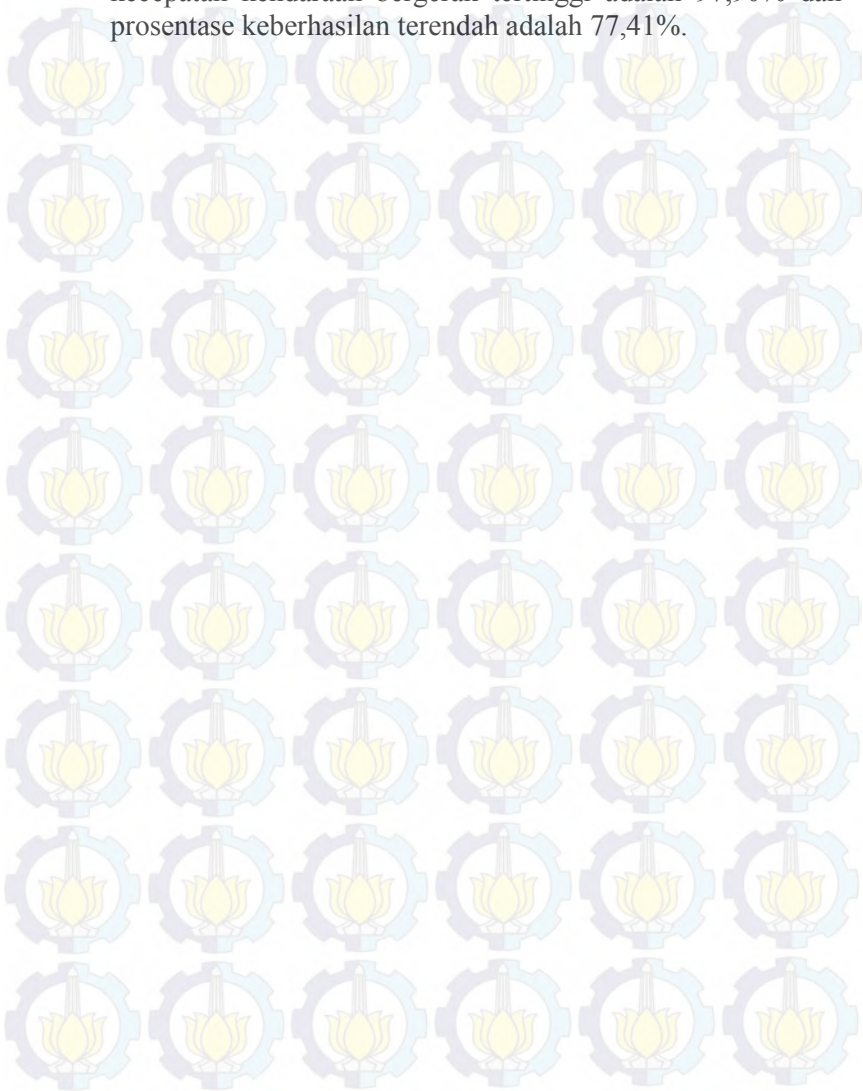
Kendaraan	<i>Framerate</i>	Sebenarnya (km/jam)	Program (km/jam)	PK
Motor	25	35	36,36	95,34%
Motor	30	35	38,67	89,51%

Berdasarkan Tabel 5.27 terlihat bahwa selain hasil deteksi kecepatan yang berbeda, tingkat akurasi keberhasilan program dalam mendeteksi kecepatan kendaraan dengan parameter *framerate* yang berbeda juga menghasilkan tingkat akurasi yang berbeda. Tingkat akurasi keberhasilan yang paling tinggi terdapat pada uji coba dengan parameter *framerate* 25.

Dari data prosentase keberhasilan pada Tabel 5.19 sampai dengan Tabel 5.27, program dapat mendeteksi kecepatan kendaraan bergerak baik dengan *framerate* 25 atau 30. Dan dari 9 Tabel prosentase keberhasilan yang tersaji di atas, tingkat akurasi keberhasilan yang paling baik lebih banyak muncul pada uji coba yang menggunakan parameter *framerate* 30.

Untuk kendaraan dengan kecepatan objek yang paling cepat, tingkat akurasi keberhasilannya lebih baik menggunakan parameter *framerate* 25.

Prosentase keberhasilan program dalam mendeteksi kecepatan kendaraan bergerak tertinggi adalah 97,90% dan prosentase keberhasilan terendah adalah 77,41%.



LAMPIRAN A

A.1 Kode Program myCameraGUI.m

```

function varargout = myCameraGUI(varargin)
% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',
    mfilename, ...
    'gui_Singleton', gui_Singleton, ...
    'gui_OpeningFcn',
    @myCameraGUI_OpeningFcn, ...
    'gui_OutputFcn', @myCameraGUI_OutputFcn,
    ...
    'gui_LayoutFcn', [], ...
    'gui_Callback', []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback =
    str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] =
    gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT

% --- Executes just before mycameragui is
% made visible.
function myCameraGUI_OpeningFcn(hObject,
    eventdata, handles, varargin)
% Choose default command line output for
mycameragui
handles.output = hObject;

```


LAMPIRAN A (LANJUTAN)

```

background =
axes('unit','normalized','position',[0 0 1
1]);
set(background,'handlevisibility','off','visible','off');
uistack(background,'bottom');
axes(handles.axes7)
a = imread('Matematika.jpg');
imshow(a)
axis off
axes(handles.axes8)
b = imread('ITS.jpg');
imshow(b)
axis off

set(handles.startVideo,'Enable','off')

global videoSrc
videoSrc = imread('Daerah.jpg');
frame = im2uint8(videoSrc);
axes(handles.cameraAxes);
imshow(frame)

set(handles.startVideo,'Enable','off')

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes mycameragui wait for user
response (see UIRESUME)
uiwait(handles.myCameraGUI);

```

LAMPIRAN A (LANJUTAN)

```

% --- Outputs from this function are returned
to the command line.
function varargout =
myCameraGUI_OutputFcn(hObject, eventdata,
handles)

% Get default command line output from
handles structure
handles.output = hObject;
varargout{1} = handles.output;

% --- Executes on button press in startVideo.
function startVideo_Callback(hObject,
eventdata, handles)
global position tracks nextId countObject
frame centroids ...
bboxes mask assignments unassignedTracks
unassignedDetections...
panjang kecepatan ids x y hitung citra
hilang indx fram

tracks = initializeTracks();
nextId = 1;
countObject = 0;
kecepatan = 0;
hitung = 1; x=0; y=0; citra=0;
skala = str2num(get(handles.skala, 'String'));
% Start/Stop Camera
kondisi =
strcmp(get(handles.startVideo, 'String'), 'Star
t Video');
for indx = 1:panjang %video sampai selesai
if(kondisi)
% Camera is off. Change button string
and start camera.

```

LAMPIRAN A (LANJUTAN)

```

set(handles.startVideo,'String','Start
Video')
    frame = readFrame();
    frame =
insertShape(frame,'Rectangle',position,'color
','red',...
    'opacity',0.9);
    [centroids, bboxes, mask] =
detectObjects(frame);
    predictNewLocationsOfTracks();
    [assignments, unassignedTracks,
unassignedDetections] = ...
        detectionToTrackAssignment();
    updateAssignedTracks();
    updateUnassignedTracks();
    deleteLostTracks();
    createNewTracks();
    displayTrackingResults();
    if hilang(ids) == 0
        a(ids) = length(citra(ids,:));
        if isempty(find(x(ids,')==0))
            index=length(x(ids,:));
            for d = 1:index-1
                deltax(ids,d) = x(ids,d)
- x(ids,d+1);
                deltay(ids,d) = y(ids,d)
- y(ids,d+1);
                pixeldistance(ids,d) =
hypot(deltax(ids,d),deltay(ids,d));
            end
        else
            indexs = find(x(ids,')==0);
            index = indexs(1)-1;
            for d = 1:index-1
                deltax(ids,d) = x(ids,d)
- x(ids,d+1);

```


LAMPIRAN A (LANJUTAN)

```

    deltay(ids,d) = y(ids,d)
-   y(ids,d+1);
    pixeldistance(ids,d) =
hypot(deltax(ids,d),deltay(ids,d));
    end
end
    totaljarak(ids) =
sum(pixeldistance(ids,:));
    jarakkalibrasi(ids) =
totaljarak(ids);
    jaraksbn(ids) =
jarakkalibrasi(ids) * skala;
    waktu(ids) = index * fram;
    v(ids) = jaraksbn(ids) /
waktu(ids);
    kecepatan = v(ids) * 3600 /
100000;
set(handles.kecepatan,'String',kecepatan);
end
    axes(handles.cameraAxes);
imshow(frame);
axes(handles.cameraFG);
imshow(mask);

set(handles.kecepatan,'String',kecepatan);
else
    % Camera is on. Stop camera and
change button string.

set(handles.startVideo,'String','Start
Video')
end
end
end

```

LAMPIRAN A (LANJUTAN)

```

% --- Executes when user attempts to close
myCameraGUI.
function myCameraGUI_CloseRequestFcn(hObject,
eventdata, handles)
% Hint: delete(hObject) closes the figure
delete(hObject);
clear all;
close;

% --- Executes on button press in closeAll.
function closeAll_Callback(hObject,
eventdata, handles)
clear all;
close;

function Vid_path_Callback(hObject,
eventdata, handles)
global obj Path
Path = get(hObject, 'String');
if exist(Path, 'file')==2
    L = length(Path);
    if L>3
        extension = Path(L-3:L);
        if strcmpi(extension, '.avi')==1
            set(hObject, 'String', ...
                'Importing video, be
patient...', ...
                'ForegroundColor', 'red', ...
                'FontWeight', 'bold');
            drawnow;
            obj = setupSystemObjects();
            set(hObject, 'String', Path, ...
                'ForegroundColor', 'black', ...
                'FontWeight', 'normal');
            for i=L:-1:1

```

LAMPIRAN A (LANJUTAN)

```

        if strcmp(Path(i),'\')==1
            break;
        end
    end
    handles.FileName = Path(L-i:L-3);
    Video_Init(hObject,handles);
end
end
end

%% Video Initialization
function Video_Init(hObject,handles)
guidata(hObject,handles);
set(handles.startVideo,'Enable','on');
Video_Callback(handles.cameraAxes,[],handles)
;
set(handles.startVideo,'Enable','off')
global position
h = imrect;
setColor(h,'red');
position = wait(h);
if ~isempty(position)
    set(handles.startVideo,'Enable','on');
end

function Video_Callback(hObject, eventdata,
handles)
global panjang h w infomov
guidata(hObject,handles);
axes(handles.cameraAxes);
% Preallocate movie structure
mov(1:panjang) = struct('cdata', zeros(h, w,
3, 'uint8'), 'colormap', []);
for k = 1 : panjang
    mov(k).cdata = read(infomov, k);
end
I = mov(1).cdata;

```


LAMPIRAN A (LANJUTAN)

```

imshow(I);

% --- Executes during object creation, after
setting all properties.
function Vid_path_CreateFcn(hObject,
eventdata, handles)
if ispc &&
isequal(get(hObject, 'BackgroundColor'),
get(0, 'defaultUiControlBackgroundColor'))
    set(hObject, 'BackgroundColor', 'white');
end

% --- Executes on button press in Browse.
function Browse_Callback(hObject, eventdata,
handles)
[FileName, PathName] =
uigetfile({'*.avi', '*.avi AVI video
file'}, ...
    'Select input AVI video...');
if FileName~=0
    set(handles.Vid_path, 'String', strcat(PathName
,FileName));
    Vid_path_Callback(handles.Vid_path, [],
handles);
end

% --- Executes on button press in reset.
function reset_Callback(hObject, eventdata,
handles)
set(handles.Vid_path, 'string', '');
set(handles.kecepatan, 'string', '');
set(handles.x1, 'string', '');
set(handles.teth1, 'string', '');
set(handles.teth2, 'string', '');
set(handles.h, 'string', '');

```

LAMPIRAN A (LANJUTAN)

```

set(handles.x2, 'string', '');
set(handles.x3, 'string', '');
set(handles.skala, 'string', '');
set(handles.frame, 'string', '');
set(handles.pixel, 'string', '');

axes(handles.cameraAxes);
plot(0);
hold off;

axes(handles.cameraFG);
plot(0);
hold off;
clear all;

% --- Executes on button press in Hasil.
function Hasil_Callback(hObject, eventdata,
handles)
global fram
x1 = str2num(get(handles.x1, 'String'));
ting = str2num(get(handles.h, 'String'));
teth1 = str2num(get(handles.teth1, 'String'));
teth2 = str2num(get(handles.teth2, 'String'));
frame = str2num(get(handles.frame, 'String'));
pixel = str2num(get(handles.pixel, 'String'));

x2 = ting * (tan(teth2));
jarak1 = x1 - x2;
skala = jarak1 / pixel ;
fram = 1 / frame;

set(handles.x3, 'string', jarak1);
set(handles.x2, 'string', x2);
set(handles.skala, 'string', skala);

```

LAMPIRAN A (LANJUTAN)

A.2 Kode Program setupSystemObjects.m

```
function obj = setupSystemObjects(videoSrc)
global panjang Path h w infomov
obj.reader = vision.VideoFileReader(Path);
infomov = VideoReader(Path);
panjang = infomov.NumberOfFrames;
h = infomov.Height;
w = infomov.Width;
obj.detector =
vision.ForegroundDetector('NumGaussians', 3,
...
'NumTrainingFrames', 40,
'MinimumBackgroundRatio', 0.7);
obj.blobAnalyser =
vision.BlobAnalysis('BoundingBoxOutputPort',
true, ...
'AreaOutputPort', true,
'CentroidOutputPort', true, ...
'MinimumBlobArea', 400);
end
```

A.3 Kode Program detectObjects.m

```
function [centroids, bboxes, mask] =
detectObjects(frame)
global obj position
frame = imcrop(frame, position);
si = fspecial('average', 7);
se = strel('square', 15);
so = strel('square', 3);
sc = strel('square', 15);
```


LAMPIRAN A (LANJUTAN)

```

% mask = obj.detector.step(frame);
mask = step(obj.detector, frame);
mask = imfilter(mask, si);
mask = imerode(mask, se);
mask = imdilate(mask, se);
mask = imopen(mask, so);
mask = imclose(mask, sc);
mask = imfill(mask, 'holes');

[~, centroids, bboxes] =
obj.blobAnalyser.step(mask);
end

```

A.4 Kode Program displayTrackingResults.m

```

function displayTrackingResults()
global frame mask tracks position ids
kecepatan countObject x y hitung ...
hilang citra indx

frame = im2uint8(frame);
mask = uint8(repmat(mask, [1, 1, 3])) .* 255;
minVisibleCount = 5;

if ~isempty(tracks)
    reliableTrackInds = ...
        [tracks(:).totalVisibleCount] >
minVisibleCount;
    reliableTracks =
tracks(reliableTrackInds);

```

LAMPIRAN A (LANJUTAN)

```

% Display the objects. If an object has
not been detected
% in this frame, display its predicted
bounding box.

if ~isempty(reliableTracks)
% Get bounding boxes.
bboxes = cat(1, reliableTracks.bbox);

% kotak untuk frame
bbox = bboxes;
bbox(:,1) = bboxes(:,1) +
position(1);
bbox(:,2) = bboxes(:,2) +
position(2);
if bbox(3) >= 30 && bbox(3) <= 50
jenis = {'Objek'};
elseif bbox(3) > 50 && bbox(3) <= 150
jenis = {'Objek'};
else
jenis = {''};
end

% Get ids.
ids = int32([reliableTracks(:).id]);
a = ids(length(ids));
if(a > countObject)
countObject = a;
hitung = 1;
end

labels = strcat(jenis);
predictedTrackInds = ...
[reliableTracks(:).consecutiveInvisibleCount]
> 0;

```

LAMPIRAN A (LANJUTAN)

```

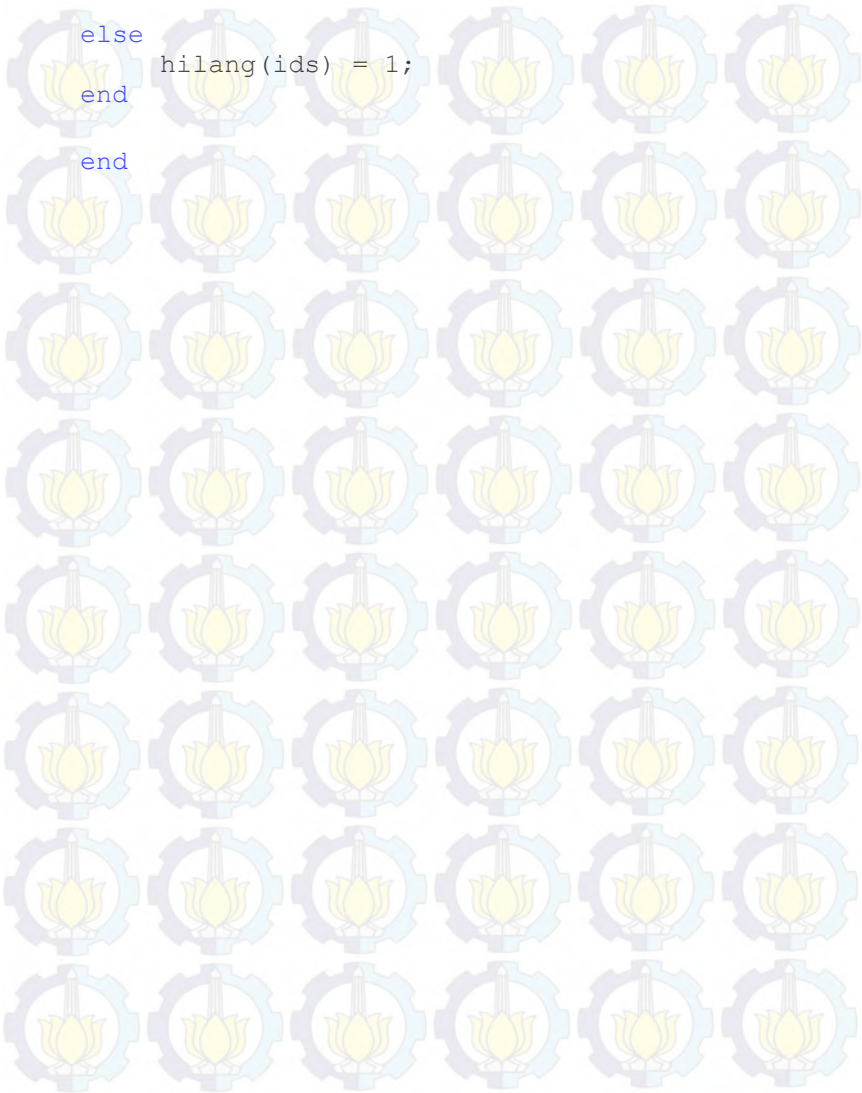
isPredicted = cell(size(labels));
kecepatan = num2str(kecepatan);
isPredicted(predictedTrackInds) = {
kecepatan} ;
labels = strcat(labels, isPredicted);
if predictedTrackInds == 0
hilang(ids) = 1;
x0 = bboxes(3)/2;
y0 = bboxes(4)/2;

% mencari centroid
x(ids, hitung) = bbox(1) + x0;
y(ids, hitung) = bbox(2) + y0;
citra(ids, hitung) = indx;
hitung = hitung + 1;
indx = indx + 1;
else
hilang(ids) = 0;
end

% Draw the objects on the frame.
frame = insertObjectAnnotation(frame,
'rectangle', ...
bbox, labels, 'Color', 'green' );
frame =
insertShape(frame, 'Rectangle', position, 'color
', ...
'blue', 'opacity', 0.9);

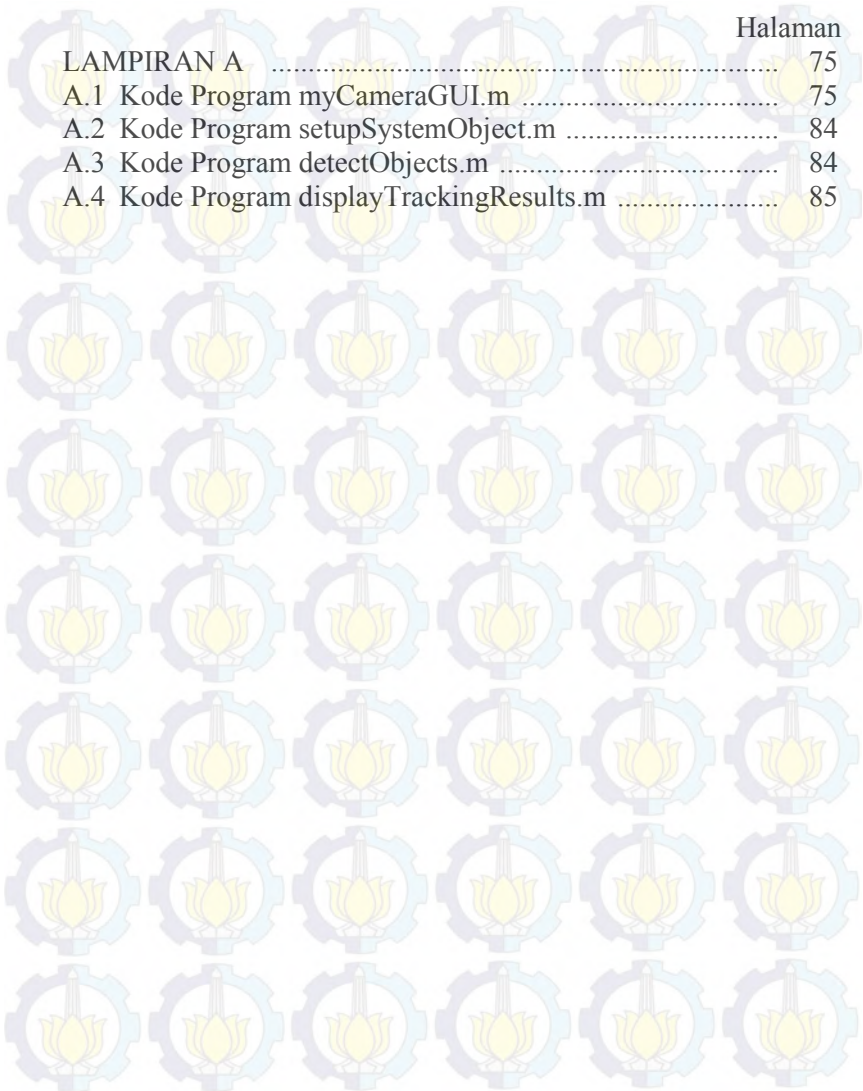
% Draw the objects on the mask.
mask = insertObjectAnnotation(mask,
'rectangle', ...
bboxes, labels, 'Color', 'red');
else
hilang(ids) = 1;
end

```


LAMPIRAN A (LANJUTAN)

DAFTAR LAMPIRAN

	Halaman
LAMPIRAN A	75
A.1 Kode Program myCameraGUI.m	75
A.2 Kode Program setupSystemObject.m	84
A.3 Kode Program detectObjects.m	84
A.4 Kode Program displayTrackingResults.m	85





“Halaman ini sengaja dikosongkan”

DAFTAR PUSTAKA

- [1] bps.go.id. (2013). “Perkembangan Jumlah Kendaraan Bermotor Menurut Jenis tahun 1987-2013”. http://www.bps.go.id/tab_sub/view.php?tabel=1&id_subyek=17¬ab=12, *posted*: 2013. Diakses pada tanggal 16-02-2015
- [2] Lin, Huei-yung. Li, Kun-Jhih. Chang, Chia-Hong. (2008). “Vehicle Speed Detection from a Single Motion Blurred Image”. **Image and Vision Computing** **26**, Hal 1327-1337
- [3] Pornpanomchai, C., and Kongkittisan, K. (2009). “Vehicle Speed Detection System”. **IEEE International Conference on Signal and Image Processing Applications**, Hal. 135-139.
- [4] Rad, A. G, Deghani. A, Karim. M. R. (2010). “Vehicle Speed Detection in Video Image Sequence using CVS Method”. **International Journal of the Physical Sciences Vol. 5(17)**, Hal. 2555-2563
- [5] Rafael C. Gonzales, Richard E. Woods. (2002). “Digital Image Processing”. **United States of America: Tom Robbins Publisher.**
- [6] Al Bovik. (2000). “Handbook of Image and Video Processing”. **San Diego: Academic Press Publisher.**
- [7] Bharti, T. Tejinder. (2013). “Background Subtraction Techniques-Review”. **International Journal of Innovative Technology and Exploring Engineering, Vol. 2, Issue 3**, Hal. 166 - 168.
- [8] Alper Yilmaz, Omar Javed and Mubarak Shah. “Object tracking: A survey”. **ACM Comput. Surv.**, 38(4):13,2006.

- [9] Lazuardi, R. Arif Firdaus. (2014). “Penghitungan Kendaraan Bergerak Berbasis Algoritma Background Subtraction Menggunakan Metode Gaussian Mixture Model”. **Tugas Akhir. Jurusan Matematika ITS.**
- [10] C. Stauffer, W.E.L. Grimson. (1999). “Adaptive Background Mixture Models for Real-time Tracking”. **The Artificial Intelligence Laboratory, Cambridge.**
- [11] Hariyanto, Zamroji. (2015). “Klasifikasi Jenis Kendaraan Bergerak Berbasis *Geometric Invariant Moment*”. **Tugas Akhir. Jurusan Matematika ITS**

BIODATA PENULIS



Farah Fajriyah, lahir di kota Gresik, 04 Januari 1994. Penulis berasal dari Kota Gresik, bertempat tinggal di Jalan Sultan Agung Timur RT 03 RW 04 Desa Sungonlegowo Kec. Bungah Kab. Gresik. Pendidikan formal yang pernah ditempuh yaitu pendidikan dasar di SDN Sungonlegowo II Bungah Gresik sepanjang tahun 1999-2005, dan pendidikan menengah pertama di SMPN 1 Bungah Gresik pada tahun 2005-2008. Dan melanjutkan pendidikan menengah atas sejak tahun 2008-2011. Mulai menyandang status mahasiswa di Jurusan Matematika FMIPA ITS pada tahun 2011 melalui jalur penerimaan SNMPTN Undangan dan menyelesaikan studinya di tahun 2016. Semasa di bangku perkuliahan, penulis mengikuti beberapa organisasi kemahasiswaan. Diantaranya adalah Himpunan Mahasiswa Matematika (Himatika) ITS, UKM Pramuka, dan UKM Cinta Rebana ITS. Di akhir-akhir tahun masa perkuliahan, penulis aktif di Lab. Ilmu Komputer Matematika untuk memperdalam *skill* berbasis keilmuan *programming*. Penulis dapat dihubungi lebih lanjut di farahfajriyah41@gmail.com.



“Halaman ini sengaja dikosongkan”